Noname manuscript No.
(will be inserted by the editor)

On the Behavior of Parallel Island Model Genetic Algorithms

Lucas A. da Silveira · Jessé B. de Barros · José L. Soncco-Álvarez · Thaynara A. de Lima · Carlos H. Llanos · Mauricio Ayala-Rincón

Received: date / Accepted: date

Abstract This work explores migration policies over different communication topologies in synchronous and asynchronous parallel island models used to improve speed-up and accuracy in genetic algorithms. This paper aims to explain the adequacy of such models from the a general perspective, trying to answer questions such as which is the best manner to implement genetic algorithms in parallel island models. The quality of the solutions and the running time provided by parallel island models are evaluated from the perspective of 4 different $\mathcal{NP}$-hard problems: reversal and translocation evolutionary distance, task mapping and scheduling and, N-Queens. The results show that no model provides better speed-up and accuracy in general. However, the experiments make evident that after parameters tuning of the breeding cycle and island migration parameters, all parallel models reach not only good speed-ups but also more accurate solutions than the sequential genetic algorithm. In general, synchronous models provide the best solutions while asynchronous models deliver the best run-times, but there are scenarios where variations occur.

Keywords Genetic algorithms · Parallel genetic algorithms · Parallel island models · Migration parameters.

Lucas A. da Silveira[†], Mauricio Ayala-Rincón[†,‡]
Depts. of [†]Computer Science and [‡]Mathematics, Univ. de Brasília, Brazil
E-mail: lucasangelosilveira2018@gmail.com, ayala@unb.br

Jessé B. de Barros, Carlos H. Llanos
Dept. of Mechanical Engineering, Univ. de Brasília, Brazil
E-mail: jesseh.barreto@gmail.com, llanos@unb.br

José L. Soncco-Álvarez
Dept. of Informatics, Univ. Nacional de San Antonio Abad del Cusco, Peru
E-mail: jose.soncco@unsaac.edu.pe

Thaynara A. de Lima
Dept. of Mathematics and Statistics, Univ. Fed. de Goiás, Goiânia, Brazil
E-mail: thaynaradelima@ufg.br

1 Introduction

Parallel islands models (PIMs) have received considerable attention because of their potential to increase speed-up and accuracy [48], [57], [62]. The communication between islands is established by a *migration topology*. The strategy is to map islands in processors, such that each island runs an evolutionary algorithm (EA) and occasionally shares genetic material with other islands [64]. A simple form of sharing is to send a copy of an individual from a local island to another target island through a process called migration. The movement of individuals is defined by a migration policy that involves parameters that need to be carefully calibrated and that are related to considerations such as: *number of individuals* that are sent (emigrants) and received (immigrants) from a local to a target island; *migration interval* or generations between one migration and another; *emigration policy* that defines whether emigrating individuals are just *moved* or *cloned* and sent to a target island, among others.

Different PIMs architectures determined by their migration topology are studied considering *synchronous* and *asynchronous* migration between their islands. In the former architectures, islands evolve simultaneously, while in the latter ones, the migration is not related to the state of evolution in all the islands. The asynchronous behavior is typically found in migration in nature, where different environmental factors are responsible for the differences in the speed of evolution. Also, topologies can be *static* or *dynamic*. In a static topology the communication between islands is specified at the beginning of execution and remains unchanged, whereas in a dynamic topology it can change.

This work aims to provide a robust feedback about the adequacy of migration policies in PIMs from the Genetic Algorithm (GA). The GA is a metaheuristic inspired by Charles Darwin's theory of natural evolution that has been successfully applied in many optimization and machine learning problems. Implementing PIMs using GA is not difficult, since GAs are naturally prone to parallelism [8], however, the difficulty lies in how to manipulate the breeding cycles and migration parameters to extract accuracy from the implementations.

Our research consists of investigating different PIMs in the literature that were enriched with synchronous and asynchronous migration configurations, considering a variety of scenarios obtained when we modify the migration parameters on the perspective of run-time and accuracy. The implemented PIMs use 12 and 24 islands for a variety of static and dynamic topologies. To guide our investigation four case-studies related with $\mathcal{NP}$-hard problems are used: unsigned reversal distance (URD), unsigned translocation distance (UTD), task mapping and scheduling (TMP), and N-Queens. In the experiments phase, several input sets are used for each case-study.

The feedback obtained from the experiments does not point to a generic model that offers better speed-up and/or better accuracy for all case-studies. After parameters tuning, all PIMs provided better solutions than the sequential GA, being the best results obtained by synchronous models except for the N-Queens problem. The speed-up is strongly influenced by the used parameters; indeed, the results show that asynchronous PIMs present better speed-ups than synchronous models, but this is not a rule. Also, in most cases 24-island models process inputs faster than their 12-island versions, but this is not the rule in PIMs implemented for the N-Queens problem and asynchronous PIMs for TMP. From the

point of view of accuracy, synchronous static 12-island models present, in general, the most competitive solutions for URD, while for UTD, both static and dynamic 12-island models computed the best solutions. For TMP the best solutions were delivered by synchronous dynamic 24-island model and for N-Queens the best results were through an asynchronous dynamic 12-island model. The results were checked through statistical tests.

The paper is organized as follows: initially, Sec. 2 presents the case-studies, which can be omitted if the reader is familiarized with them, and the terminology about the migration parameters in PIMs; Sec. 3 presents the implemented PIMs; afterwards, Sec. 4 presents experiments, and Section 5 discusses them; finally, after Sec. 6 on related work, Sec. 7 concludes and proposes future work.

Note to the reviewers The submission includes extended descriptions of the case-studies (Subsections 2.1 to 2.4), related work (Section 6), and statistical test (Subsection 5.4) that can be substantially shortened whenever the reviewers find it convenient. The source code and data used in the experiments are available at `http://genoma.cic.unb.br`.

2 Case-Studies and Parallel Island Models (PIMs)

In this section, the four case-studies used for experiments are briefly described: unsigned reversal and translocation distance, task mapping and scheduling, and N-Queens.

2.1 Reversal Distance Problem

In bioinformatics, a unichromosomal genome with n genes can be represented as a permutation $\pi = (\pi_1, \pi_2, ..., \pi_n)$, where $\pi(i) = \pi_i$, $1 \leq i \leq n$ and $\pi_i \in \{1, \ldots, n\}$. A reversal $\rho^{j,k}$, for $1 \leq j \leq k \leq n$ is an operation on π that inverts the sub string 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 $\pi = (1, 4, 3, 5, 8, 2, 7, 6)$, $\rho^{5,7}(\pi)$ gives $\pi' = (1, 4, 3, 5, 7, 2, 8, 6)$. The *reversal distance problem* consists in to find a shortest sequence of reversals $\rho_1, \cdots, \rho_l$ needed to transform a permutation π into another permutation σ. There are two different types of permutations: signed and unsigned. In the unsigned case, genes are abstracted without any orientation, while in the signed one, each π_i has a positive or negative sign reflecting the orientation of the gene within the genome. A reversal acts over a signed permutation by inverting also the orientation of the genes in a specific sub string. By simple algebraic properties of permutations, the reversal distance problem results equivalent to transform an arbitrary permutation π into the identity permutation $\imath$ (that is the permutation sorted in increasing order and, for the signed case, in which each gene has a positive orientation) by a minimum number of reversals; such number is called the *reversal distance* of π and the so-called *sorting by reversals problem* consists in determining the reversal distance of a permutation [4]. The signed version of this problem was proved to be in $\mathcal{P}$, namely, an $O(n^2)$ algorithm was first given by Hanenhalli and Pevzner in [32] and further, a linear one in [3] was proposed by Bader et al., while the unsigned one was proved to be $\mathcal{NP}$-hard by Caprara in [10]. The interesting case-study considered in this work is the *Unsigned Reversal Distance* (URD) problem, where

permutations have no orientation. The signed case is abbreviated as SRD and also has been extensively studied in the field of combinatorics and algorithmics of permutations (e.g., [30], [42]). In evolutionary and genetic approaches, the fitness function to solve URD is computed applying SRD solution approaches as [3] (that is the one used in our procedures) as described in [56].

2.2 Translocation Distance Problem

The *Translocation Distance problem* consists in computing the length of a minimal sequence of operations called *translocations* that transform one multi-chromosome genome into another. As for the reversal distance problem signed and unsigned genomes are considered. An unsigned genome is composed by a set of chromosomes $\{X_1, \cdots, X_t\}$, where each X_v, for $1 \leq v \leq t$, consists of a gene sequence such that each gene x_i is represented by a different natural number appearing only once in the genome. Consider two chromosomes $X = (x_1, x_2, \ldots, x_k)$ and $Y = (y_1, y_2, \ldots, y_m)$ of an unsigned genome. There are two kinds of translocations:

- A *prefix-prefix* translocation $\rho_{pp}(X, Y, i, j)$, $1 < i \leq k$, $1 < j \leq m$ that transforms the chromosomes X and Y into two new chromosomes $(x_1, \ldots, x_{i-1}, y_j, \ldots, y_m)$ and $(y_1, \ldots, y_{j-1}, x_i, \ldots, x_k)$.
- A *prefix-suffix* translocation $\rho_{ps}(X, Y, i, j)$, $1 < i \leq k$, $1 < j \leq m$ that transforms the chromosomes X and Y into two new chromosomes $(x_1, \ldots, x_{i-1}, y_{j-1}, \ldots, y_1)$ and $(y_m, \ldots, y_j, x_i, \ldots, x_k)$.

Formally, the *Unsigned Translocation Distance* (UTD) problem consists in finding the minimum number of translocations for transforming an unsigned genome A into another genome B, where A and B have the same (number of) genes n, and number of chromosomes N. Without loss of generality, when the genomes A and B have the same genes and number of chromosomes, the genes of A and B can be renamed, so that B can be rewritten as an identity. Thus, for the experiments, it is assumed that genome B is an identity genome, that is a genome with all its genes sorted in increasing order; for instance, $A = \{(1, 3, 7)(5, 2, 6, 4)\}$ and $B = \{(1, 2, 3, 4)(5, 6, 7)\}$. The UTD problem is the one addressed in this paper, and was shown to be $\mathcal{NP}$-hard by Zhu and Wang in [67]. Section 4.3 presents the experiments with the proposed islands models for this case-study. As for SRD, the Signed Translocation Distance problem (STD) considers signed genomes. Let n be the number of genes in the genome; Hannenhalli in [31] provided the first polynomial-time algorithm with minimum translocation distance in $O(n^3)$ and the best known algorithm, proposed by Bergeron et al. in [6], computes the minimum translocation distance in $O(n)$. In genetic and evolutionary approaches, the fitness function to solve UTD problems is given by a STD solution as [6] that is the one applied in our approach (e.g., [53]).

2.3 Task Mapping and Scheduling

The *Task Mapping Problem* (TMP), also known as *Task Scheduling*, consists of mapping tasks of a *Real-Time Application* (RTA) onto one of the multiple processor cores of a *Real-Time System* (RTS). The correctness of a RTS depends

upon the logical soundness of its results and its compliance with time restrictions (deadlines).

The focus of the case-study is the application of PIMs in TMP for a type of multiple processor system called *Multiprocessor System-on-Chip* (MPSoC) as presented by Wolf et al. in [65]. The MPSoC used in this case-study has a communication architecture that consists of a *Network-on-a-Chip*, a well-established intra-chip communication architecture paradigm as presented in [14].

Due to the real-time setting in case-study, the *Network-on-a-Chip* used fits the category of a *Real-Time Network-on-a-Chip* (RTNoC) as defined by Hesham et al. in [33], with a wormhole packet switching strategy and virtual channels allowing the preemption of data flows during their transmission by ones with higher priority level.

In order to perform a static schedulability analysis of possible mapping solutions, this case-study uses a specific model for both the RTNoC-based MPSoC platform and the RTA mapped onto it that is similar to the one present by Indrusiak in [35]. The platform model is a triple $\Psi = \langle \Pi, R, \Lambda \rangle$, where Π is a set of homogeneous processing elements connected to its network interface elements, $\Pi = \{\pi_1, ..., \pi_n\}$; R is a set of routing elements to switch messages $R = \{\rho_1, \rho_2, ..., \rho_n\}$ that connect elements in Π and themselves using unidirectional links, $\Lambda = \{\lambda_{\pi_1,\rho_1}, \lambda_{\rho_1,\pi_1}, \lambda_{\rho_1,\rho_2}, ..., \lambda_{\rho_n,\rho_{n-1}}\}$, in a regular mesh-grid topology. Figure 1, adapted from the work proposed by Indrusiak in [35], illustrates a NoC-based platform with its components interconnected in a mesh-grid fashion.

The application model comprises a set of tasks Γ, such that $\tau \in \Gamma$ is given as a tuple $\langle C, T, D, P, \phi \rangle$ formed by the worst-case running time, inter-arrival period, relative deadline time, priority-level, and a transmitted message ϕ, respectively. At the end of its execution, the task τ may transmit a message ϕ towards another task. A message ϕ is represented as a tuple $\langle \tau_d, L, Z \rangle$, where τ_d is the destination task, L the basic latency of message transmission and Z the message size in bits. The basic latency L_i for a message ϕ_i is calculated using Eq. (1) and depends upon the mapping of the source task τ_i that defines the route of links in which this message will travel across the platform. This route of links is a set $route(\phi_i)$. The basic latency calculation also depends on specific parameters such as routers and links maximum latencies per flit, L_ρ and L_λ, respectively.

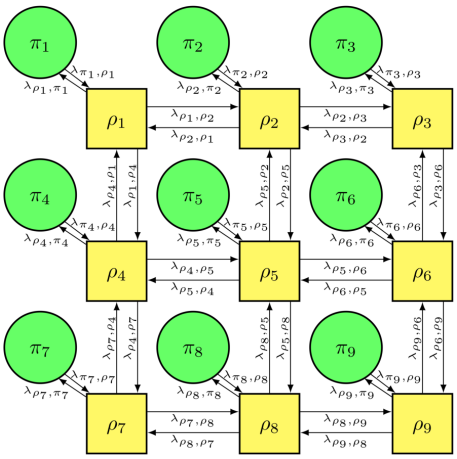


Fig. 1: 3×3 mesh-grid RTNoC platform. Circles and squares represent, respectively, processor elements and routers.

$$L_i = |route(\phi_i)| \cdot (L_\lambda + L_\rho) - L_\rho + Z_i L_\rho \tag{1}$$

The schedulability analysis considers the utilization factor of processor cores and links. The utilization factor for a processor core, U_π, presents the ratio in which the

processor is used by the tasks that are mapped onto it and is expressed by Eq. (2) (according to [44]), where $map^{-1}(\pi)$ is the set of tasks mapped to processor π.

The same principle is used to evaluate the utilization rate of links in the RT-NoC to check whether they do not exceed their maximum communication bandwidth. The link utilization is represented by U_λ, and is calculated by Eq.(3), where $map^{-1}(\lambda)$ is the set of messages that are transmitted over the link λ.

$$U_\pi = \sum_{\tau_i \in map^{-1}(\pi)} \frac{C_i}{T_i} \qquad (2) \qquad\qquad U_\lambda = \sum_{\phi_i \in map^{-1}(\lambda)} \frac{L_i}{T_i} \qquad (3)$$

This simple analysis is necessary to avoid non-schedulable solutions that are those for which the resource demands exceed the available resource capacity, implying that tasks and messages do not comply with the timing restrictions. This holds for any given processor π if its utilization factor is such that $U_\pi > 1$ and for any given link λ such that $U_\lambda > 1$. However, the utilization analysis is not sufficient because even in cases where all processors and links are not over-utilized, the task mapping may not be fully schedulable depending on the task set in the mapped RTA. Even though these tests are necessary but not sufficient, they are attractive as methods to quickly identify whether mappings suit the time requirements.

Given a set of tasks Γ and a platform Ψ, a task mapping solution can be encoded as an n-dimensional vector of positive integers $\boldsymbol{x}$, where each jth component, x_j, represents the index of the processor in the platform Ψ responsible for processing the task $\tau_j \in \Gamma$. The search-based optimization for a task mapping solution that reduces the number of overused resources is expressed in Eq. (4), where f_{util} (see Algorithm 1) is a function that calculates the number of processors and links over burdened in the system given a task placement $\boldsymbol{x}$ using Eqs. (2) and (3). The optimization goal using f_{util} as a fitness function is to find a task placement $\boldsymbol{x}^*$ that respects processors and links maximum capacities.

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ f_{util}(\boldsymbol{x}) \qquad (4)$$

The considered TMP case-study is $\mathcal{NP}$-complete similar to the knapsack problem as given by Garey and Johnson in [26]; thus, a brute force approach is unfeasible. Indeed, a case where n tasks are being mapped onto a system with m processors gives a search-space with m^n possible task mapping solutions. The study of TMP in multiple processors RTSs is fundamental to the embedded devices research field since their application is required in critical systems. Another aspect is that the advance of chip manufacturing allied with the increasing computing requirements of current RTSs makes it natural for embedded devices to use a large number of processors in the same chip to tackle massively parallel problems. Section 4.4 presents the experiments, with the proposed islands models using the TMP problem as a case-study.

2.4 N-Queens

The *N-Queens problem* [27] is an expansion of the 8-Queens placement problem that aims to place n queen chess pieces in an $n \times n$ chessboard in a configuration in which no two queens can attack each other; i.e., a configuration where no placed n points in an $n \times n$ grid has the same row, column, or diagonals. The problem

Algorithm 1 Fitness function $\boldsymbol{f_{util}}$

INPUT: Tasks Placement ($\boldsymbol{x}$)
OUTPUT: Number of Over Capacity Elements

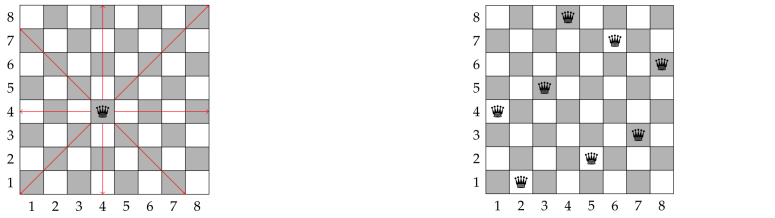
procedure $\boldsymbol{f_{util}}(\boldsymbol{x})$
  Check Task Mapping $\boldsymbol{x}$
  $overcap = 0$
  for $p = 1$ to $|\Pi|$ **do**
    Calculate U_{π_p} ▷ Eq. (2)
    if $U_{\pi_p} > 1$ **then**
      overcap = overcap + 1
    end if
  end for
  for $l = 1$ to $|\Lambda|$ **do**
    Calculate U_{λ_l} ▷ Eq. (3)
    if $U_{\lambda_l} > 1$ **then**
      overcap = overcap + 1
    end if
  end for
return *overcap*
end procedure

is divided into two types: (a) the search of a configuration in which no queen attack each other (*witness problem*), and (b) the search for all witness solutions of a board configuration (*counting problem*). These problems are related with the N-Queens completion problem that consists in completing a configuration from a given partial solution and is known to be both $\mathcal{NP}$-complete and $\#\mathcal{P}$-complete (see [27]). The case-study in this work is the *witness problem* that is commonly used as a benchmark in different machine learning-based methods; for example, it has been used in constructive backtracking algorithms [59] and EA based metaheuristics such as GA [40] and PSO [66].

The search for a solution for the N-Queens placement corresponds to a minimization problem according to Eq. 5. Considering the search space is given by n different positions in an $n \times n$-chessboard it size is given by $(n^2)!/(n!\,(n^2-n)!)$ and the measure of each possible solution is given by the number of attacking movements between queens at these positions. Here, each candidate placement solution is encoded as an n-dimensional vector $\boldsymbol{x}$, where the pair (j, x_j), for j an index of $\boldsymbol{x}$, represent the placement of a queen onto the jth row and x_jth column of the chessboard. Notice that in this form the search space is reduced to a set S of n^n possible placements. The goal is to minimize a function $f_{attack} : \mathrm{S} \to \mathbb{N}$ that maps each $\boldsymbol{x} \in \mathrm{S}$ into the number of queen pieces under attacking movements given by a placement $\boldsymbol{x}$. If $f_{attack}(\boldsymbol{x}) = 0$ then $\boldsymbol{x}$ is a solution that satisfies the N-Queens placement problem. Figure 2a illustrates the placement of a single queen piece at (4, 4) position on an 8×8 board and Figure 2b shows a placement where none of the 8 queen pieces attack each other.

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ f_{attack}(\boldsymbol{x}) \tag{5}$$

The function f_{attack} is linearly computed according to Algorithm (pseudocode) 2, where given an $\boldsymbol{x}$ placement, it counts the number of queens in each column, principal and secondary diagonals, and then accumulates the number of attacks in each column, and principal and secondary diagonals.



(a) Movement of a queen chess piece.

(b) A placement $\boldsymbol{x}$ where $f_{attack}(x) = 0$

Fig. 2: Illustration of queen chess pieces interaction on an 8×8 board.

Algorithm 2 N-Queens Attack Counter

INPUT: Placement ($\boldsymbol{x}$ with $|\boldsymbol{x}| = n$)
OUTPUT: Number of Attacks ($attacks$)

1: **procedure** $NQueensAttacks(\boldsymbol{x})$
2: queens-col$[1..n]$, queens-dp$[1-n..n-1]$, quens-ds$[2..2n]$
3: $attacks = 0$
4: **for** $j = 1$ to n **do**
5: queens-col$[x_j]$++
6: queens-dp$[x_j - j]$++
7: queens-ds$[x_j + j]$++
8: **end for**
9: **for** $j = 1$ to n **do**
10: **if** queens-col$[j] \geq 2$ **then**
11: $attacks$+= queens-col$[j]$ (queens-col$[j]$ $-1)/2$
12: **end if**
13: **end for**
14: **for** $j = 2$ to $2n$ **do**
15: **if** queens-ds$[j] \geq 2$ **then**
16: $attacks$+= queens-ds$[j]$ (queens-ds$[j]$ $-1)/2$
17: **end if**
18: **if** queens-dp$[j-1-n] \geq 2$ **then**
19: $attacks$+= queens-dp$[j-1-n]$ (queens-dp$[j-1-n]$ $-1)/2$
20: **end if**
21: **end for**
22: **return** $attacks$
23: **end procedure**

2.5 Parallel Island Models (PIMs)

In PIMs, each island runs an EA with its population evolving independently and, periodically individuals are shared between islands by migration. The migration process can be *synchronous* occurring when all islands reach the number of generations controlled by the *migration interval*, or *asynchronous*, occurring when each island controls its migration process through the migration interval independently of each other.

An organizational architecture known as *migration topology* is responsible for establishing a communication model between islands, where at the given migration interval, individuals are selected in local islands and sent to target islands. For each island, the population is bi-partitioned ranking individuals from the one with the best fitness to the one with the worst fitness. This ranking is considered to select

individuals in the migration process. The *migration parameters* (for short, Mig_P) are abstracted below as argued by Sudholt in [60].

- *NumMigIndividuals*: number of migration individuals.
- *TypeEmIndividual*: type of individuals selected for emigration among 1. **best**, 2. **worst**, and 3. **random**.
- *EmPolicy*: emigration policy to decide whether emigrating individuals in the local island are 1. **cloned** or 2. **removed** to be sent to the target island.
- *TypeImIndividual*: type of individuals in the target island that are replaced by immigrants among 1. **worst**, 2. **random**, and 3. **similar**, where by similar individuals, it is understood replacement of individuals with the same fitness rank than the immigrants.
- *MigrationInterval*: percentage of breeding cycles (number of generations) in which migration occurs.

When these parameters are properly configured, it is expected that accuracy and speed-up in PIMs be improved.

3 Implemented Parallel Island Models

The PIMs use homogeneously a simple genetic algorithm ($\mathcal{GA}_{\text{S}}$), which is responsible for the evolution of the native individuals in each island. The initial population of $\mathcal{GA}_{\text{S}}$ consists of individuals randomly generated with a breeding cycle fixed in n generations. The individuals are represented as an array of integers, and the representation of these individuals varies from problem to problem. Considering URD and UTD problems, from an unsigned genome given as input a population of signed genomes is randomly generated. For example, the signed genomes $(+1, -3, +2)(+4, -7, +6, +8)$ and $(+1, -2, -4, -3, +5)$ could be obtained from given inputs $(1, 3, 2)(4, 7, 6, 8)$ and $(1, 2, 4, 3, 5)$, respectively. Regarding the TMP and the N-Queens problems, each individual is respectively a task mapping solution $\boldsymbol{x}$ or a candidate placement solution $\boldsymbol{x}$. The representation of individuals for all these problems is indeed very similar (integer vectors), which simplifies the adaptation of PIMs to process each different case-study.

The fitness function is linked to the applied case-study: for UTD, the algorithm to solve STD proposed in [6], which runs in $O(n)$ is used; for URD, the linear algorithm to solve SRD introduced in [3] is applied; for TMP, an utilization-based test is applied, as proposed in [35], that has complexity $O(n_\lambda^2 + n_\pi^2)$, where n is the number of tasks in the used RTA model and n_λ and n_π are the number of links and processing cores in the platform, respectively; finally, the algorithm used for evaluating the fitness function for N-Queens counts the queens in each column, principal and secondary diagonal and accumulates the number of attacks, which was implemented with complexity linear.

Algorithm 3 presents the pseudo code for $\mathcal{GA}_{\text{S}}$. The breeding cycle works with three procedures: *Crossover, Mutation* and *Replace* and receives the input $\boldsymbol{x}$ according to the problem, and genetic *breeding cycle parameters* for the percentage of *selection* and *replacement*, and probability of application of *mutation* and *crossover*, which are stored in the BC_P structure.

According to the *selection* parameter, a percentage of the best individuals is selected from the population Pop_{GA} and stored in a vector M_{best}. The *Crossover*

procedure, given in Algorithm 4, works on M_{best}. No all individuals in M_{best} are subjected to crossing since the value of the *crossover* rate parameter restricts crossing of parent individuals. For those pairs individuals selected for crossing, they are fragmented using a single cut-off point, producing four fragments that later combined produce two offspring stored in the vector Pop_d. At the end of Algorithm 4, Pop_d contains all offspring.

Algorithm 3 Pseudo code for $\mathcal{GA}_{\mathrm{S}}$

INPUT: $\boldsymbol{x}$, according to the problem, *maxIteration*, and breeding cycle parameters BC_P
OUTPUT: Solution to $\boldsymbol{x}$

1: **procedure** $GA(X, BC_P)$
2: Generate the initial population Pop_{GA} from $\boldsymbol{x}$
3: Compute fitness of Pop_{GA}
4: **for** $i = 1$ to *maxIteration* **do**
5: $M_{best} \leftarrow$ Select the $BC_P.selection * |Pop_{GA}|$ best individuals in Pop_{GA}
6: $Pop_d \leftarrow Crossover(M_{best}, Pop_{GA}, BC_P.crossover)$
7: $Pop_d \leftarrow Mutation(Pop_d, BC_P.mutation)$
8: $Pop_{GA} \leftarrow Replace(Pop_{GA}, Pop_d, BC_P.replacement)$
9: **end for**
10: **return** *best solution*
11: **end procedure**

Algorithm 4 Pseudo code for the crossover operator

INPUT: M_{best}, Pop_{GA} and *crossover*
OUTPUT: Pop_d are offspring individuals

1: **procedure** $Crossover(\ M_{best}, Pop_{GA}, crossover\ )$
2: $i \leftarrow 1$
3: $Pop_d \leftarrow [\,]$ ▷ empty vector
4: **while** $i < |M_{best}|$ **do**
5: $p_1 \leftarrow M_{best}[i]$
6: $i \leftarrow i + 1$
7: $p_2 \leftarrow M_{best}[i]$
8: $rand \leftarrow Random(1, 100)$
9: **if** $rand \leq crossover * 100$ **then**
10: $cut \leftarrow Random(1, |p_1|-1)$
11: $Ind_1 \leftarrow Copy(p1, 1, cut - 1) \circ Copy(p_2, \text{cut}, |p_2|)$ ▷ $Copy(p, i, j)$ copies p from position i to j and $\circ$ concatenates vectors
12: $Ind_2 \leftarrow Copy(p_2, 1, cut - 1) \circ Copy(p1, cut, |p1|)$
13: $Pop_d \leftarrow Pop_d \circ [Ind_1, Ind_2]$
14: **end if**
15: **end while**
16: **return** Pop_d
17: **end procedure**

Algorithm 5 tries to apply mutation to each offspring in Pop_d. The probability of mutation should be small to avoid compromising the inherited genetic material. It is important to emphasize that mutating a gene varies from problem to problem. For UTD and URD, a mutation only inverts the signal of a gene; for N-Queens, it consists of allocating the queen in a random position on the board; and for TMP, it

consists of randomly allocating a task in a processor. A mutation of an individual replaces the original individual only if its fitness is better.

Finally, Algorithm 6 includes descendants in Pop_d into the current population. The percentage of individuals to be eventually replaced from the current population, Pop_{GA}, is given by the *replacement* parameter. The fitness of each individual in Pop_d is compared with the fitness of a randomly selected individual from the *replacement* percent of individuals in Pop_{GA}, and the individual in Pop_{GA} is replaced by the descendant only if its fitness is worse; otherwise, the descendant is discarded.

Algorithm 5 Pseudo code for the mutation operator

INPUT: Pop_d, *mutation*
OUTPUT: mutated offspring individuals Pop_d

procedure *Mutation*($\boldsymbol{Pop_d}$, ***mutation***)
  $i \leftarrow 1$
  while $i \leq |Pop_d|$ **do**
    $ind_i \leftarrow Pop_d[i]$
    $j \leftarrow 1$
    while $j \leq |ind_i|$ **do**
      $rand \leftarrow Random(1, 1000)/1000$
      if $rand \leq mutation$ **then**
        $ind_i[j] \leftarrow ind_i[j] * -1$ ▷ applies to UTD and URD problems
        $ind_i[j] \leftarrow Random(1, numberQueens)$ ▷ applies to N-Queens problem
        $ind_i[j] \leftarrow Random(1, numberProcessors)$ ▷ applies to TMP problem
      end if
      $j \leftarrow j + 1$
    end while
    if $Fitness(ind_i) < Fitness(Pop_d[i])$ **then**
      $Pop_d[i] \leftarrow ind_i$
    end if
    $i \leftarrow i + 1$
  end while
return Pop_d
end procedure

Algorithm 6 Pseudo code for the replacement operator

INPUT: Pop_{GA}, Pop_d, *replacement*
OUTPUT: updated Pop_{GA}

procedure *Replace*($\boldsymbol{Pop_{GA}}$, $\boldsymbol{Pop_d}$, ***replacement***)
  $i \leftarrow 1$
  while $i \leq |Pop_d|$ **do**
    $ind_i \leftarrow Pop_d[i]$
    $remainder \leftarrow |Pop_{GA}| - |Pop_{GA}| * replacement$
    $j \leftarrow Random(1, |Pop_{GA}| * replacement) + remainder$
    $ind_j \leftarrow Pop_{GA}[j]$
    if $Fitness(ind_j) < Fitness(ind_i)$ **then**
      $Pop_{GA}[j] \leftarrow ind_i$
    end if
    $i \leftarrow i + 1$
  end while
return Pop_{GA}
end procedure

As suggested in [51], to build the initial population in all PIMs, the population Pop_{GA} of $\mathcal{GA}_{\text{S}}$ was partitioned into populations of equal size and distributed among the islands as seen in Algorithm 7.

Algorithm 7 Generation of the island populations

INPUT: $\boldsymbol{X}$, No. of islands N_I and population size P_S
OUTPUT: Each island with its population

procedure $InitPopulation(\boldsymbol{X}, N_I, P_S)$
    Island 1 generates P_S individuals
    for $i = 2$ to N_I **do**
        Island 1 sends P_S/N_I individuals to Island i
    end for
    for $i = 2$ to N_I **do**
        Island i receives individuals from Island 1
    end for
end procedure

In the following, for short the standard parameters of $\mathcal{GA}_{\text{S}}$ are denoted as GA_P.

3.1 PIMs with Static Topology

The models use bidirectional static communication topologies: torus, binary tree, complete graph and a unidirectional ring topology (see Figure 3). Torus differs from the net topology in its neighborhood. All nodes on torus are adjacent to 4 other nodes, while on the net internal and boundary nodes are adjacent to 3 and 4 nodes, respectively. Regarding communication balancing, interactions between nodes (representing islands) are larger in the torus topology, in which all nodes communicate with four other nodes, causing greater spread of genes through the neighborhood when compared with tree, ring and net topologies. In the tree topology, internal nodes communicate with at least two and at most three other nodes, whereas leaf nodes communicate with a single node. In the net all frontier nodes are connected with three nodes except the internal nodes which are connected with four nodes. Ring topology provides a smooth scenario for the dissipation of genetic material, since each island connects with two islands only.

These topologies provide an environment conducive to observe how miscegenation will impact the run-time and accuracy of results. For each static topology, two PIMs with 12 and 24 island were proposed (according to experimental constraints), where subscripts 12 and 24 represent the number of islands and subscript suffixes S and A differentiate synchronous and asynchronous models, respectively:

- $\mathcal{P}_{\text{C12S}}$, $\mathcal{P}_{\text{C12A}}$, $\mathcal{P}_{\text{C24S}}$ and $\mathcal{P}_{\text{C24A}}$ for the complete graph topology.
- $\mathcal{P}_{\text{R12S}}$, $\mathcal{P}_{\text{R12A}}$, $\mathcal{P}_{\text{R24S}}$ and $\mathcal{P}_{\text{R24A}}$ for the ring topology.
- $\mathcal{P}_{\text{Tr12S}}$, $\mathcal{P}_{\text{Tr12A}}$, $\mathcal{P}_{\text{Tr24S}}$ and $\mathcal{P}_{\text{Tr24A}}$ for the tree topology.
- $\mathcal{P}_{\text{To12S}}$, $\mathcal{P}_{\text{To12A}}$, $\mathcal{P}_{\text{To24A}}$ and $\mathcal{P}_{\text{To24A}}$ for the torus topology.
- $\mathcal{P}_{\text{N12S}}$, $\mathcal{P}_{\text{N12A}}$ with 4×3-net topology and $\mathcal{P}_{\text{N24S}}$, $\mathcal{P}_{\text{N24A}}$ using 6×4-net topology.

Algorithm 8 is executed, after generation of the initial population with Algorithm 7, by each island and calls algorithms 9 and 11 (that calls algorithm 10) for the proposed static and dynamic topologies, respectively.

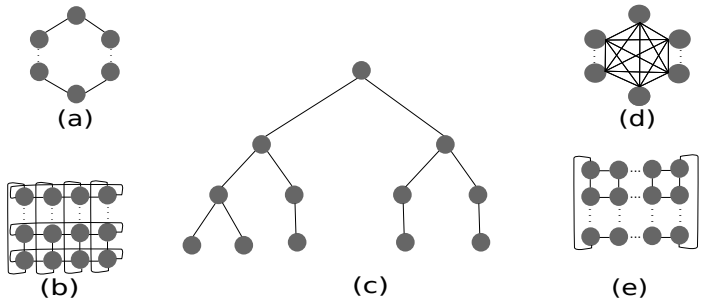


Fig. 3: **(a)** ring, **(b)** torus, **(c)** example of a binary tree with 12 nodes, **(d)** complete graph, and **(e)** $x \times y$-net, where x and y represent rows and columns in the net.

3.2 PIMs with Dynamic Topology

The quality and diversity of populations on each island were measured to apply dynamism. The islands are qualified as **good**, **bad** and **medium**. The status good, bad, and medium are related to the diversity in the islands using two metrics: variance and average. The variance measures the diversity into each island, such that a high variance is associating little resemblance between native individuals on the island. The average is associated with the quality of the population on each island. Since the problems addressed are minimization, islands with low average fitness have the potential to be considered good islands. Then, for dynamic topologies, the algorithm adds these metrics for each island and ranks islands in decreasing order before migration starts.

Algorithm 10 creates communications between islands in each migratory process according to the discussion in previous paragraph. The central idea is to create keys using the average and variance metrics (line 8) from the population on each island. The *Mergesort* algorithm is used to sort the *IdIsland* arrangement according to the MediaAverage arrangement (line 10). It was adopted that the first, middle and final third of the sorted islands in *IdIsland* are qualified as *good*, *medium* and *bad*, respectively. For the migration process, described in Algorithm 11, the creation of the communication between islands is delegated to island one (line 2).

The synchronous and asynchronous dynamic 12- and 24-island PIMs are:

- $\mathcal{P}_{\text{Rd12S}}$, $\mathcal{P}_{\text{Rd12A}}$, $\mathcal{P}_{\text{Rd24S}}$ and $\mathcal{P}_{\text{Rd24A}}$: random island communication (Rd).
- $\mathcal{P}_{\simeq\text{12S}}$, $\mathcal{P}_{\simeq\text{12A}}$,$\mathcal{P}_{\simeq\text{24S}}$ and $\mathcal{P}_{\simeq\text{24A}}$: communication between islands with the same classification: good with good, medium with medium and bad with bad ($\simeq$).
- $\mathcal{P}_{\text{gbmm12S}}$, $\mathcal{P}_{\text{gbmm12A}}$, $\mathcal{P}_{\text{gbmm24S}}$ and $\mathcal{P}_{\text{gbmm24A}}$: communication between good and bad, and between medium and medium islands (gbmm).

Before each migration, the communication between islands is generated as specified for each dynamic PIM in Algorithms 10 and 11. In case the communication is $\simeq$, pairs of contiguous islands are communicated and, case it is gbmm, pair of islands, which are selected from the begin and end to the center of the sorted list of islands, are communicated.

Algorithm 8 Pseudo code for PIMs

INPUT: X, Number of islands N_I, $maxIteration$ and parameters: BC_P, Mig_P
OUTPUT: Solution to X

procedure $IslandModels(X, N_I, P_S, BC_P, Mig_P)$
  Compute fitness of Pop_{GA}
  cont $\leftarrow Mig_P.MigrationInterval * maxIteration$
  for $i = 1$ to $maxIteration$ **do**
    $M_{best} \leftarrow$ Select the $BC_P.selection * |Pop_{GA}|$ best individuals in Pop_{GA}
    $Pop_d \leftarrow Crossover(M_{best}, Pop_{GA}, BC_P.crossover)$
    $Pop_d \leftarrow Mutation(Pop_d, BC_P.mutation)$
    $Pop_{GA} \leftarrow Replace(Pop_{GA}, Pop_d, BC_P.replacement)$
    cont $\leftarrow$ cont -1
    if cont $== 0$ **then**
      if static topology is used **then** $StaticMigrationTopology(N_I, Mig_P)$
      end if ▷ Alg. 9
      if dynamic topology is used **then** $DynamicMigrationTopology(N_I, Mig_P)$
      end if ▷ Alg. 11
      cont $\leftarrow Mig_P.MigrationInterval * maxIteration$
    end if
    if $Mig_P.Synchronization == true$ **then** Sync all islands
    end if
  end for
return *best solution*
end procedure

Algorithm 9 Migration of individuals with static topologies for each island

INPUT: Number of islands N_I, and migration parameters Mig_P
OUTPUT: Emigrant and immigrant individuals allocated on the target islands

procedure $StaticMigrationTopology(N_I, Mig_P)$
  $Ind \leftarrow$Select $Mig_P.NumMigIndividuals$ of $Mig_P.TypeEmIndividual$
  if $Mig_P.EmPolicy == Remove$ **then**
    Remove Ind, the selected individuals
  end if
  if $Mig_P.Topology == torus$ **then** ReceiveIndividuals $\leftarrow torus(N_I, Ind)$
  else if $Mig_P.Topology == tree$ **then** ReceiveIndividuals $\leftarrow tree(N_I, Ind)$
  else if $Mig_P.Topology == net$ **then** ReceiveIndividuals $\leftarrow net(N_I, Ind)$
  else if $Mig_P.Topology == ring$ **then** ReceiveIndividuals $\leftarrow ring(N_I, Ind)$
  else ReceiveIndividuals $\leftarrow completeGraph(N_I, Ind)$
  end if
  if $Mig_P.TypeImIndividual == best$ **then** Replace $Mig_P.NumMigIndividuals$ *best* individuals by the ReceiveIndividuals
  else if $Mig_P.TypeImIndividual == worst$ **then** Replace $Mig_P.NumMigIndividuals$ *worst* individuals by the ReceiveIndividuals
  else Replace $Mig_P.NumMigIndividuals$ *random* individuals by the ReceiveIndividuals
  end if
end procedure

4 Experiments

The PIMs were implemented using the `MPI` library of the `C` language in the linux operating system. The experiments were executed on a computer using two pro-

Algorithm 10 Building dynamic links between islands - Done by island one

INPUT: Number of islands N_I and migration parameters Mig_P
OUTPUT: Definition of the communication between the islands

procedure *DynamicTopology*(N_I, Mig_P)
  IdIsland, MediaAverage : array[1..N_I]
  if Mig_P.*Topology* != *Random* **then**
    for $i = 1$ to N_I **do**
      Average ← *AveragePop*(i) ▷ Average of population in island i
      Variance ← *VariancePop*(i) ▷ Variance of population in island i
      IdIsland[i] ← i
      MediaAverage[i] ← Variance + Average
    end for
    MergeSort(IdIsland, MediaAverage) ▷ Indices in IdIsland change according to merge sort of MediaAverage
    if Mig_P.*Topology* == gbmm **then**
      for $i = 1$ to $N_I/2$ **do**
        Communication(IdIsland[i], IdIsland[$N_I - i + 1$])
      end for
    end if
    if Mig_P.*Topology* == $\simeq$ **then**
      for $i = 1$ to $N_I/2$ **do**
        Communication(IdIsland[$2i - 1$], IdIsland[$2i$])
      end for
    end if
  else Random communication between islands
  end if
end procedure

Algorithm 11 Migration of individuals with dynamic topology for each island

INPUT: Number of islands N_I, and migration parameters, Mig_P
OUTPUT: Emigrant and Immigrant individuals allocated on the target islands

procedure *DynamicMigrationTopology*(N_I, Mig_P)
  if Island == 1 **then**
    DynamicTopology(N_I, Mig_P) ▷ Alg. 10
  end if
  Ind ← Select Mig_P.*NumMigIndividuals* of Mig_P.*TypeEmIndividual*
  if Mig_P.*EmPolicy* == *Remove* **then**
    Remove *Ind*, the selected individuals
  end if
  if Mig_P.*Topology* == $\simeq$ **then**
    ReceiveInd ← $\simeq$(N_I, *Ind*) ▷ ReceiveIndividuals
  else if Mig_P.*Topology* == gbmm **then**
    ReceiveInd ← gbmm(N_I, *Ind*)
  else ReceiveInd ← *Random*(N_I, *Ind*)
  end if
  if Mig_P.*TypeImIndividual* == *best* **then**
    Replace Mig_P.*NumMigIndividuals best* individuals by the ReceiveInd
  else if Mig_P.*TypeImIndividual* == *worst* **then**
    Replace Mig_P.*NumMigIndividuals worst* individuals by the ReceiveInd
  else
    Replace Mig_P.*NumMigIndividuals random* individuals by the ReceiveInd
  end if
end procedure

cessors Xeon E5-2620 with hyper-threading. Each processor has six cores with a CPU clock rate of 2.4 GHz.

As discussed by Cantú-Paz in [8], Whitley et al. in [64], Duarte et al. in[17] for a fair comparison, PIMs and sequential EAs must have total populations with the same size. Thus, the experiments compare run-time and accuracy of PIMs and their sequential versions over populations that have the same total amount of individuals. The sequential $\mathcal{GA}_{\text{S}}$ uses populations of size $24n \log n$, where n is the length of the input, while PIMs have two configurations: 1) models with 24 islands, in which each island has $n \log n$ individuals, and 2) models with 12 islands, each with $2n \log n$ individuals. Thus, all experiments were run with total populations of the same size.

The number of breeding cycles (generations) in sequential $\mathcal{GA}_{\text{S}}$ and PIMs is fixed as the input size. The size for problems URD and UTD is the length of the genome, the size of the N-Queens problem is the number of queens, and the size of TMP is the number of tasks to be addressed. The experiments for TMP were performed for instances using a fixed mesh-grid platform with 6×4 processors (see Figure 1), however, it is noteworthy that the PIMs work with any other mesh-grid platform by simply passing the platform as an argument in the models.

In some of the next tables, starting from Table 3, for brevity, the names of the models introduced in Section 3 are replaced by numbers as given in Table 1.

Table 1: Numbers encoding synchronous and asyncrhonous PIMs.

1=$\mathcal{P}_{\text{N12S}}$	3=$\mathcal{P}_{\text{To12S}}$	5=$\mathcal{P}_{\text{Tr12S}}$	7=$\mathcal{P}_{\text{R12S}}$	9= $\mathcal{P}_{\text{C12S}}$	11=$\mathcal{P}_{\text{Rd12S}}$	13=$\mathcal{P}_{\simeq\text{12S}}$	15=$\mathcal{P}_{\text{gbmm12S}}$
2=$\mathcal{P}_{\text{N24S}}$	4= $\mathcal{P}_{\text{To24A}}$	6= $\mathcal{P}_{\text{Tr24S}}$	8=$\mathcal{P}_{\text{R24S}}$	10=$\mathcal{P}_{\text{C24S}}$	12=$\mathcal{P}_{\text{Rd24S}}$	14=$\mathcal{P}_{\simeq\text{24S}}$	16=$\mathcal{P}_{\text{gbmm24S}}$
17=$\mathcal{P}_{\text{N12A}}$	19=$\mathcal{P}_{\text{To12A}}$	21=$\mathcal{P}_{\text{Tr12A}}$	23=$\mathcal{P}_{\text{R12A}}$	25= $\mathcal{P}_{\text{C12A}}$	27=$\mathcal{P}_{\text{Rd12A}}$	29=$\mathcal{P}_{\simeq\text{12A}}$	31=$\mathcal{P}_{\text{gbmm12A}}$
18=$\mathcal{P}_{\text{N24A}}$	20= $\mathcal{P}_{\text{To24A}}$	22= $\mathcal{P}_{\text{Tr24A}}$	24=$\mathcal{P}_{\text{R24A}}$	26=$\mathcal{P}_{\text{C24A}}$	28=$\mathcal{P}_{\text{Rd24A}}$	30=$\mathcal{P}_{\simeq\text{24A}}$	32=$\mathcal{P}_{\text{gbmm24A}}$

The speed-up of each PIM is computed as the ratio of the running time required by the algorithm $\mathcal{GA}_{\text{S}}$ by the one of the PIM. Each execution is measured using the Linux command *time*. The ratio is computed as the average of ten runs for each different input considering for each problem a number of inputs of significant size as specified in Sections 4.2, 4.3, 4.4 and 4.5.

4.1 Parameter Setup

EA and GA performance are highly dependent on the values of their parameters. Carefully choosing the values of the parameters is essential to guarantee a suitable parameter tunning. Here we will be using parameter tuning. By parameter tuning, we mean that the commonly practised approach is to find good parameter values before running the GA and after that, executing the GA using these values that will remain fixed during the breeding cycle. We adopt the so-called "taxonomy T1" discussed in [21]. In essence, the tuning algorithm works by the "generate and test principle", that is, through the generation of parameter vectors and testing them to establish their usefulness. Here, we use the tuner known in the literature as "iterative tuner" that starts with a small number of parameter vectors and creates new vectors iteratively during execution. In the tuner algorithm described in the sequence, we are starting with just a randomly generated parameter vector.

Table 2: Estimated Values for the Parameters

	Parameter	Estimated values
Breeding Cycle - BC_P	*crossover*	$0.02, 0.04, \cdots, 0.98, 1.0$
	mutation	$0.01, 0.011, \cdots, 0.019, 0.02$
	selection	$2\%, 4\%, \cdots, 98\%, 100\%$
	replacement	$2\%, 4\%, \cdots, 98\%, 100\%$
Migration - Mig_P	*NumMigIndividuals*	1,2,3,4,5,6,7,8,9,10,11,12,13
	TypeEmIndividual	1=Best, 2=Worst, 3=Random
	EmPolicy	1=Clone, 2=Remove
	TypeImIndividual	1=Worst, 2=Random, 3=Similar
	MigrationInterval	$2\%, 4\%, \cdots, 98\%, 100\%$

So, to improve accuracy, the $\mathcal{GA}_{\mathrm{S}}$ and PIMs were submitted to a parameter tuning taking into account each case-study. Table 2 presents the ranges of possible parameter values. For the parameters involving percentage, the tested values range between 2% and 100% checking in this manner a great variety of possibilities. The crossover rate was set from 0.02 to 1.0, while the mutation probability ranges between 0.01 and 0.02 since in evolutionary processes it is expected mutation appears rarely. For migration parameters *TypeEmIndividual*, *EmPolicy*, *TypeImIndividual* we just label the individuals as explained in Section 2.5. Regarding *NumMigIndividuals*, the tested values are naturals from 1 to 13 due to experience collected from other works, where the number of individuals was never greater than 13, as seen in [50,51,13,53].

Algorithm 12 presents a method used to calibrate the breeding cycle parameters, whereas the migration parameters are calibrated as shown in Algorithm 13. Initially, the breeding and migration parameters are initialized with random values taken from Table 2. Then, each parameter belonging to the breeding cycle is adjusted. For each v_i value of a P_k parameter belonging to the breeding cycle shown in Table 2, the PIM to be adjusted is executed 10 times for each instance of each case-study. The *vi* value that provides the best accuracy is fixed to evaluate the next parameter, P_{k+1}, later. Subsequently, the migration parameters are calibrated as shown in Algorithm 13. The calibration method is similar to that used for breeding cycle parameters, with the difference that the breeding cycle parameters fixed in Algorithm 12 are used to calibrate the migration parameters. Thus it is only necessary to start the migration parameters at the beginning of the calibration.

The size of inputs used in the experiments setup for each case-study are described below.

- URD: 100 randomly generated permutations with 150 genes.
- UTD: 100 randomly generated genomes with 150 genes and five chromosomes.
- TMP: 10 task mapping problems with 50 task sets, as defined in Section 2.3, and using a platform 6×4 processors. The task sets were generated using uniform distributions to create the τ_i tuple' elements including messages and their destination tasks.
- N-Queens: six randomly generated inputs, each of n queens for $n \in \{100, 110, \ldots, 150\}$.

Tables 3, 5, 7 and 9 provide the best values obtained for each parameter in synchronous PIMs and Tables 4, 6, 8 and 10, for asynchronous PIMs. Algorithm

12 calibrates the parameters of the $\mathcal{GA}_{\text{S}}$ ignoring the migration parameters. The breeding cycle parameters calibrated for each case-study can be found in Table 11.

Algorithm 12 Calibration of the breeding cycle parameters

INPUT: problem instance X, PIM, $\mathcal{GA}_{\text{S}}$ and parameters: BC_P, Mig_p
OUTPUT: BC_P calibrated

procedure *CalibrationBCP*(PIM, BC_P, Mig_P, X)
  $P_1 \leftarrow(2\%, 4\%, \cdots, 98\%, 100\%)$, $P_2 \leftarrow(1\%, 1.1\%, \cdots, 1.9\%, 2\%)$
  out_1 : array[1..$|P_1|$], out_2 : array[1..$|P_2|$]
  Initialize BC_P with random values
  Initialize MIG_P with random values
  for $i = 1$ to **4 do**
    if $i \leq 3$ **then**
      for $j = 1$ to $|P_1|$ **do**
        Run PIM with parameter $P_1[j]$ 10 times to solve the instances of problem ***X*** and store the average of the 10 executions in $out_1[j]$
      end for
      $index_{best} \leftarrow$ receives the index of the lowest value in out_1
      if $i == 1$ **then**
        $BC_P.crossover \leftarrow P_1[index_{best}]$
      else if $i == 2$ **then**
        $BC_P.selection \leftarrow P_1[index_{best}]$
      else
        $BC_P.replacement \leftarrow P_1[index_{best}]$
      end if
    else
      for $j = 1$ to $|P_2|$ **do**
        Run PIM with parameter $P_2[j]$ 10 times to solve the instances of problem ***X*** and store the average of the 10 executions in $out_2[j]$
      end for
      $index_{best} \leftarrow$ receives the index of the lowest value in out_2
      $BC_P.mutation \leftarrow P_2[index_{best}]$
    end if
  end for
end procedure

Table 3: Parameter setup in synchronous PIMs to solve URD.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Parameter	Static PIMs										Dynamic PIMs					
crossover	.92	.92	.90	.88	.96	.94	.96	.72	.90	.90	.98	.96	.96	.92	.98	.96
mutation	.014	.01	.01	.01	.01	.01	.01	.012	.01	.013	.01	.01	.01	.01	.01	.01
selection	80%	92%	92%	96%	98%	94%	90%	86%	90%	86%	96%	94%	84%	86%	78%	94%
replacement	60%	38%	38%	38%	50%	50%	60%	46%	50%	60%	38%	50%	40%	40%	50%	46%
NumMigIndividuals	2	1	3	1	3	2	3	5	1	4	6	4	7	10	2	5
TypeEmIndividual	1	2	2	2	1	3	2	2	3	3	1	3	2	3	3	3
EmPolicy	2	1	2	2	2	1	1	2	1	1	2	2	1	1	1	2
TypeImIndividual	1	1	2	1	2	2	1	1	2	1	1	1	2	1	2	3
MigrationInterval	10%	20%	94%	64%	56%	10%	50%	32%	10%	30%	88%	84%	20%	60%	30%	68%

Algorithm 13 Calibration of the migration parameters

INPUT: problem instance X, PIM and migration parameters Mig_P
OUTPUT: Mig_P calibrated

procedure *CalibrationMP*(PIM, Mig_P, X)
  $P_1 \leftarrow(1,2)$, $P_2 \leftarrow(1,2,3)$, $P_3 \leftarrow(1,2,\cdots,12,13)$
  $P_4 \leftarrow(2\%,4\%,\cdots,98\%,100\%)$
  out_1 : array[1..$|P_1|$], out_2 : array[1..$|P_2|$], out_3 : array[1..$|P_3|$], out_4 : array[1..$|P_4|$]
  Initialize MIG_P with random values
  for $i = 1$ to **5 do**
    if $i == 1$ **then**
      for $j = 1$ to $|P_1|$ **do**
        Run PIM with parameter $P_1[j]$ 10 times to solve the instances of problem ***X*** and store the average of the 10 executions in $out_1[j]$
      end for
      $index_{best} \leftarrow$ receives the index of the lowest value in out_1
      $BC_P.EmPolicy \leftarrow P_1[index_{best}]$
    else if $i == 2$ or $i == 4$ **then**
      for $j = 1$ to $|P_2|$ **do**
        Run PIM with parameter $P_2[j]$ 10 times to solve the instances of problem ***X*** and store the average of the 10 executions in $out_2[j]$
      end for
      $index_{best} \leftarrow$ receives the index of the lowest value in out_2
      if $i == 2$ **then**
        $BC_P.TypeEmIndividuals \leftarrow P_2[index_{best}]$
      else
        $BC_P.TypeImIndividual \leftarrow P_2[index_{best}]$
      end if
    else if $i == 3$ **then**
      for $j = 1$ to $|P_3|$ **do**
        Run P_{IM} with parameter $P_3[j]$ 10 times to solve the instances of problem ***X*** and store the average of the 10 executions in $out_3[j]$
      end for
      $index_{best} \leftarrow$ receives the index of the lowest value in out_3
      $BC_P.NumMigIndividuals \leftarrow P_3[index_{best}]$
    else
      for $j = 1$ to $|P_4|$ **do**
        Run P_{IM} with parameter $P_4[j]$ 10 times to solve the instances of problem ***X*** and store the average of the 10 executions in $out_4[j]$
      end for
      $index_{best} \leftarrow$ receives the index of the lowest value in out_4
      $BC_P.MigrationInterval \leftarrow P_4[index_{best}]$
    end if
  end for
end procedure

4.2 Experiments for URD

Packages of one hundred unsigned permutations were randomly generated containing n genes of lengths $n \in \{100, 110, \ldots, 150\}$. Parameters were taken from Table 3. For each permutation, the PIMs and the sequential $\mathcal{GA}_{\mathrm{S}}$ were executed ten times starting from the same population for all permutations of the same length. Subsequently, the averages of the ten results for each permutation and algorithm were calculated. These averages represent the number of reversals for each unsigned permutation. The average (A) results and standard deviation (SD) for each package of permutations of each length (L) are shown in Tables 12 and 13 for synchronous PIMs and in Tables 14 and 15 for asynchronous PIMs. Considering

Table 4: Parameter setup in asynchronous PIMs to solve URD.

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Parameter	Static PIMs										Dynamic PIMs					
crossover	.98	.82	.94	.94	.98	.94	.98	.98	.86	.92	.94	.94	.94	.94	.96	.94
mutation	.01	.012	.011	.011	.015	.013	.012	.01	.017	.016	.02	.017	.013	.015	.011	.016
selection	80%	84%	86%	86%	92%	86%	94%	82%	84%	80%	88%	86%	98%	86%	94%	86%
replacement	28%	70%	70%	60%	70%	70%	70%	50%	30%	70%	36%	30%	38%	70%	70%	70%
NumMigIndividuals	5	5	5	7	9	5	1	10	8	5	5	5	5	5	5	9
TypeEmIndividual	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EmPolicy	2	2	2	2	2	2	2	2	1	1	2	2	1	2	2	1
TypeImIndividual	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MigrationInterval	30%	30%	30%	30%	30%	30%	30%	10%	30%	30%	30%	30%	30%	30%	30%	30%

Table 5: Parameter setup in synchronous PIMs to solve UTD.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Parameter	Static PIMs										Dynamic PIMs					
crossover	.88	.98	.96	.84	.92	.84	.96	.98	.94	.96	.94	.96	.94	.98	.96	.96
mutation	.014	.01	.01	.014	.01	.01	.01	.01	.01	.01	.012	.011	.01	.01	.01	.014
selection	96%	68%	82%	96%	80%	96%	94%	94%	96%	94%	40%	54%	98%	96%	48%	94%
replacement	60%	70%	68%	30%	44%	30%	60%	80%	30%	60%	70%	44%	70%	28%	66%	50%
NumMigIndividuals	3	5	8	7	8	6	9	2	5	4	3	5	9	7	6	5
TypeEmIndividual	1	1	1	1	1	1	3	2	1	1	1	1	3	1	3	1
EmPolicy	2	1	2	1	2	1	1	2	2	1	2	2	2	1	1	1
TypeImIndividual	2	1	1	2	1	2	1	2	1	2	1	1	3	3	2	3
MigrationInterval	32%	30%	42%	56%	90%	56%	10%	60%	30%	10%	10%	10%	10%	40%	40%	10%

Table 6: Parameter setup in asynchronous PIMs to solve UTD.

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Parameter	Static PIMs										Dynamic PIMs					
crossover	.64	.62	.92	.62	.66	.82	.68	.84	.86	.72	.72	.72	.74	.72	.72	.78
mutation	.01	.017	.013	.011	.011	.015	.016	.012	.01	.012	.011	.013	.016	.018	.011	.016
selection	50%	40%	40%	40%	66%	40%	64%	40%	70%	50%	72%	40%	64%	40%	68%	40%
replacement	70%	60%	70%	90%	70%	70%	70%	70%	64%	80%	90%	70%	70%	70%	28%	70%
NumMigIndividuals	5	1	5	1	5	4	1	4	5	1	1	2	5	13	5	1
TypeEmIndividual	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EmPolicy	2	2	2	2	2	2	1	2	1	1	2	1	1	2	2	1
TypeImIndividual	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MigrationInterval	30%	30%	30%	30%	30%	30%	30%	10%	10%	30%	30%	30%	30%	20%	30%	30%

Table 7: Parameter setup in synchronous PIMs to solve TMP

.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Parameter	Static PIMs										Dynamic PIMs					
crossover	.80	.94	.98	.92	.96	.86	.88	.96	.82	0.90	.98	.78	.80	.94	.80	.98
mutation	.014	.01	.011	.012	.012	.01	.01	.01	.01	.013	.011	.01	.01	.01	.01	.01
selection	40%	30%	90%	30%	76%	40%	78%	20%	40%	40%	62%	30%	40%	30%	40%	68%
replacement	70%	70%	36%	70%	70%	90%	70%	80%	16%	70%	78%	18%	66%	70%	68%	72%
NumMigIndividuals	5	5	5	6	2	5	1	2	5	1	5	5	5	5	5	5
TypeEmIndividual	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	3
EmPolicy	2	2	2	2	2	2	2	2	1	1	2	1	2	2	1	2
TypeImIndividual	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1
MigrationInterval	30%	30%	74%	30%	66%	30%	26%	60%	32%	10%	28%	34%	30%	30%	28%	12%

the accuracy of the results, the best synchronous PIMs with static and dynamic topologies are respectively $\mathcal{P}_{\text{Tr12S}}$ and $\mathcal{P}_{\text{gbmm12S}}$. Looking at asynchronous PIMs,

Table 8: Parameter setup in asynchronous PIMs to solve TMP.

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Parameter	Static PIMs										Dynamic PIMs					
crossover	.80	.84	.92	.94	.96	.92	.92	.98	90%	.92	.92	.86	.80	.88	.80	.84
mutation	.011	.013	.012	.01	.014	.014	.017	.01	.018	.02	.02	.012	.014	.019	.011	.013
selection	66%	62%	64%	68%	42%	72%	86%	66%	40%	80%	64%	70%	42%	54%	40%	52%
replacement	38%	70%	36%	42%	36%	40%	70%	40%	38%	28%	70%	70%	70%	60%	70%	68%
NumMigIndividuals	5	5	5	5	7	5	3	4	4	5	1	5	5	3	5	4
TypeEmIndividual	3	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1
EmPolicy	2	2	1	1	1	1	2	1	1	1	2	2	2	1	1	1
TypeImIndividual	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MigrationInterval	32%	30%	34%	28%	32%	30%	34%	26%	10%	10%	74%	28%	30%	10%	26%	12%

Table 9: Parameter setup in synchronous PIMs to solve N-Queens.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Parameter	Static PIMs										Dynamic PIMs					
crossover	.98	.98	.98	.98	.98	.98	.92	.96	.98	.98	.98	.98	.98	.98	.98	.98
mutation	.014	.01	.011	.012	.012	.01	.01	.01	.01	.013	.011	.01	.01	.01	.01	.01
selection	50%	54%	66%	74%	94%	60%	62%	80%	54%	70%	56%	40%	56%	54%	54%	54%
replacement	28%	50%	46%	60%	54%	54%	22%	74%	50%	70%	36%	26%	56%	44%	54%	56%
NumMigIndividuals	2	11	6	5	13	7	2	13	12	4	1	4	3	7	13	4
TypeEmIndividual	3	3	3	1	2	1	1	1	1	1	1	3	2	1	1	1
EmPolicy	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1
TypeImIndividual	1	3	2	1	3	2	1	1	1	1	2	1	2	2	3	1
MigrationInterval	76%	12%	50%	50%	50%	10%	26%	46%	30%	10%	34%	54%	40%	46%	50%	20%

Table 10: Parameter setup in asynchronous PIMs to solve N-Queens.

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Parameter	Static PIMs										Dynamic PIMs					
crossover	.98	.98	.98	.96	.98	.98	.96	.96	.98	.98	.98	.98	.98	.98	.98	.98
mutation	.01	.011	.015	.013	.014	.011	.013	.012	.016	.02	.01	.016	.017	.019	.014	.013
selection	68%	66%	54%	74%	94%	60%	68%	54%	78%	56%	52%	60%	54%	74%	54%	78%
replacement	54%	50%	46%	60%	70%	54%	60%	54%	70%	68%	36%	70%	70%	64%	54%	34%
NumMigIndividuals	2	11	4	12	9	7	2	13	12	4	1	4	4	5	13	1
TypeEmIndividual	3	3	3	1	1	1	1	1	1	1	3	3	3	1	1	2
EmPolicy	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1
TypeImIndividual	1	3	2	1	3	2	1	1	1	1	2	1	2	2	3	1
MigrationInterval	54%	50%	50%	50%	20%	54%	26%	10%	32%	10%	10%	54%	56%	44%	60%	30%

Table 11: Parameter setup for the sequential GAs.

Parameter	URD	UTD	TMP	N-Queens
crossover	.90	.90	.90	.88
mutation	.02	.02	.016	.012
selection	60%	80%	92%	96%
replacement	60%	70%	38%	38%

we have $\mathcal{P}_{\text{R24A}}$ as the best static and $\mathcal{P}_{\text{gbmm12A}}$ the best dynamic. The best absolute results are highlighted in blue and are given by the synchronous models.

Table 16 presents the speed-ups for all PIMs highlighting the best in bold. The speed-up is computed as the ratio average of the set of the one hundred inputs of length 150 used in the experiments, being each input executed ten times.

Table 12: Results for URD with synchronous static PIMs: net, torus, tree, ring and complete graph.

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{N12S}}$		$\mathcal{P}_{\mathrm{N24S}}$		$\mathcal{P}_{\mathrm{To12S}}$		$\mathcal{P}_{\mathrm{To24A}}$		$\mathcal{P}_{\mathrm{Tr12S}}$		$\mathcal{P}_{\mathrm{Tr24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	79.05	2.39	77.57	1.82	77.68	1.72	77.58	1.78	77.71	1.75	77.5	1.74	77.64	1.72
110	88.1	2.33	**86.13**	1.74	86.31	1.69	86.18	1.73	86.38	1.67	**86.13**	1.64	86.27	1.66
120	97.3	2.4	94.7	1.7	94.88	1.67	94.79	1.69	94.91	1.68	94.67	1.75	94.88	1.7
130	106.67	2.44	103.44	1.9	103.6	1.79	103.48	1.87	103.71	1.85	**103.31**	1.83	103.64	1.8
140	115.75	2.28	111.89	2.24	112.16	2.21	111.94	2.21	112.23	2.17	**111.88**	2.19	112.1	2.2
150	125.35	2.25	120.8	1.67	120.96	1.61	120.81	1.58	121.01	1.57	**120.64**	1.57	120.93	1.52

	$\mathcal{P}_{\mathrm{R12S}}$		$\mathcal{P}_{\mathrm{R24S}}$		$\mathcal{P}_{\mathrm{C12S}}$		$\mathcal{P}_{\mathrm{C24S}}$	
L	A	SD	A	SD	A	SD	A	SD
100	**77.47**	1.79	77.99	1.78	77.56	1.82	77.7	1.75
110	86.15	1.7	86.65	1.64	86.21	1.76	86.34	1.68
120	**94.66**	1.67	95.36	1.72	94.84	1.73	94.94	1.65
130	103.41	1.8	104.16	1.84	103.51	1.9	103.7	1.83
140	111.89	2.23	112.72	2.27	112.07	2.23	112.31	2.18
150	120.65	1.61	121.58	1.55	120.87	1.68	121.04	1.55

Table 13: Results for URD with synchronous dynamic PIMs.

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{Rd12S}}$		$\mathcal{P}_{\mathrm{Rd24S}}$		$\mathcal{P}_{\simeq 12\mathrm{S}}$		$\mathcal{P}_{\simeq 24\mathrm{S}}$		$\mathcal{P}_{\mathrm{gbmm12S}}$		$\mathcal{P}_{\mathrm{gbmm24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	79.05	2.39	77.9	1.78	77.7	1.78	77.95	1.75	77.71	1.75	**77.53**	1.75	77.64	1.77
110	88.1	2.33	86.57	1.69	86.36	1.65	86.67	1.7	86.32	1.67	**86.12**	1.69	86.27	1.69
120	97.3	2.4	95.18	1.71	94.98	1.7	95.32	1.68	94.93	1.67	**94.76**	1.69	94.84	1.64
130	106.67	2.44	103.97	1.88	103.69	1.83	104.02	1.83	103.7	1.81	**103.42**	1.8	103.63	1.79
140	115.75	2.41	112.49	2.15	112.25	2.1	112.52	2.15	112.22	2.12	**111.87**	2.18	112.11	2.18
150	125.35	2.25	121.31	1.59	121.06	1.5	121.43	1.56	121.02	1.61	**120.72**	1.64	120.92	1.6

Table 14: Results for URD with asynchronous static PIMs: net, torus, tree, ring and complete graph.

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{N12A}}$		$\mathcal{P}_{\mathrm{N24A}}$		$\mathcal{P}_{\mathrm{To12A}}$		$\mathcal{P}_{\mathrm{To24A}}$		$\mathcal{P}_{\mathrm{Tr12A}}$		$\mathcal{P}_{\mathrm{Tr24A}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	79.05	2.39	77.59	1.8	77.81	1.82	77.53	1.82	77.63	1.79	**77.47**	1.81	77.68	1.79
110	88.1	2.33	86.18	1.72	86.5	1.67	86.16	1.74	86.27	1.72	86.16	1.75	86.28	1.71
120	97.3	2.4	94.81	1.66	95.1	1.73	94.74	1.73	94.83	1.66	**94.7**	1.7	94.88	1.66
130	106.67	2.44	103.46	1.86	103.88	1.79	103.45	1.77	103.57	1.84	103.39	1.92	103.62	1.78
140	115.75	2.28	112.0	2.24	112.42	2.18	111.94	2.23	112.1	2.15	**111.91**	1.8	112.18	2.19
150	125.35	2.25	120.83	1.61	121.24	1.66	120.69	1.67	120.89	1.58	**120.68**	1.64	120.99	1.57

	$\mathcal{P}_{\mathrm{R12A}}$		$\mathcal{P}_{\mathrm{R24A}}$		$\mathcal{P}_{\mathrm{C12A}}$		$\mathcal{P}_{\mathrm{C24A}}$	
L	A	SD	A	SD	A	SD	A	SD
100	77.5	1.82	77.55	1.79	77.73	1.84	77.7	1.76
110	**86.15**	1.7	86.17	1.65	86.34	1.72	86.41	1.73
120	95.26	1.68	94.76	1.67	95.45	1.71	95.05	1.61
130	**103.38**	1.89	103.45	1.76	103.63	1.79	103.71	1.75
140	111.93	2.2	112.04	2.19	112.18	2.28	112.31	2.19
150	120.69	1.63	120.82	1.6	120.94	1.62	121.08	1.57

4.3 Experiments for UTD

Experiments were performed with the selected parameters in Table 5 for calculating translocation distances for all PIMs and the sequential $\mathcal{GA}_{\mathrm{S}}$. For this purpose, synthetic genomes were taken from [50] with n genes, for $n \in \{100, 110, \ldots, 150\}$, and with N chromosomes, for $N \in \{3, 4, 5\}$. Then, for all these lengths and number

Table 15: Results for URD with asynchronous dynamic PIMs.

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{Rd12A}}$		$\mathcal{P}_{\mathrm{Rd24A}}$		$\mathcal{P}_{\simeq 12\mathrm{A}}$		$\mathcal{P}_{\simeq 24\mathrm{A}}$		$\mathcal{P}_{\mathrm{gbmm12A}}$		$\mathcal{P}_{\mathrm{gbmm24A}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	79.05	2.39	77.56	1.81	77.71	1.79	**77.5**	1.8	77.63	1.78	77.54	1.76	77.67	1.75
110	88.1	2.33	86.17	1.71	86.37	1.66	86.14	1.73	86.28	1.74	**86.13**	1.65	86.26	1.67
120	97.3	2.4	94.72	1.66	94.96	1.71	94.67	1.71	94.94	1.66	**94.66**	1.65	94.93	1.67
130	106.67	2.44	103.47	1.9	103.68	1.86	103.42	1.86	103.69	1.81	**103.36**	1.76	103.66	1.84
140	115.75	2.41	111.97	2.18	112.23	2.1	**111.89**	2.16	112.19	2.17	111.91	2.2	112.19	2.18
150	125.35	2.25	120.8	1.63	121.06	1.49	120.67	1.66	121.03	1.56	**120.66**	1.6	121.02	1.58

Table 16: Speed-up considering URD problem for all PIMs (See Tab. 1).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5.79	5.86	5.41	5.92	4.86	5.73	5.35	**7.17**	5.54	6.28	4.91	5.8	5.67	6.3	5.63	5.47
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
5.68	**7.07**	5.73	6.45	5.36	6.41	5.07	6.45	6.44	6.85	5.37	6.44	5.33	6.37	5.4	6.14

of chromosomes (n, N), packages of hundred genomes were processed, and the average of the results for each algorithm was calculated. Each algorithm was executed ten times for each genome provided as input, and the average of the ten results was given as output. Tables 17 and 18 present the average translocation distance and the standard deviation for the synchronous and asynchronous PIMs. The best results are highlighted for synchronous and asynchronous PIMs, colouring in blue the best absolute results. In general, the best results using synchronous PIMs processing inputs with 3 and 4 chromosomes were obtained by static models $\mathcal{P}_{\mathrm{N12S}}$ and $\mathcal{P}_{\mathrm{R12S}}$, while in the experiments with 5 chromosomes, the dynamic model $\mathcal{P}_{\simeq 12\mathrm{S}}$ presented, on average, the best results. In experiments involving asynchronous PIMs, the $\mathcal{P}_{\mathrm{C12A}}$ static model was the best representative.

The speed-ups of the PIMs regarding the sequential $\mathcal{GA}_{\mathrm{S}}$, were measured as the ratio average of the one hundred genomes with 150 genes and 5 chromosomes. Then, each model was executed ten times for each genome, and the average running time was considered. The computed speed-ups are available in Table 19 with best speed-up highlighted in bold.

4.4 Experiments for TMP

For this experiment, 60 entries were processed varying the used synthetic generated RTA number of tasks and flows. The sixty entries are divided into tens containing 30, 40, 50, 60, 70 and 80 tasks. The parameters used are those given in Tables 7 and 8. For each PIM and $\mathcal{GA}_{\mathrm{S}}$ using input I_{n_l} belonging to the set of ten inputs of length $n \in \{30, 40, 50, 60, 70, 80\}$, the following method was applied: each algorithm executed input I_{n_l} ten times; the average result of the ten runs was taken as result for the algorithm.

Accuracy of results for synchronous static PIMs are shown in Tables 20 and 21. The best results for static and dynamic topologies are provided by $\mathcal{P}_{\mathrm{N24S}}$ and $\mathcal{P}_{\mathrm{gbmm24S}}$. According to Tables 22 and 23, $\mathcal{P}_{\mathrm{To24A}}$ was the best asynchronous static PIM and $\mathcal{P}_{\mathrm{Rd24A}}$ the best dynamic PIM. The best absolute results, highlighted in blue, were with two exceptions provided by the synchronous dynamic model $\mathcal{P}_{\mathrm{gbmm24S}}$.

Table 17: Results for experiments in UTD problem considering genomes with 3, 4 and 5 chromosomes obtained from synchronous PIMs.

3 chromosomes

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{N12S}}$		$\mathcal{P}_{\mathrm{N24S}}$		$\mathcal{P}_{\mathrm{To12S}}$		$\mathcal{P}_{\mathrm{To24S}}$		$\mathcal{P}_{\mathrm{Tr12S}}$		$\mathcal{P}_{\mathrm{Tr24S}}$		$\mathcal{P}_{\mathrm{R12S}}$		$\mathcal{P}_{\mathrm{R24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	65.44	2.33	63.24	3.08	63.32	2.63	63.26	2.17	63.31	2.13	63.3	2.56	63.31	2.74	**63.22**	2.67	63.27	2.81
110	72.94	2.21	**70.7**	2.92	70.84	2.56	70.75	2.88	70.83	2.85	70.76	2.91	70.85	2.97	70.73	2.92	70.82	2.95
120	80.06	2.67	77.82	2.61	77.97	2.61	77.83	2.66	77.94	2.62	77.89	2.64	77.95	2.62	77.81	2.59	77.88	2.6
130	86.83	2.78	85.55	2.77	85.71	2.74	85.55	2.69	85.71	2.77	85.62	2.68	85.72	2.76	**85.52**	2.67	85.66	2.72
140	94.59	2.88	92.75	2.7	92.95	2.67	92.75	2.68	92.91	2.69	92.77	2.64	92.91	2.68	**92.66**	2.66	92.87	2.68
150	102.10	2.75	100.29	2.57	100.55	2.63	100.28	2.59	100.53	2.61	100.34	2.56	100.52	2.59	**100.27**	2.58	100.46	2.69

$\mathcal{P}_{\mathrm{C12S}}$		$\mathcal{P}_{\mathrm{C24S}}$		$\mathcal{P}_{\mathrm{Rd12S}}$		$\mathcal{P}_{\mathrm{Rd24S}}$		$\mathcal{P}_{\simeq 12\mathrm{S}}$		$\mathcal{P}_{\simeq 24\mathrm{S}}$		$\mathcal{P}_{\mathrm{gbmm12S}}$		$\mathcal{P}_{\mathrm{gbmm24S}}$	
A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
63.26	2.77	63.28	2.73	63.36	2.84	63.41	2.86	63.24	2.62	63.28	2.69	63.38	2.56	63.26	2.51
70.78	2.91	70.8	2.92	70.91	2.77	71.0	2.72	70.8	2.92	70.81	2.76	70.87	2.98	70.76	2.93
77.83	2.57	77.91	2.64	78.02	2.62	78.13	2.66	**77.8**	2.6	77.92	2.71	77.97	2.64	77.84	2.59
85.55	2.67	85.66	2.73	85.78	2.8	85.91	2.79	85.59	2.73	85.63	2.53	85.78	2.77	85.57	2.7
92.76	2.67	92.85	2.64	93.02	2.71	93.2	2.75	92.72	2.65	92.91	2.66	92.99	2.68	92.8	2.67
100.35	2.64	100.48	2.61	100.63	2.63	100.85	2.68	100.34	2.59	100.52	2.97	100.57	2.62	100.4	2.6

4 chromosomes

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{N12S}}$		$\mathcal{P}_{\mathrm{N24S}}$		$\mathcal{P}_{\mathrm{To12S}}$		$\mathcal{P}_{\mathrm{To24S}}$		$\mathcal{P}_{\mathrm{Tr12S}}$		$\mathcal{P}_{\mathrm{Tr24S}}$		$\mathcal{P}_{\mathrm{R12S}}$		$\mathcal{P}_{\mathrm{R24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	61.53	2.46	**61.07**	2.76	61.1	2.73	**61.07**	2.86	61.1	2.84	61.09	2.66	61.1	2.75	61.09	2.65	61.09	2.64
110	68.68	2.73	67.58	2.76	67.65	2.74	67.54	2.88	67.61	2.93	67.6	2.57	67.63	2.76	**67.53**	2.76	67.59	2.83
120	75.46	2.81	**74.68**	2.78	74.82	2.85	74.72	2.86	74.75	2.92	74.71	2.77	74.74	2.82	74.7	2.66	74.76	2.82
130	82.45	2.89	**80.93**	2.81	81.05	2.83	**80.93**	2.81	81.02	2.83	80.99	2.81	81.03	2.85	80.97	2.74	81.02	2.83
140	89.94	2.82	**87.79**	2.91	88.13	2.89	88.0	2.89	88.11	2.91	88.02	2.67	88.1	2.98	87.96	2.97	88.08	2.99
150	97.68	2.91	**94.87**	2.79	95.07	2.83	**94.87**	2.83	95.07	2.82	94.95	2.83	95.05	2.82	94.9	2.79	95.03	2.81

$\mathcal{P}_{\mathrm{C12S}}$		$\mathcal{P}_{\mathrm{C24S}}$		$\mathcal{P}_{\mathrm{Rd12S}}$		$\mathcal{P}_{\mathrm{Rd24S}}$		$\mathcal{P}_{\simeq 12\mathrm{S}}$		$\mathcal{P}_{\simeq 24\mathrm{S}}$		$\mathcal{P}_{\mathrm{gbmm12S}}$		$\mathcal{P}_{\mathrm{gbmm24S}}$	
A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
61.11	2.35	61.1	2.34	61.13	2.38	61.16	2.35	**61.07**	2.75	61.09	2.77	61.13	2.65	**61.07**	2.74
67.58	2.57	67.62	2.52	67.69	2.76	67.74	2.78	67.56	2.79	67.59	2.83	67.67	2.52	67.58	2.51
74.73	2.86	74.71	2.81	74.85	2.82	74.94	2.83	74.73	2.78	74.77	2.89	74.89	2.26	74.71	2.21
80.96	2.85	81.0	2.81	81.12	2.8	81.2	2.87	80.94	2.71	80.99	2.88	81.11	2.86	80.95	2.81
88.01	2.81	88.06	2.85	88.22	2.84	88.36	2.88	87.96	2.79	88.09	2.87	88.17	2.89	88.01	2.91
94.95	2.9	95.01	2.93	95.15	2.9	95.28	2.93	94.93	2.84	95.06	2.83	95.12	2.97	94.97	2.93

5 chromosomes

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{N12S}}$		$\mathcal{P}_{\mathrm{N24S}}$		$\mathcal{P}_{\mathrm{To12S}}$		$\mathcal{P}_{\mathrm{To24S}}$		$\mathcal{P}_{\mathrm{Tr12S}}$		$\mathcal{P}_{\mathrm{Tr24S}}$		$\mathcal{P}_{\mathrm{R12S}}$		$\mathcal{P}_{\mathrm{R24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	58.81	2.76	57.63	2.89	57.65	2.86	57.63	2.88	57.63	2.94	57.63	2.67	57.63	2.64	**57.62**	2.74	57.63	2.75
110	65.02	2.83	64.58	2.9	64.61	2.94	**64.55**	2.91	64.61	2.87	64.58	2.89	64.61	2.84	64.59	2.89	64.56	2.84
120	72.02	2.27	71.77	2.89	71.83	2.88	71.78	2.91	71.8	2.87	71.77	2.91	71.8	2.96	71.77	2.88	71.78	2.85
130	78.79	2.69	77.79	2.94	77.91	2.93	**77.78**	2.91	77.85	2.9	77.81	2.97	77.85	2.89	77.79	2.91	77.83	2.9
140	86.13	2.39	84.62	2.76	84.7	2.69	84.65	2.74	84.69	4.23	84.66	2.93	84.69	2.72	84.63	2.82	84.67	2.72
150	92.5	2.83	90.74	2.89	90.88	2.81	90.78	2.85	90.88	2.78	90.83	2.87	90.9	2.78	90.74	2.82	90.85	2.79

$\mathcal{P}_{\mathrm{C12S}}$		$\mathcal{P}_{\mathrm{C24S}}$		$\mathcal{P}_{\mathrm{Rd12S}}$		$\mathcal{P}_{\mathrm{Rd24S}}$		$\mathcal{P}_{\simeq 12\mathrm{S}}$		$\mathcal{P}_{\simeq 24\mathrm{S}}$		$\mathcal{P}_{\mathrm{gbmm12S}}$		$\mathcal{P}_{\mathrm{gbmm24S}}$	
A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
57.63	2.85	57.64	2.83	57.67	2.69	57.66	2.74	**57.62**	2.86	**57.62**	2.82	57.66	2.87	**57.62**	2.84
64.57	2.89	64.59	2.85	64.65	2.94	64.71	2.9	64.56	2.92	64.6	2.64	64.62	2.89	64.57	2.84
71.78	2.78	71.8	2.77	71.86	2.87	71.88	2.78	**71.74**	2.91	71.82	2.87	71.83	2.77	71.75	2.86
77.84	2.96	77.86	2.93	77.89	2.99	77.95	2.94	77.79	2.94	77.84	2.77	77.91	2.91	77.81	2.88
84.65	2.88	84.68	2.82	84.77	2.93	84.83	2.86	**84.61**	2.95	84.68	2.84	84.72	2.82	**84.61**	2.79
90.78	2.84	90.84	2.82	90.96	2.87	91.1	2.88	**90.71**	2.81	90.82	2.75	90.92	2.85	90.8	2.87

The speed-up was computed using ten inputs with eighty tasks each. Each PIM and the sequential $\mathcal{GA}_{\mathrm{S}}$ perform each input ten times and the speed-up was

Table 18: Results for experiments in UTD problem considering genomes with 3, 4 and 5 chromosomes obtained from asynchronous PIMs.

3 chromosomes

	$\mathcal{GA}_S$		$\mathcal{P}_{N12A}$		$\mathcal{P}_{N24A}$		$\mathcal{P}_{To12A}$		$\mathcal{P}_{To24A}$		$\mathcal{P}_{Tr12A}$		$\mathcal{P}_{Tr24A}$		$\mathcal{P}_{R12A}$		$\mathcal{P}_{R24A}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	65.44	2.33	63.48	2.23	63.6	2.27	63.35	2.22	63.6	2.27	63.38	2.24	63.47	2.2	63.4	2.24	63.41	2.16
110	72.94	2.21	71.02	2.11	71.23	2.08	70.82	2.09	71.23	2.07	70.92	2.08	71.03	2.02	70.93	2.01	71.01	2.04
120	80.06	2.67	78.17	2.82	78.47	2.78	78.06	2.78	78.47	2.78	78.07	2.79	78.21	2.7	78.05	2.75	78.12	2.66
130	86.83	2.78	85.96	2.87	86.28	2.88	85.77	2.9	86.28	2.88	85.86	2.89	85.97	2.79	85.84	2.83	85.93	2.75
140	94.59	2.88	93.26	2.84	93.63	2.77	93.1	2.75	93.63	2.77	93.09	2.79	93.36	2.75	93.09	2.82	93.22	2.75
150	102.10	2.75	100.87	2.85	101.36	2.76	100.6	2.69	101.36	2.76	100.75	2.76	100.99	2.74	100.71	2.78	100.82	2.66

$\mathcal{P}_{C12A}$		$\mathcal{P}_{C24A}$		$\mathcal{P}_{Rd12A}$		$\mathcal{P}_{Rd24A}$		$\mathcal{P}_{\simeq 12A}$		$\mathcal{P}_{\simeq 24A}$		$\mathcal{P}_{gbmm12A}$		$\mathcal{P}_{gbmm24A}$	
A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
63.31	2.22	63.47	2.21	63.33	2.21	63.57	2.25	63.34	2.22	63.48	2.2	63.36	2.23	63.55	2.23
70.81	2.02	71.06	2.06	70.9	2.03	71.16	2.06	70.86	2.02	71.08	2.0	70.87	2.07	71.08	2.01
77.88	2.69	78.21	2.71	78.03	2.73	78.32	2.74	77.99	2.75	78.21	2.72	78.04	2.74	78.21	2.69
85.61	2.82	86.03	2.81	85.8	2.87	86.11	2.81	85.72	2.86	86.05	2.83	85.76	2.79	86.06	2.85
92.84	2.78	93.29	2.7	93.01	2.81	93.45	2.8	92.98	2.78	93.31	2.8	93.04	2.77	93.39	2.76
100.45	2.72	100.95	2.68	100.69	2.7	101.14	2.72	100.55	2.74	101.07	2.74	100.71	2.73	101.05	2.74

4 chromosomes

	$\mathcal{GA}_S$		$\mathcal{P}_{N12A}$		$\mathcal{P}_{N24A}$		$\mathcal{P}_{To12A}$		$\mathcal{P}_{To24A}$		$\mathcal{P}_{Tr12A}$		$\mathcal{P}_{Tr24A}$		$\mathcal{P}_{R12A}$		$\mathcal{P}_{R24A}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	61.53	2.76	61.18	2.78	61.24	2.69	61.13	2.78	61.24	2.79	61.11	2.86	61.23	2.67	61.15	2.63	61.17	2.78
110	68.68	2.73	67.71	2.63	67.92	2.61	67.68	2.62	67.92	2.61	67.67	2.64	67.81	2.59	67.66	2.57	67.77	2.63
120	75.46	2.81	74.93	2.85	75.17	2.73	74.86	2.71	75.17	2.83	74.82	2.89	74.99	2.78	74.82	2.79	74.94	2.73
130	82.45	2.89	81.14	2.81	81.47	2.96	81.14	2.93	81.47	2.96	81.13	2.95	81.27	2.9	81.11	2.93	81.23	2.89
140	89.94	2.82	88.29	2.84	88.62	2.85	88.22	2.71	88.62	2.85	88.18	2.79	88.44	2.82	88.21	2.78	88.36	2.76
150	97.68	2.91	95.27	2.85	95.66	2.79	95.15	2.87	95.66	2.79	95.17	2.76	95.41	2.83	95.19	2.86	95.27	2.73

$\mathcal{P}_{C12A}$		$\mathcal{P}_{C24A}$		$\mathcal{P}_{Rd12A}$		$\mathcal{P}_{Rd24A}$		$\mathcal{P}_{\simeq 12A}$		$\mathcal{P}_{\simeq 24A}$		$\mathcal{P}_{gbmm12A}$		$\mathcal{P}_{gbmm24A}$	
A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
61.11	2.76	61.2	2.75	**61.1**	2.77	61.2	2.87	61.22	2.78	61.17	2.85	61.13	2.74	61.2	2.77
67.62	2.61	67.81	2.67	67.7	2.89	67.84	2.76	67.63	2.71	67.8	2.67	67.69	2.68	67.82	2.61
74.74	2.88	74.98	2.86	74.89	2.72	75.1	2.83	74.79	2.74	75.03	2.71	74.82	2.89	75.02	2.93
81.02	2.87	81.3	2.91	81.09	2.92	81.38	2.92	81.05	2.9	81.25	2.89	81.06	2.92	81.34	2.93
88.08	2.87	88.41	2.85	88.25	2.72	88.56	2.78	88.11	2.67	88.38	2.74	88.23	2.83	88.48	2.92
95.01	2.61	95.44	2.69	95.16	2.71	95.49	2.79	95.1	2.88	95.43	2.71	95.13	2.85	95.45	2.67

5 chromosomes

	$\mathcal{GA}_S$		$\mathcal{P}_{N12A}$		$\mathcal{P}_{N24A}$		$\mathcal{P}_{To12A}$		$\mathcal{P}_{To24A}$		$\mathcal{P}_{Tr12A}$		$\mathcal{P}_{Tr24A}$		$\mathcal{P}_{R12A}$		$\mathcal{P}_{R24A}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
100	58.81	2.76	57.68	2.57	57.71	2.59	57.67	2.69	57.71	2.61	**57.65**	2.72	57.69	2.78	**57.65**	2.69	57.67	2.57
110	65.02	2.83	64.65	2.96	64.73	2.94	64.63	2.93	64.73	2.94	64.66	2.92	64.67	2.92	64.62	2.98	64.63	2.9
120	72.02	2.27	71.89	2.18	72.02	2.78	71.87	2.76	72.02	2.58	71.86	2.96	71.95	2.75	71.84	2.56	71.9	2.61
130	78.79	2.69	77.95	2.64	78.15	2.85	77.92	2.98	78.15	2.55	77.93	2.96	78.07	2.7	77.91	2.81	78.02	2.96
140	86.13	2.39	84.8	2.36	85.01	2.32	84.79	2.34	85.01	2.32	84.74	2.31	84.9	2.3	84.74	2.32	84.89	2.28
150	92.50	2.83	91.05	2.95	91.36	2.97	90.97	2.89	91.36	2.97	90.99	2.89	91.18	2.87	90.94	2.9	91.11	2.89

$\mathcal{P}_{C12A}$		$\mathcal{P}_{C24A}$		$\mathcal{P}_{Rd12A}$		$\mathcal{P}_{Rd24A}$		$\mathcal{P}_{\simeq 12A}$		$\mathcal{P}_{\simeq 24A}$		$\mathcal{P}_{gbmm12A}$		$\mathcal{P}_{gbmm24A}$	
A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
57.65	2.69	57.71	2.49	57.68	2.5	57.72	2.5	**57.65**	2.48	57.68	2.48	57.66	2.48	57.67	2.48
64.62	2.91	64.69	2.9	64.67	2.9	64.71	2.94	**64.61**	2.92	64.66	2.89	64.65	2.89	64.71	2.91
71.82	2.13	71.93	2.62	71.87	2.65	72.03	2.67	71.86	2.76	71.94	2.54	71.86	2.63	71.95	2.46
77.85	2.94	78.04	2.96	77.91	2.75	78.12	2.62	77.89	2.8	78.06	2.99	77.9	2.98	78.09	2.56
84.69	2.28	84.91	2.3	84.79	2.31	84.96	2.25	84.71	2.3	84.88	2.31	84.73	2.31	84.95	2.31
90.82	2.84	91.15	2.89	90.99	2.93	91.25	2.87	90.91	2.82	91.15	2.92	90.99	2.93	91.23	2.97

computed as the total ratio average. Table 24 shows the speed-ups achieved for synchronous and asynchronous PIMs, where the best value is highlighted.

Table 19: Speed-up considering UTD problem for all PIMs (See Tab. 1).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
10.32	13.3	10.65	11.2	11.59	11.21	9.87	10.23	9.46	10.76	**19.65**	15.83	9.43	10.06	16.06	10.26
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
21.5	21.26	21.40	22.03	18.31	22.55	18.53	22.60	14.48	21.18	15.2	22.34	17.19	**23.65**	16.41	23.43

Table 20: Results for TMP with synchronous static PIMs using topologies: net, torus, tree, ring and complete graph.

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{N12S}}$		$\mathcal{P}_{\mathrm{N24S}}$		$\mathcal{P}_{\mathrm{To12S}}$		$\mathcal{P}_{\mathrm{To24S}}$		$\mathcal{P}_{\mathrm{Tr12S}}$		$\mathcal{P}_{\mathrm{Tr24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
30	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
40	0.35	0.67	0.24	0.46	0.03	0.11	0.04	0.13	0.03	0.11	**0.0**	0.0	**0.0**	0.0
50	4.25	2.4	2.56	2.31	**1.7**	1.91	1.86	1.88	1.9	2.27	1.9	2.1	2.2	2.24
60	10.9	2.39	8.34	2.22	**6.93**	2.38	7.52	2.27	7.5	2.34	7.42	2.3	7.37	1.98
70	19.9	2.26	16.6	3.29	15.27	2.87	15.52	3.0	15.63	3.35	15.36	2.78	15.5	2.81
80	29.15	2.53	22.9	2.63	**21.2**	2.46	21.27	2.78	21.3	2.87	21.53	2.84	21.57	2.89

$\mathcal{P}_{\mathrm{R12S}}$		$\mathcal{P}_{\mathrm{R24S}}$		$\mathcal{P}_{\mathrm{C12S}}$		$\mathcal{P}_{\mathrm{C24S}}$	
A	SD	A	SD	A	SD	A	SD
0.0	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
0.04	0.13	0.1	0.22	0.18	0.4	**0.0**	0.0
3.14	2.43	1.72	2.0	2.3	2.2	1.71	1.99
7.96	2.21	7.07	2.1	8.22	2.14	7.0	2.09
16.04	2.92	**15.03**	3.4	16.26	3.17	15.2	3.31
22.34	2.56	21.40	2.60	22.47	2.54	21.5	2.45

Table 21: Results for TMP with synchronous dynamic PIMs.

	$\mathcal{GA}_{\mathrm{S}}$		$\mathcal{P}_{\mathrm{Rd12S}}$		$\mathcal{P}_{\mathrm{Rd24S}}$		$\mathcal{P}_{\simeq 12\mathrm{S}}$		$\mathcal{P}_{\simeq 24\mathrm{S}}$		$\mathcal{P}_{\mathrm{gbmm12S}}$		$\mathcal{P}_{\mathrm{gbmm24S}}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
30	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
40	0.35	0.67	0.06	0.13	0.12	0.25	0.14	0.27	**0.0**	0.0	0.12	0.27	**0.0**	0.0
50	4.25	2.4	1.82	1.97	2.08	2.25	2.42	2.19	1.63	1.99	2.3	2.35	**1.6**	1.97
60	10.9	2.39	7.44	2.14	7.92	2.2	8.28	2.32	7.07	2.08	8.02	2.28	**6.88**	1.97
70	19.9	2.26	15.4	2.9	16.3	3.23	16.24	2.93	15.43	3.17	16.36	2.94	**14.54**	2.99
80	29.15	2.93	21.2	2.47	21.23	2.65	22.83	2.35	21.33	2.74	22.46	2.37	**21.14**	2.49

4.5 Experiments for N-Queens

For the experiments, thirty-one inputs with size $n \in \{50, 55, \cdots, 195, 200\}$ were used, where n is the number of queens in an $n \times n$ board. The parameter configuration presented in Table 9 was abstracted and used as below. For each entry with $n \in \{50, 55, \cdots, 195, 200\}$ queens the following steps were performed: each PIM including the sequential $\mathcal{GA}_{\mathrm{S}}$ were executed ten times; the average of these ten runs was taken as solution for the input provided.

Tables 25, 26, 27 and 28 present the results for synchronous and asynchronous PIMs, respectively. It is worth noting that for these experiments, it was not possible to calculate the standard deviation since a single entry was used for each length. The $\mathcal{P}_{\mathrm{Tr24S}}$ was the best synchronous static PIM, while $\mathcal{P}_{\mathrm{Rd12S}}$ and $\mathcal{P}_{\simeq 12\mathrm{S}}$ were the ones that provided the best solutions among the synchronous dynamic models. In experiments involving asynchronous PIMs, $\mathcal{P}_{\mathrm{C24A}}$ and $\mathcal{P}_{\simeq 12\mathrm{A}}$ were the best static and dynamic, respectively.

Table 22: Results for TMP with asynchronous static PIMs using topologies: net, torus, tree, ring and complete graph.

	$\mathcal{GA}_S$		$\mathcal{P}_{N12A}$		$\mathcal{P}_{N24A}$		$\mathcal{P}_{To12A}$		$\mathcal{P}_{To24A}$		$\mathcal{P}_{Tr12A}$		$\mathcal{P}_{Tr24A}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
30	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
40	0.35	0.67	0.18	0.45	0.03	0.11	**0.0**	0.0	**0.0**	0.0	0.02	0.06	**0.0**	0.0
50	4.25	2.4	2.36	2.15	2.03	2.08	1.94	2.0	1.8	2.09	1.88	2.17	1.73	2.02
60	10.9	2.39	8.22	2.18	7.63	2.1	7.58	2.24	6.9	2.2	7.18	2.22	7.23	2.0
70	19.9	3.26	16.58	3.14	15.97	3.23	15.7	3.04	**14.87**	2.79	15.28	2.94	15.03	3.28
80	29.15	2.53	23.1	2.17	21.93	2.83	21.97	2.3	**21.37**	2.53	21.63	2.45	21.57	2.5

$\mathcal{P}_{R12A}$		$\mathcal{P}_{R24A}$		$\mathcal{P}_{C12A}$		$\mathcal{P}_{C24A}$	
A	SD	A	SD	A	SD	A	SD
0.0	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
0.3	0.64	0.03	0.0	0.07	0.21	0.03	0.11
1.57	1.75	1.73	2.12	1.86	1.95	**1.53**	1.8
8.87	2.13	**6.87**	2.14	7.48	2.2	7.3	2.11
15.83	3.24	14.9	3.03	15.4	3.03	15.23	3.06
24.33	2.62	21.43	2.49	21.93	2.79	21.67	2.35

Table 23: Results for TMP with asynchronous dynamic PIMs.

	$\mathcal{GA}_S$		$\mathcal{P}_{Rd12A}$		$\mathcal{P}_{Rd24A}$		$\mathcal{P}_{\simeq 12A}$		$\mathcal{P}_{\simeq 24A}$		$\mathcal{P}_{gbmm12A}$		$\mathcal{P}_{gbmm24A}$	
L	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD	A	SD
30	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	0.0	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
40	0.35	0.67	0.05	0.11	0.04	0.09	0.12	0.32	**0.03**	0.07	0.1	0.22	0.1	0.25
50	4.25	2.4	2.02	2.2	**1.64**	1.82	2.48	2.35	1.9	1.97	2.42	2.22	1.96	2.03
60	10.9	2.39	7.42	2.2	**7.26**	2.31	8.32	2.29	7.37	2.2	8.32	2.26	7.34	2.16
70	19.9	3.26	15.46	2.93	**15.32**	2.96	16.16	2.89	15.77	3.17	16.44	3.21	15.54	3.0
80	29.15	2.93	**21.68**	2.67	21.74	2.49	22.3	2.36	21.79	2.48	22.8	2.55	**21.68**	2.45

Table 24: Speed-up considering TMP for all PIMs (See Tab. 1).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
14.78	9.55	5.81	12.01	6.9	**15.35**	7.28	14.63	14.51	15.09	8.25	12.05	15.04	11.98	10.7	8.49
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
8.77	10.97	8.69	8.9	12.56	8.55	5.66	8.79	13.97	7.88	8.61	6.71	14.47	11.78	**15.26**	12.77

The speed-ups are estimated running the $\mathcal{GA}_S$ and each PIM ten times over a single 250-queens input. The speed-up is computed as the ratio average of these ten runs for each PIM. See Table 29 where the best speed-up is highlighted.

5 Discussion of the Results

Parameter setup is analyzed before discussing accuracy and speed-up.

5.1 Parameter setup

The values obtained in the parameter setup phase (Tables 3 to 10) for all PIMs and applied in the experiments to the four case-studies are analyzed. The discussion points out the parameter values of the models that provided better results and presents the percentage of PIMs for which such values are used for each case-study.

Table 25: Results for N-Queens with synchronous static PIMs.

N	$\mathcal{GA}_{\text{S}}$	$\mathcal{P}_{\text{N12S}}$	$\mathcal{P}_{\text{N24S}}$	$\mathcal{P}_{\text{To12S}}$	$\mathcal{P}_{\text{To24S}}$	$\mathcal{P}_{\text{Tr12S}}$	$\mathcal{P}_{\text{Tr24S}}$	$\mathcal{P}_{\text{R12S}}$	$\mathcal{P}_{\text{R24S}}$	$\mathcal{P}_{\text{C12S}}$	$\mathcal{P}_{\text{C24S}}$
50	18.3	12.2	11.4	11.2	11.0	11.5	**9.7**	13.5	10.5	11.1	10.7
55	20.7	12.3	11.5	12.2	11.8	11.9	**10.7**	13.9	12.1	11.3	11.7
60	22.2	12.5	12.5	12.9	12.6	13.6	**11.4**	15.1	12.8	12.7	12.3
65	24.9	14.3	13.7	14.1	13.3	14.8	**12.7**	15.7	13.4	13.6	12.8
70	25.1	15.0	14.1	15.1	14.5	16.1	**14.0**	17.6	14.9	14.1	14.1
75	28.8	16.0	15.2	15.9	14.5	15.8	**14.2**	17.7	14.6	15.5	15.3
80	27.1	16.1	16.5	15.9	15.7	17.0	**14.8**	18.8	15.5	15.7	15.8
85	31.7	17.3	17.0	16.6	**15.7**	17.5	15.8	19.8	16.9	17.2	16.9
90	29.0	19.2	16.9	17.1	17.3	18.6	**16.2**	21.2	17.0	17.5	17.6
95	29.6	19.1	18.5	**18.1**	18.4	18.6	18.7	21.6	18.5	19.6	**18.1**
100	33.3	19.3	19.1	19.0	19.4	19.7	**18.4**	22.9	18.9	18.7	19.1
105	31.6	20.8	19.6	20.3	20.6	20.5	**19.3**	23.7	19.7	20.4	19.8
110	35.2	21.7	**20.1**	20.9	20.7	20.9	20.8	24.9	20.4	20.9	21.0
115	42.5	22.0	21.4	21.3	21.9	22.6	21.3	24.8	21.9	**21.1**	21.3
120	35.0	22.5	22.6	22.8	22.3	23.1	**21.9**	25.8	22.5	22.2	22.6
125	35.2	24.4	24.2	23.5	23.6	24.4	**22.7**	27.5	23.4	24.3	23.2
130	38.9	24.6	24.3	23.8	24.2	24.9	**23.4**	26.8	24.5	24.6	24.3
135	39.6	24.9	25.2	25.2	25.3	26.4	25.1	29.2	25.1	**24.3**	25.1
140	39.5	26.8	25.7	26.3	25.9	26.7	26.0	30.0	26.6	**25.5**	25.7
145	39.2	26.7	27.6	**25.8**	26.6	26.9	**25.8**	28.5	27.5	25.9	26.8
150	41.1	27.6	28.0	**27.0**	28.3	27.1	28.7	31.8	27.6	27.1	27.9
155	45.8	29.5	28.7	29.5	28.6	29.2	29.2	33.5	28.4	**28.0**	28.7
160	46.3	30.4	29.7	29.0	29.2	30.0	29.7	34.3	30.4	**28.8**	29.4
165	44.1	31.0	31.0	**30.2**	31.2	30.7	31.9	33.9	30.5	30.3	31.0
170	44.5	31.6	31.9	30.7	30.9	32.0	32.2	34.7	30.6	**30.1**	31.5
175	44.7	31.9	31.8	31.8	**31.6**	33.3	32.1	34.3	32.9	32.8	32.5
180	50.6	33.6	33.3	33.1	**32.7**	33.5	33.5	37.2	33.4	33.3	33.4
185	47.5	34.6	34.8	**33.1**	33.9	34.4	33.9	37.3	34.4	32.9	34.3
190	50.2	36.5	36.1	**34.1**	34.3	34.3	35.4	39.6	35.3	34.5	34.8
195	55.5	37.0	35.8	36.5	35.7	35.5	36.3	40.4	36.7	**35.4**	**35.4**
200	52.4	36.8	37.1	36.7	37.7	38.6	37.3	41.0	**36.4**	36.9	38.0

Crossover: for synchronous PIMs for UTD and URD, most models use a rate above 0.8. The scenario is similar with asynchronous models in URD, however, for asynchronous models in UTD, 75% of the models are set with probabilities less than 0.8. In general, the best solutions were provided with a crossover equal to 0.96.

For TMP, synchronous PIMs use a rate at least 0.78, and 56.25% of them use a probability greater than or equal to 0.9. Similar behavior is found in asynchronous PIMs. For N-Queens, all synchronous and asynchronous models use probability greater than 0.9 and approximately 85% of the models work with parameter fixed at 0.98.

Mutation: the rate of 0.01 is the most used for synchronous PIMs: 68.75% for UTD, 81.25% for URD, 62.5% for N-Queens and 62.5% for TMP. For asynchronous PIMs, the mutation probabilities are greater than the synchronous versions. The three synchronous PIMs providing good results for UTD work with a mutation probability fixed at 0.01 ($\mathcal{P}_{\text{R12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$) and 0.014 ($\mathcal{P}_{\text{N12S}}$). The best PIMs for URD ($\mathcal{P}_{\text{Tr12S}}$), TMP ($\mathcal{P}_{\text{gbmm24S}}$) and N-Queens ($\mathcal{P}_{\text{Tr24S}}$) have a mutation probability fixed at 0.01.

Selection: synchronous and asynchronous PIMs in URD use a percentage equal to or greater than 78%. When we look at the PIMs implemented for UTD, 62.5% of synchronous models use a percentage above 90% while 62.5% of asynchronous models have a configuration with a percentage equal to or less than 50%. For

Table 26: Results for N-Queens with synchronous dynamic PIMs.

N	$\mathcal{GA}_{\mathrm{S}}$	$\mathcal{P}_{\mathrm{Rd12S}}$	$\mathcal{P}_{\mathrm{Rd24S}}$	$\mathcal{P}_{\simeq 12\mathrm{S}}$	$\mathcal{P}_{\simeq 24\mathrm{S}}$	$\mathcal{P}_{\mathrm{gbmm12S}}$	$\mathcal{P}_{\mathrm{gbmm24S}}$
50	18.3	11.1	11.5	11.0	**10.3**	12.3	11.0
55	20.7	**11.2**	11.9	12.1	11.9	13.2	11.9
60	22.2	12.8	13.1	13.4	**12.5**	12.6	12.7
65	24.9	**13.1**	14.1	13.5	13.2	16.7	13.7
70	25.1	14.5	13.9	14.2	**13.6**	15.1	13.9
75	28.8	**14.1**	15.2	15.1	14.6	17.9	15.3
80	27.1	15.5	16.0	**15.4**	15.6	16.1	16.4
85	31.7	16.6	16.6	17.1	**16.3**	19.9	16.7
90	29.0	17.6	17.5	**16.9**	17.6	17.2	17.6
95	29.6	**17.7**	18.1	18.0	18.1	20.8	18.6
100	33.3	19.9	19.5	19.3	19.3	19.3	**18.7**
105	31.6	**19.4**	20.3	19.5	20.2	23.2	19.9
110	35.2	21.0	21.3	20.6	21.4	**20.4**	21.5
115	42.5	**21.4**	22.4	22.0	21.7	23.8	22.5
120	35.0	22.7	22.5	22.3	**22.2**	23.0	22.8
125	35.2	23.8	23.1	**22.9**	23.0	24.2	23.7
130	38.9	24.3	**24.1**	24.6	24.9	**24.1**	25.1
135	39.6	**24.3**	25.0	24.9	25.0	26.9	24.5
140	39.5	**25.2**	26.3	26.1	25.9	25.9	26.0
145	39.2	26.8	27.4	**26.5**	27.1	30.1	27.1
150	41.1	27.2	27.5	**26.7**	27.0	26.9	27.4
155	45.8	**28.3**	28.9	28.6	28.9	30.4	29.2
160	46.3	**28.4**	29.8	29.5	29.7	29.7	28.9
165	44.1	**30.0**	30.7	30.6	30.4	33.3	30.9
170	44.5	**30.2**	32.5	30.5	32.1	30.7	31.5
175	44.7	31.9	32.2	**31.3**	32.1	34.8	31.8
180	50.6	32.4	34.2	33.1	33.5	33.5	**32.1**
185	47.5	**32.9**	35.2	33.4	34.8	35.3	34.1
190	50.2	34.1	35.3	35.0	35.3	**33.6**	36.0
195	55.5	36.4	36.1	35.7	**35.6**	39.2	35.9
200	52.4	36.8	37.1	**35.5**	37.1	35.8	37.5

TMP, 62% of synchronous PIMs use a percentage less than 50%, and only 25% of asynchronous PIMs use rates below 50%. For N-Queens, the best synchronous and asynchronous PIMs use values between 50% and 60% and only two PIMs have selection rates adjusted above 80%. In general, for all problems, the best solutions were provided by models using selection rates above 50%.

Replacement: The best results for the four problems were achieved by PIMs using replacement rates between 36% and 72%. For URD, the synchronous PIMs used replacement rates between 38% and 60%, while asynchronous PIMs work at a higher rate than synchronous versions, which 62.5% of all asynchronous PIMs are fixed with a rate greater than or equal to 60%. For UTD, 37.5% of the synchronous PIMs use a rate less than 50%, while the remainder PIMs do not reach rates above 80%. Regarding asynchronous PIMs, only one PIM uses a rate below 60% (28%), and most PIMs work with a rate between 60% and 70%, in addition, there is a small group (18.75%) of models configured with a rate between 80% and 90%. For TMP, 81.25%, and 50% of the synchronous and asynchronous PIMs, respectively, use a rate above 60%. Emphasizing that, half of the other asynchronous PIMs are fixed with rates below 50%. For N-Queens, in general, no PIM exceeds 74%, and rates less than or equal to 50% were used by 50% and 25% of the synchronous and asynchronous PIMs, respectively.

NumMigIndividuals: the most used configurations for the four case-studies have at most 5 migrating individuals. The best adapted PIMs for TMP have 5 migrants,

Table 27: Results for N-Queens with asynchronous static PIMs.

N	$\mathcal{GA}_{\text{S}}$	$\mathcal{P}_{\text{N12A}}$	$\mathcal{P}_{\text{N24A}}$	$\mathcal{P}_{\text{To12A}}$	$\mathcal{P}_{\text{To24A}}$	$\mathcal{P}_{\text{Tr12A}}$	$\mathcal{P}_{\text{Tr24A}}$	$\mathcal{P}_{\text{R12A}}$	$\mathcal{P}_{\text{R24A}}$	$\mathcal{P}_{\text{C12A}}$	$\mathcal{P}_{\text{C24A}}$
50	18.3	11.7	**10.6**	11.0	11.0	12.5	10.8	11.3	11.2	12.0	**10.6**
55	20.7	11.7	11.7	12.2	12.2	13.0	**11.5**	11.6	11.2	12.3	11.7
60	22.2	12.6	12.6	13.7	12.5	12.7	12.4	12.8	12.9	12.5	**11.9**
65	24.9	13.4	12.9	13.7	13.3	14.5	13.7	14.3	13.1	13.8	**12.8**
70	25.1	**14.0**	**14.0**	14.5	14.4	16.1	**14.0**	14.8	14.2	14.7	14.1
75	28.8	15.4	**14.3**	15.1	15.4	16.2	15.2	15.4	15.3	15.5	15.6
80	27.1	16.1	**15.5**	**15.5**	16.1	16.7	**15.5**	16.3	16.1	17.0	**15.5**
85	31.7	17.5	16.8	17.1	**16.6**	18.0	16.9	17.0	17.0	17.2	16.8
90	29.0	17.5	17.8	17.3	17.5	18.7	17.1	17.9	17.2	17.9	**16.9**
95	29.6	19.0	18.4	17.9	18.7	19.3	18.2	18.9	18.2	19.4	**17.7**
100	33.3	18.7	19.1	20.2	19.7	21.0	**18.6**	19.3	18.8	**18.6**	19.2
105	31.6	20.2	20.5	19.9	20.2	21.4	19.6	21.0	20.6	20.6	**19.3**
110	35.2	**20.0**	21.3	20.2	21.2	22.4	20.6	22.0	21.0	20.7	20.6
115	42.5	**20.7**	21.2	21.0	22.4	23.2	21.8	22.4	21.1	22.3	21.2
120	35.0	22.4	23.4	22.4	23.2	23.5	22.4	23.4	23.5	**22.2**	23.0
125	35.2	**22.5**	24.3	22.9	23.3	23.6	22.9	23.8	23.5	23.8	22.9
130	38.9	24.1	24.5	24.0	24.6	25.6	**23.9**	24.7	24.7	24.0	24.8
135	39.6	25.1	25.3	24.9	24.7	26.1	**24.2**	25.3	25.6	25.4	24.5
140	39.5	25.4	26.0	25.7	26.5	27.4	26.3	**24.7**	25.9	26.6	25.9
145	39.2	26.6	26.6	27.2	27.0	28.4	26.6	26.8	27.1	27.3	**26.5**
150	41.1	27.6	27.0	27.5	28.1	27.8	27.4	**26.9**	28.0	27.2	27.2
155	45.8	27.9	28.6	29.4	28.8	28.8	28.0	28.0	29.6	27.9	**27.3**
160	46.3	**28.8**	30.1	29.0	30.0	30.3	29.8	29.8	29.8	29.2	29.7
165	44.1	30.0	30.9	31.0	**29.4**	31.0	31.0	30.6	31.2	30.7	29.6
170	44.5	**30.8**	31.3	31.3	32.4	32.0	31.6	31.6	32.5	32.0	**30.8**
175	44.7	**31.7**	**31.7**	33.2	31.8	33.5	33.0	32.7	32.0	32.8	31.8
180	50.6	33.7	**32.4**	33.6	34.6	33.1	33.3	32.8	34.5	33.4	34.0
185	47.5	34.4	33.8	33.9	33.7	34.6	34.7	**33.0**	35.8	34.8	34.1
190	50.2	34.7	35.6	34.8	36.6	35.8	35.7	35.0	34.9	36.2	**34.6**
195	55.5	35.6	35.7	35.5	37.0	36.7	36.0	36.5	36.8	36.3	**35.3**
200	52.4	37.2	37.9	**36.3**	37.4	37.9	38.2	37.4	38.2	37.3	37.0

whereas for URD, UTD and N-Queens, there is a heterogeneous scenario. The best results for URD are provided by $\mathcal{P}_{\text{Tr12S}}$ and $\mathcal{P}_{\text{gbmm12S}}$ with the parameter set to 3 and 4 individuals; and for UTD, $\mathcal{P}_{\text{N12S}}$ migrating 3 individuals $\mathcal{P}_{\text{R12S}}$, $\mathcal{P}_{\simeq\text{12S}}$ migrating 9 individuals, while for N-Queens $\mathcal{P}_{\text{Tr24S}}$ migrate 7 individuals.

TypeEmIndividual: for asynchronous PIMs, selection of the best individuals for emigration is the used configuration both for URD and UTD, and for 87.5% and 56.25% of such PIMs for TMP and N-Queens, respectively. Emigration of the best individuals is the most used configuration of synchronous PIMs for three case-studies: 75% for UTD, 62.5% for N-Queens and 81.25% for TMP. Regarding URD, 43.75% of all synchronous PIMs select emigration of random individuals, 37.5% the worst and 18.75% the best individuals. However, the best solutions for TMP ($\mathcal{P}_{\text{gbmm24S}}$) were obtained by emigrating random individuals, whereas in UTD, the best solutions were provided by $\mathcal{P}_{\text{N12S}}$ emigrating the best; $\mathcal{P}_{\text{R12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$ emigrating random individuals.

EmPolicy: for URD and UTD, 50% of the synchronous PIMs sent clones and 50% remove natives in the local island to be sent to the target island, while in asynchronous PIMs for URD and UTD, respectively 75% and 62.5% use the strategy of remove native individuals. For TMP, 75% of synchronous PIMs remove native individuals, while 62.5% asynchronous PIMs clone individuals. For N-Queens, 93.75% of all synchronous and asynchronous PIMs use the strategy of cloning emigrating individuals. Regarding accuracy, the PIM $\mathcal{P}_{\text{Tr24S}}$ that provides the best solutions

Table 28: Results for N-Queens with asynchronous dynamic PIMs.

N	$\mathcal{GA}_{\text{S}}$	$\mathcal{P}_{\text{Rd12A}}$	$\mathcal{P}_{\text{Rd24A}}$	$\mathcal{P}_{\simeq\text{12A}}$	$\mathcal{P}_{\simeq\text{24A}}$	$\mathcal{P}_{\text{gbmm12A}}$	$\mathcal{P}_{\text{gbmm24A}}$
50	18.3	**10.5**	11.0	10.7	10.9	11.6	11.8
55	20.7	11.7	11.8	**11.6**	11.8	12.0	12.3
60	22.2	13.5	12.5	12.9	12.7	12.4	**11.8**
65	24.9	**13.5**	13.7	14.1	13.9	14.2	13.7
70	25.1	13.9	**13.6**	**13.6**	14.7	15.1	14.6
75	28.8	15.3	14.9	15.3	**14.7**	15.5	15.6
80	27.1	16.0	15.8	15.9	**15.2**	16.3	15.7
85	31.7	**16.5**	17.2	16.6	17.3	17.3	17.0
90	29.0	17.4	17.3	17.3	**17.0**	17.7	17.3
95	29.6	18.3	18.3	18.6	18.6	18.2	**18.0**
100	33.3	19.0	18.7	19.0	**18.4**	19.3	19.7
105	31.6	20.3	**19.6**	**19.6**	19.9	19.8	20.1
110	35.2	21.0	**20.4**	20.6	21.1	**20.4**	21.0
115	42.5	21.9	22.0	21.0	**20.5**	21.1	21.8
120	35.0	21.7	**21.4**	23.3	22.6	22.4	23.5
125	35.2	22.5	23.0	**22.1**	23.4	23.0	23.8
130	38.9	**23.9**	24.4	**23.9**	24.8	24.0	24.3
135	39.6	**24.5**	25.2	24.6	24.7	25.4	26.0
140	39.5	25.7	**24.8**	25.6	25.7	25.6	26.5
145	39.2	26.4	26.5	**26.1**	27.3	26.9	28.0
150	41.1	**26.6**	27.8	26.8	27.0	27.2	28.5
155	45.8	28.3	28.4	**27.7**	**27.7**	28.7	29.0
160	46.3	29.7	29.9	**28.3**	29.4	28.9	30.3
165	44.1	30.4	29.9	**28.7**	30.5	30.8	30.4
170	44.5	31.1	30.8	**30.6**	31.2	30.8	32.1
175	44.7	**31.6**	32.2	32.4	32.6	32.1	33.1
180	50.6	**32.5**	32.6	32.8	32.9	33.1	34.0
185	47.5	33.8	33.7	**32.5**	33.5	33.8	34.4
190	50.2	34.4	34.9	**33.5**	35.7	34.9	36.4
195	55.5	36.2	36.2	**35.1**	37.0	35.5	37.4
200	52.4	36.4	**35.5**	37.0	36.7	36.0	39.1

Table 29: Speed-up considering N-Queens problem for all PIMs (See Tab. 1).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
12.54	9.34	10.9	8.62	9.06	9.04	10.38	8.7	11.86	8.63	9.75	10.03	11.67	8.42	11.86	9.47

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
12.56	9.57	12.92	9.08	9.49	9.52	11.72	9.5	10.23	10.21	**13.95**	12.45	12.62	8.97	12.72	8.83

for N-Queens clones individuals. For UTD, from the four PIMs that provided the most competitive solutions only $\mathcal{P}_{\text{R12S}}$ uses the cloning strategy, and the remaining, $\mathcal{P}_{\text{N12S}}$, $\mathcal{P}_{\text{To12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$ remove emigrants. For URD and TMP, the models that provided the best solutions, respectively, $\mathcal{P}_{\text{Tr12S}}$ and $\mathcal{P}_{\text{gbmm24S}}$ also remove emigrants.

TypeImIndividual: for URD 56.25% of the synchronous PIMs keep the strategy of removing the worst individuals. Considering UTD, there is a balance with 43.75% of the synchronous PIMs removing the worst individuals, 37.5% random individuals and 18.75% similar individuals and 93.75% of the synchronous PIMs removing the worst individuals was the most popular configuration in TMP. Removing the worst individuals on target island is the strategy used in all asynchronous PIMs for URD, UTD and TMP, while for N-Queens, 50% of the asynchronous models remove the worst and 31.25% remove random natives, and the remaining replace similar individuals. For N-Queens, each synchronous PIM maintain the same strategy than its asynchronous version. The best solutions for URD ($\mathcal{P}_{\text{Tr12S}}$) and TMP ($\mathcal{P}_{\text{gbmm24S}}$) were obtained by replacing the worst natives, N-

Queens ($\mathcal{P}_{\text{Tr24S}}$) replacing the random natives, while for UTD the best solutions were obtained by synchronous PIMs: $\mathcal{P}_{\text{N12S}}$, removing random individuals; $\mathcal{P}_{\simeq\text{12S}}$, removing similar individuals; and $\mathcal{P}_{\text{R12S}}$ removing the worst natives.
MigrationInterval: most models independent of the problem use a migration interval less than or equal to 50% of the breeding cycle. Considering accuracy, PIMs that best adapted to each problem are synchronous and use an interval fixed in 56% for URD ($\mathcal{P}_{\text{Tr12S}}$), 12% for TMP ($\mathcal{P}_{\text{gbmm24S}}$) and 10% for N-Queens ($\mathcal{P}_{\text{Tr24S}}$). Lastly, the migration interval defined in the best PIMs for UTD are: 32% for $\mathcal{P}_{\text{N12S}}$ and 10% both for $\mathcal{P}_{\text{R12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$.

5.2 Accuracy

All PIMs provide better (smaller) outputs than the sequential $\mathcal{GA}_{\text{S}}$ for the four (minimization) case-studies. The quality of the results is discussed, comparing synchronous and asynchronous models.
URD: (Tables 12, 13, 14, and 15) concerning the synchronous PIMs with static topologies, the best average solutions were provided by $\mathcal{P}_{\text{Tr12S}}$ and $\mathcal{P}_{\text{R12S}}$, while the worst solutions were obtained by ring topology using 24-island ($\mathcal{P}_{\text{R24S}}$). For synchronous dynamic PIMs the worst solutions were provided by $\mathcal{P}_{\simeq\text{12S}}$, and the best by $\mathcal{P}_{\text{gbmm12S}}$. The best asynchronous static PIMs were $\mathcal{P}_{\text{Tr12A}}$ and $\mathcal{P}_{\text{R12A}}$, while $\mathcal{P}_{\text{N24A}}$ provided the worst solutions. For asynchronous dynamic PIMs, $\mathcal{P}_{\simeq\text{24A}}$ outputs the worst while $\mathcal{P}_{\text{gbmm12A}}$ the best solutions. Comparing synchronous versus asynchronous PIMs, the experiments showed a balance as described: $\mathcal{P}_{\text{N12S}}$, $\mathcal{P}_{\text{N24S}}$, $\mathcal{P}_{\text{Tr12S}}$, $\mathcal{P}_{\text{Tr24S}}$, $\mathcal{P}_{\text{R12S}}$, $\mathcal{P}_{\text{C12S}}$, $\mathcal{P}_{\text{C24S}}$ and $\mathcal{P}_{\text{gbmm24S}}$ surpassed their asynchronous versions; $\mathcal{P}_{\text{Rd24S}}$ and $\mathcal{P}_{\text{Rd24A}}$, $\mathcal{P}_{\text{gbmm12S}}$ and $\mathcal{P}_{\text{gbmm12A}}$ output similar results; and, $\mathcal{P}_{\text{To12A}}$, $\mathcal{P}_{\text{To24A}}$, $\mathcal{P}_{\text{R24A}}$, $\mathcal{P}_{\text{Rd12A}}$, $\mathcal{P}_{\simeq\text{24A}}$ and $\mathcal{P}_{\text{gbmm12A}}$ provided better solutions than their synchronous versions. In general, the models that best and worst adapted to the URD problem were the synchronous PIMs $\mathcal{P}_{\text{Tr12S}}$ and $\mathcal{P}_{\text{R24S}}$, respectively.
UTD: (Tables 17 and 18) The experiments showed that asynchronous models are not competitive regarding their synchronous versions. Synchronous and asynchronous 12-island PIMs provided better solutions than their 24-island versions, except for $\mathcal{P}_{\text{gbmm24S}}$ that gives better solutions than $\mathcal{P}_{\text{gbmm12S}}$. The best synchronous static PIMs are $\mathcal{P}_{\text{N12S}}$, $\mathcal{P}_{\text{To12S}}$, and $\mathcal{P}_{\text{R12S}}$, using 3×4-net, torus, and ring topologies, respectively. Focusing on synchronous dynamic PIMs, $\mathcal{P}_{\text{Rd24S}}$ outputs the worst solutions while $\mathcal{P}_{\simeq\text{12S}}$ the best ones, however, surpassing $\mathcal{P}_{\text{N12S}}$, $\mathcal{P}_{\text{To12S}}$, and $\mathcal{P}_{\text{R12S}}$ only for entries containing 5 chromosomes. Analyzing the results of asynchronous PIMs, a different scenario is found, the best static model, $\mathcal{P}_{\text{C12A}}$, surpasses the best dynamic PIM $\mathcal{P}_{\simeq\text{12A}}$ in all experiment scenarios.
TMP: (Tables 20, 21, 22, and 23) For synchronous PIMs, most (62.5%) 24-island models were better adapted to the problem than their 12-island versions as seen in Figure 4. Regarding synchronous and asynchronous static PIMs, the best models were $\mathcal{P}_{\text{N24S}}$ and $\mathcal{P}_{\text{To24A}}$, respectively. For synchronous PIMs, the dynamic model $\mathcal{P}_{\text{gbmm24S}}$ was the one that best adapted. The model $\mathcal{P}_{\text{Rd24A}}$ provided the best results in asynchronous dynamic PIMs but was surpassed by $\mathcal{P}_{\text{To24A}}$ that computed the best outputs for

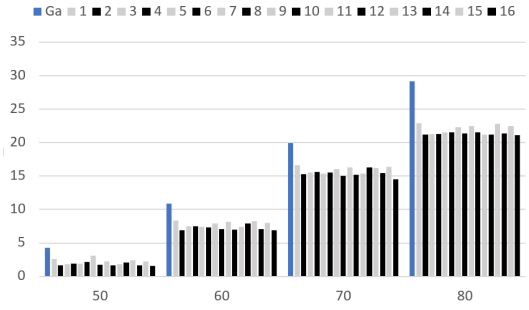


Fig. 4: Syncrhonous 12- vs 24-island PIMs.

asynchronous static PIMs. The synchronous dynamic PIMs provided on average, better quality solutions than their asynchronous versions, the exception is the $\mathcal{P}_{\text{Rd24A}}$ which has better solutions than the synchronous version. The scenario is different for static PIMs; versions from the tree topology showed similar results with 12-island and the synchronous models $\mathcal{P}_{\text{To24S}}$, $\mathcal{P}_{\text{R24S}}$ and $\mathcal{P}_{\text{C12S}}$ were overcome by their respective asynchronous versions.

N-Queens: (Tables 25, 26, 27, and 28) Figure 5 illustrates the behavior of the best synchronous PIMs; namely $\mathcal{P}_{\text{Tr24S}}$, $\mathcal{P}_{\text{Rd12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$. The experiments show that there is a slight advantage of the synchronous static model $\mathcal{P}_{\text{Tr24S}}$ regarding the dynamic models $\mathcal{P}_{\text{Rd12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$, when we look at inputs of small size but ,in general, the two synchronous dynamic PIMs deliver better solutions. Evaluating the quality of the solutions of the best asynchronous PIMs, the dynamic model $\mathcal{P}_{\simeq\text{12A}}$ is superior to the static model $\mathcal{P}_{\text{C24A}}$. Comparing static PIMs, the synchronous models $\mathcal{P}_{\text{To12S}}$, $\mathcal{P}_{\text{To24S}}$, $\mathcal{P}_{\text{Tr12S}}$, $\mathcal{P}_{\text{Tr24S}}$ and $\mathcal{P}_{\text{C12S}}$, and the asynchronous models $\mathcal{P}_{\text{N12A}}$, $\mathcal{P}_{\text{N24A}}$, $\mathcal{P}_{\text{R12A}}$, $\mathcal{P}_{\text{R24A}}$ and $\mathcal{P}_{\text{C24A}}$ provided better solutions than their respective asynchronous and synchronous versions. Concerning dynamic PIMs, the synchronous models $\mathcal{P}_{\text{Rd12S}}$ and $\mathcal{P}_{\text{gbmm24S}}$, and the asynchronous models $\mathcal{P}_{\text{Rd24A}}$, $\mathcal{P}_{\simeq\text{12A}}$, $\mathcal{P}_{\simeq\text{24A}}$ and $\mathcal{P}_{\text{gbmm12A}}$ surpassed their asynchronous and synchronous versions, respectively. An observation is the fact that the synchronous models $\mathcal{P}_{\text{Rd12S}}$ and $\mathcal{P}_{\simeq\text{24S}}$ and their asynchronous versions provided results that alternate their quality. Regarding the number of islands, all dynamic 12-island models were better than their 24-island versions except $\mathcal{P}_{\text{gbmm12S}}$ that is worse than $\mathcal{P}_{\text{gbmm24S}}$ and $\mathcal{P}_{\text{Rd12A}}$ and $\mathcal{P}_{\text{Rd24A}}$ that provide very similar results. For synchronous static PIMs, only the 24-island models $\mathcal{P}_{\text{Tr24S}}$ and $\mathcal{P}_{\text{R24S}}$ are better than their 12-island versions. Considering asynchronous PIMs, only $\mathcal{P}_{\text{Tr24A}}$ and $\mathcal{P}_{\text{C24A}}$ surpassing 12-island versions.

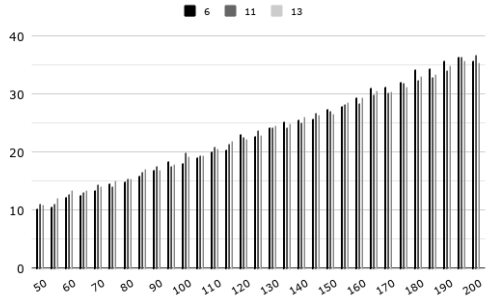


Fig. 5: Best synchronous PIMs for N-Queens.

5.3 Speed-up

Increasing the number of islands, asynchronous behaviour and sparse topologies would increase speed-up; however, when calibrating the parameters to improve accuracy, this statement may not be valid. In fact, the expected running time behaviour of PIMs is not observed in Tables 16, 19, 24, 29 since all models use different parameter configurations. Indeed, a simple experiment for all case-studies in which all PIMs use the same breeding cycle parameters of their sequential $\mathcal{GA}_{\text{S}}$ versions (Table 11) and migration parameters were set uniformly (as *EmPolicy* = 2, *TypeEmIndividual* = 1, *NumMigIndividuals* = 13, *TypeImIndividual* = 1, *MigrationInterval* = 10%) provided, as expected, better speed-ups for all asynchronous PIMs than for their synchronous versions, better speed-ups for all 24-island PIMs than for their 12-island versions. Also, the running times of PIMs with dense topologies such as torus and complete graphs are longer than the running time

of sparse PIMs such as those with tree and ring topologies. Finally, and also as expected, in the category of dynamic PIMs the better running times were obtained by the random models since they do not execute additional comparisons to classify islands. In the remaining of this section, we analyze the impact on the running time for all case-studies taking into account the weight of the parameter setup fixed (in Tables 3 to 10) to improve the accuracy of the results.

URD: (Table 16) Asynchronous and synchronous 24-island PIMs presented better speed-ups. Analyzing synchronous models, $\mathcal{P}_{\text{R24S}}$ presented the best speed-up (7.17), however the accuracy of its results is low, while $\mathcal{P}_{\text{Tr12S}}$, which provided the best solutions, achieved the worst speed-up (4.86). When analyzing the speed-ups of PIMs $\mathcal{P}_{\text{R12S}}$(5.35), and $\mathcal{P}_{\text{gbmm12S}}$(5.63), models with accuracy close to $\mathcal{P}_{\text{Tr12S}}$, there is a slight improvement in the running time, however, they did not provide the best speed-ups. The synchronous PIMs are responsible for providing the best solutions. Concerning speed-up, only 31.25% of the synchronous models present better speed-ups than their asynchronous versions, as shown in Table 16. The small running time advantage of asynchronous over synchronous PIMs can be explained by the values of parameters fixed combined with the overhead imposed in the synchronization of the evolutionary process in the synchronous versions.

UTD: (Table 19) for synchronous PIMs, the best speed-up was given by $\mathcal{P}_{\text{Rd12S}}$ (19.65) which presents competitive solutions with several other models($\mathcal{P}_{\text{N24S}}$, $\mathcal{P}_{\text{To24S}}$, $\mathcal{P}_{\text{Tr24S}}$, $\mathcal{P}_{\text{R24S}}$ $\mathcal{P}_{\text{C24S}}$, $\mathcal{P}_{\text{Rd24S}}$ and $\mathcal{P}_{\text{gbmm24S}}$), while the smallest speed-up was delivered by the $\mathcal{P}_{\simeq\text{12S}}$, one of the best solution providers for the UTD problem.

Asynchronous PIMs did not present better accuracy than synchronous versions, but in relation to speed-up they were faster being the best speed-up provided by $\mathcal{P}_{\simeq\text{24A}}$(23.65), a model that provided non-competitive accuracy results. The *crossover* and *selection* parameters in synchronous PIMs are higher than those fixed for their asynchronous versions (Tables 5 and 6) causing a slower breeding cycle, which increments the running time of synchronous PIMs.

TMP: (Table 24) The running times of many asynchronous PIMs have been overcome by their synchronous versions, which may be explained by the small values set for the breeding cycle parameters of many synchronous PIMs (Tables 7 and 8). The synchronous PIM $\mathcal{P}_{\text{Tr24S}}$ provided the best speed-up (15.35) and also competitive outputs surpassing the accuracy of most than 50% of all PIMs. The synchronous PIM $\mathcal{P}_{\text{gbmm24S}}$ that is the one that best adapted to TMP, obtained a small speed-up equal (8.49), being approximately 80% slower than $\mathcal{P}_{\text{Tr24S}}$.

Regarding the number of islands, synchronous 12-island PIMs have speed-ups that were surpassed by their 24-island versions for the topologies torus, tree, ring, complete graph and Random, and asynchronous 12-island PIMs were surpassed by their 24-island versions for the topologies net, torus and ring. In addition, the worst speed-ups were obtained by the synchronous and asynchronous 12-island models $\mathcal{P}_{\text{To12S}}$ (5.81) and $\mathcal{P}_{\text{R12A}}$ (5.66), respectively. Better running times of 12-island models than those of their 24-island versions are explained by setting of *crossover* and/or *selection* parameters with lower values than those of their 24-island versions (Tables 7 and 8).

***N*-Queens:** (Table 29) The best speed-ups were obtained by the synchronous and asynchronous 12-island; namely, $\mathcal{P}_{\text{N12S}}$ (12.54) $\mathcal{P}_{\text{Rd12A}}$ (13.95). However, only the model $\mathcal{P}_{\text{N12S}}$ presented competitive accuracies. The PIMs with the best accuracies ($\mathcal{P}_{\text{Tr24S}}$, $\mathcal{P}_{\text{Rd12S}}$ and $\mathcal{P}_{\simeq\text{12S}}$, $\mathcal{P}_{\text{C24A}}$, and $\mathcal{P}_{\simeq\text{12A}}$) are at the forefront of the worst running times, the exception is the asynchronous PIM $\mathcal{P}_{\simeq\text{12A}}$ that delivered the

fourth best speed-up. In general, 12-island PIMs presented better speed-ups than their 24-island versions. Such behavior is explained by the values of the breeding cycle parameters, particularly the *selection* parameter (Tables 9 and 10).

5.4 Statistical Analysis

For validating the results of the experiments, statistical tests (Friedman and Holm) were applied according to [15] (see also [25,16]). These statistical tests were applied in several related works such as [56,51,13,53].

The statistical tests use the best static and dynamic models both for the synchronous and asynchronous PIMs. The sample of each model is the set of outputs from the experiments, except for TMP and N-Queens, for which another set of results has been obtained. The sample size for URD and UTD problems is 100, whereas for TMP and N-Queens problems is 50. Since the case-studies are minimization problems, to apply properly the statistical tests, each output x (in a sample) was pre-processed by computing its multiplicative inverse (i.e., $1/x$).

The methodology proposed in [15] is as follows:

- The Friedman test is applied to test the null hypothesis that the performance of all algorithms is the same over a given set of inputs.
- If the null hypothesis is rejected in the previous step then the Holm test is applied to test the null hypothesis that the performance of a control algorithm is the same concerning one of the other algorithms for a given set of inputs.

The Friedman and Holm tests are available in the CONTROLTEST package at `https://sci2s.ugr.es/sicidm`. The significance level used in the tests is $\alpha = 0.05$. The Holm test uses as control PIM the one with the best (minimum) rank obtained by the Friedman test. PIMs with statistically significant difference regarding the control PIM (p-value $\leq \alpha/i$) have their p-value highlighted in Tables 30 to 35.

Table 30 shows the results of the Holm test for URD, for which $\mathcal{P}_{\text{Tr12S}}$ appears as control PIM in 3 cases out of 6. However $\mathcal{P}_{\text{Tr12S}}$ just has statistically significance difference regarding $\mathcal{GA}_{\text{S}}$.

Tables 31, 32, and 33 show the results of the Holm test for UTD considering 3, 4 and 5 chromosomes. From these tables, the PIMs that appear as control algorithms in most cases are, $\mathcal{P}_{\text{R12S}}$ for Table 31 (4 cases out of 6), $\mathcal{P}_{\text{N12S}}$ for Table 32 (4 cases out of 6), and $\mathcal{P}_{\simeq\text{12S}}$ for Table 33 (5 cases out of 6).

Table 34 shows the results of the Holm test for TMP, where $\mathcal{P}_{\text{gbmm24S}}$ appears as the control algorithm in all cases. The null hypothesis of the Friedman test for $L = 30$ was not rejected.

Table 35 shows the results of the Holm test for N-Queens. The sample consists of 50 instances of the problem varying from 50 to 246 $(50, 54, \ldots, 246)$. Here, the model $\mathcal{P}_{\simeq\text{12A}}$ appears as the control algorithm. Since the experiments were conducted with a larger amount of input due to the needs of the statistical methods used, the results presented in Table 35 confirm what had already been observed in the discussion in Section 5.2: as the number of queens is increased, the asynchronous PIM $\mathcal{P}_{\simeq\text{12A}}$ starts to stand out.

Table 30: Holm test for URD.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
100	$\mathcal{P}_{\text{Tr12A}}$ (Rank: 2.30)	4	$\mathcal{GA}_{\text{S}}$	4.89	**5.04E-31**	0.0125
		3	$\mathcal{P}_{\text{gbmm12S}}$	2.67	0.1026	0.0167
		2	$\mathcal{P}_{\text{gbmm12A}}$	2.66	0.112	0.025
		1	$\mathcal{P}_{\text{Tr12S}}$	2.47	0.46	0.05
110	$\mathcal{P}_{\text{gbmm12S}}$ (Rank: 2.435)	4	$\mathcal{GA}_{\text{S}}$	4.97	**8.62E-30**	0.0125
		3	$\mathcal{P}_{\text{Tr12A}}$	2.65	0.3252	0.0167
		2	$\mathcal{P}_{\text{Tr12S}}$	2.49	0.788	0.025
		1	$\mathcal{P}_{\text{gbmm12A}}$	2.445	0.96	0.05
120	$\mathcal{P}_{\text{Tr12S}}$ (Rank: 2.35)	4	$\mathcal{GA}_{\text{S}}$	4.97	**8.01E-32**	0.0125
		3	$\mathcal{P}_{\text{gbmm12S}}$	2.78	0.0517	0.0167
		2	$\mathcal{P}_{\text{Tr12A}}$	2.47	0.592	0.025
		1	$\mathcal{P}_{\text{gbmm12A}}$	2.42	0.75	0.05
130	$\mathcal{P}_{\text{Tr12S}}$ (Rank: 2.29)	4	$\mathcal{GA}_{\text{S}}$	5.00	**1.09E-33**	0.0125
		3	$\mathcal{P}_{\text{gbmm12S}}$	2.73	0.0491	0.0167
		2	$\mathcal{P}_{\text{Tr12A}}$	2.54	0.273	0.025
		1	$\mathcal{P}_{\text{gbmm12A}}$	2.43	0.55	0.05
140	$\mathcal{P}_{\text{gbmm12S}}$ (Rank: 2.41)	4	$\mathcal{GA}_{\text{S}}$	5.00	**5.04E-31**	0.0125
		3	$\mathcal{P}_{\text{Tr12A}}$	2.64	0.3037	0.0167
		2	$\mathcal{P}_{\text{gbmm12A}}$	2.49	0.737	0.025
		1	$\mathcal{P}_{\text{Tr12S}}$	2.46	0.81	0.05
150	$\mathcal{P}_{\text{Tr12S}}$ (Rank: 2.26)	4	$\mathcal{GA}_{\text{S}}$	5.00	**1.61E-34**	0.0125
		3	$\mathcal{P}_{\text{gbmm12S}}$	2.69	0.0517	0.0167
		2	$\mathcal{P}_{\text{Tr12A}}$	2.54	0.210	0.025
		1	$\mathcal{P}_{\text{gbmm12A}}$	2.51	0.27	0.05

6 Related work

PIMs for GAs were first studied by Grosso in [29]. From the controlled experiments, the author found that the improvement of average population fitness was faster in islands whose population is smaller than in a single large population. Grosso also observed that when the islands were isolated, the quality of the solution found after convergence was worse in the models than in sequential GA. Regarding the migration interval, the results showed that high frequency leads to premature convergence and balancing is the best choice to make the models competitive.

In [61], Tanese performed an experimental study about frequency of migrations and amount of individuals exchanged at each migration. Tanese compared the runtime of PIMs with and without communication. Both parallel and sequential GA have the same amount of individuals (256 individuals) running for 500 generations. From the experiments, it was noticed that PIMs without communication could find individuals (at some point during execution) with solutions presenting quality as a GA with a single population of individuals, but the average quality of the final population was much lower in isolated PIMs. On the other hand, for PIMs with a migratory interval, the end average quality increased significantly and in some cases was better than the final quality in a sequential GA. However, the author reiterated that this result must be interpreted with caution because the GA did not fully converge after 500 generations.

A very important theoretical question is whether (and under what conditions) PIMs can find better solutions than sequential GAs. In [58], Starkweather et al. observed that relatively isolated islands converge to different solutions, and migration is an essential part to achieve good quality solutions. The authors also

Table 31: Holm test for UTD considering genomes with 3 chromosomes.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
100	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.73)	5	$\mathcal{GA}_{\text{S}}$	6.00	**5.48E-35**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.34	0.0222	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.24	0.0539	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	2.86	0.637	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.82	0.75	0.05
110	$\mathcal{P}_{\text{N12S}}$ (Rank: 2.63)	5	$\mathcal{GA}_{\text{S}}$	6.00	**3.66E-37**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.48	**0.0012**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.26	**0.0164**	0.0167
		2	$\mathcal{P}_{\text{R12S}}$	2.87	0.354	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.74	0.66	0.05
120	$\mathcal{P}_{\simeq\text{12S}}$ (Rank: 2.65)	5	$\mathcal{GA}_{\text{S}}$	6.00	**9.63E-37**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.65	**1.57E-4**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.07	0.1124	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	2.81	0.533	0.025
		1	$\mathcal{P}_{\text{R12S}}$	2.81	0.53	0.05
130	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.62)	5	$\mathcal{GA}_{\text{S}}$	6.00	**2.88E-37**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.65	**9.90E-5**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.07	0.0890	0.0167
		2	$\mathcal{P}_{\simeq\text{12S}}$	2.92	0.265	0.025
		1	$\mathcal{P}_{\text{N12S}}$	2.72	0.71	0.05
140	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.36)	5	$\mathcal{GA}_{\text{S}}$	6.00	**4.57E-43**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.77	**8.88E-8**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.28	**5.07E-4**	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	2.85	0.061	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.73	0.16	0.05
150	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.46)	5	$\mathcal{GA}_{\text{S}}$	6.00	**1.02E-40**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.69	**3.66E-6**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.36	**6.70E-4**	0.0167
		2	$\mathcal{P}_{\simeq\text{12S}}$	2.86	0.131	0.025
		1	$\mathcal{P}_{\text{N12S}}$	2.61	0.57	0.05

reported that PIMs with moderate migration interval are propitious to find better solutions.

The success of PIMs with respect to Eldredge and Gould's theory of punctuated equilibrium is discussed by Cohoon et al. in [12]. According to this theory, evolution is characterized by long periods of isolated island evolution, punctuated by periods of geologically rapid changes associated with migration events [28]. The authors pointed out that there is a high probability of a GA reaching premature convergence, so it would be a good idea to separate isolated species into subpopulations. By adding individuals of different species to a subpopulation belonging to a respective island after convergence, new blocks of genetic material would become available; thus, through the management of the parameters involved in the migration process a PIM could, in part, avoid the problem of premature convergence.

Bianchini and Brown in [7] performed experiments and concluded that adding more islands to the PIMs is more advantageous than increasing the total population size.

In [64], Whitley et al. reviewed the benefits of using island model to improve the accuracy of results regarding sequential GAs. From a formal theoretical study, the authors statistically estimated the number of islands that a PIM needs to provide at least equivalent results when compared to a sequential GA. The quantity M' of islands is estimated as: $M' = \ln(1 - y_t)/\ln(1 - x_t/M)$, where M is the current number of islands, y_t the probability of the sequential GA to solve the problem,

Table 32: Holm test for UTD considering genomes with 4 chromosomes.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
100	$\mathcal{P}_{\text{N12S}}$ (Rank: 2.89)	5	$\mathcal{GA}_{\text{S}}$	5.92	**1.84E-30**	0.01
		4	$\mathcal{P}_{\text{C12A}}$	3.16	0.3075	0.0125
		3	$\mathcal{P}_{\simeq\text{12A}}$	3.12	0.3847	0.0167
		2	$\mathcal{P}_{\text{R12S}}$	2.99	0.705	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.91	0.92	0.05
110	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.70)	5	$\mathcal{GA}_{\text{S}}$	5.95	**1.11E-34**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.25	0.0359	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.22	0.0494	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	3.00	0.257	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.87	0.508	0.05
120	$\mathcal{P}_{\text{N12S}}$ (Rank: 3.05)	5	$\mathcal{GA}_{\text{S}}$	4.63	**2.63E-9**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.68	0.0182	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.33	0.2899	0.0167
		2	$\mathcal{P}_{\simeq\text{12S}}$	3.24	0.473	0.025
		1	$\mathcal{P}_{\text{R12S}}$	3.05	1.00	0.05
130	$\mathcal{P}_{\text{N12S}}$ (Rank: 2.76)	5	$\mathcal{GA}_{\text{S}}$	5.99	**2.81E-34**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.34	0.0298	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.20	0.0963	0.0167
		2	$\mathcal{P}_{\text{R12S}}$	2.90	0.597	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.79	0.92	0.05
140	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.68)	5	$\mathcal{GA}_{\text{S}}$	6.00	**5.12E-36**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.41	**0.0061**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.29	0.0211	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	2.85	0.533	0.025
		1	$\mathcal{P}_{\simeq\text{12S}}$	2.76	0.78	0.05
150	$\mathcal{P}_{\text{N12S}}$ (Rank: 2.62)	5	$\mathcal{GA}_{\text{S}}$	6.00	**2.26E-37**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.66	**8.47E-5**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.18	0.0327	0.0167
		2	$\mathcal{P}_{\simeq\text{12S}}$	2.80	0.484	0.025
		1	$\mathcal{P}_{\text{R12S}}$	2.73	0.68	0.05

x_t representing probability of GA solve a sub-problem belonging to the problem, and x_t/M the probability of any island to solve the problem addressed.

Selection pressure caused by the migration was evaluated by Cantú-Paz in [9]. According to the author, the response to super linear speed-up is linked to population convergence caused by additional selection pressure due to some migration policies. From the experiments, it was observed that convergence happened quickly with a migration policy, where good individuals selected on local islands replace the worst individuals on the target island; that random individuals replacing random individual has a moderate impact; and, that convergence happens slowly when both migrants and immigrants are random. The author states that the difference between the faster and slower convergence times are quite large and therefore, that the migration policy must be carefully investigated in PIMs.

In [1], Alba and Tomassini described parallelism techniques for EAs in general (GA, Evolutionary Programming, Evolutionary Strategy). The authors discussed the advantages and disadvantages of parallel EAs mentioning successful applications. Also, they discussed how decentralization of emigration and immigration can be beneficial in the search to improve the run-time as well as the use of tools to implement parallel EAs.

The effects of migration policy in PIMs are investigated by Hong et al. in [34], who proposed a model that adjusts the migration interval. Each island has its migration interval. Once a target island receives immigrants from a local island, it computes the fitness average of the target island; if the accuracy of the population

Table 33: Holm test for UTD considering genomes with 5 chromosomes.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
100	$\mathcal{P}_{\text{R12S}}$ (Rank: 2.97)	5	$\mathcal{GA}_{\text{S}}$	5.75	**7.99E-26**	0.01
		4	$\mathcal{P}_{\text{C12A}}$	3.16	0.4611	0.0125
		3	$\mathcal{P}_{\simeq\text{12A}}$	3.12	0.5580	0.0167
		2	$\mathcal{P}_{\simeq\text{12S}}$	3.00	0.910	0.025
		1	$\mathcal{P}_{\text{N12S}}$	2.99	0.94	0.05
110	$\mathcal{P}_{\simeq\text{12S}}$ (Rank: 2.86)	5	$\mathcal{GA}_{\text{S}}$	5.95	**1.63E-31**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.16	0.2568	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.12	0.3165	0.0167
		2	$\mathcal{P}_{\text{R12S}}$	2.98	0.637	0.025
		1	$\mathcal{P}_{\text{N12S}}$	2.92	0.82	0.05
120	$\mathcal{P}_{\simeq\text{12S}}$ (Rank: 2.71)	5	$\mathcal{GA}_{\text{S}}$	5.90	**1.78E-33**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.40	**0.0086**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.15	0.0963	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	2.92	0.416	0.025
		1	$\mathcal{P}_{\text{R12S}}$	2.91	0.45	0.05
130	$\mathcal{P}_{\simeq\text{12S}}$ (Rank: 2.84)	5	$\mathcal{GA}_{\text{S}}$	5.95	**6.68E-32**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.34	0.0563	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.12	0.2814	0.0167
		2	$\mathcal{P}_{\text{N12S}}$	2.88	0.880	0.025
		1	$\mathcal{P}_{\text{R12S}}$	2.86	0.94	0.05
140	$\mathcal{P}_{\simeq\text{12S}}$ (Rank: 2.79)	5	$\mathcal{GA}_{\text{S}}$	5.99	**8.93E-34**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.33	0.0394	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.16	0.1564	0.0167
		2	$\mathcal{P}_{\text{R12S}}$	2.87	0.748	0.025
		1	$\mathcal{P}_{\text{N12S}}$	2.84	0.85	0.05
150	$\mathcal{P}_{\simeq\text{12S}}$ (Rank: 2.63)	5	$\mathcal{GA}_{\text{S}}$	6.00	**3.66E-37**	0.01
		4	$\mathcal{P}_{\simeq\text{12A}}$	3.63	**1.57E-4**	0.0125
		3	$\mathcal{P}_{\text{C12A}}$	3.20	0.0312	0.0167
		2	$\mathcal{P}_{\text{R12S}}$	2.78	0.558	0.025
		1	$\mathcal{P}_{\text{N12S}}$	2.75	0.64	0.05

improves then the emigration interval of the local island increases; otherwise, it decreases. The experiments showed that the proposed PIM provides better results than models with static migration intervals.

In the same direction, a migration policy for PIMs with target island defined by attractiveness was proposed by Duarte et al. in [17]. In this policy, migrations are synchronous to occur *point to point*, where links between islands are unidirectional and weighted (with values in the interval $[0, 1]$) dynamically according to the attractiveness of the local to the target island. The weights represent the probability of each communication being used for a migration. When emigrants reach target islands, they are kept in the migrant populations for several generations, before being included in native populations. Native and migrant individuals participate in the breeding cycle. To be connected to an attractive island, such island must have higher weights than the other possible target islands. Experiments were performed for ten optimization problems observing that the proposed PIM on average provided better results than others and an excellent performance. Subsequently, Duarte et al. in [18] 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. The hybrid model consists of five islands and produced better solutions than those given by the model in [17]. Recently, Duarte et al. ([19]) optimized the hybrid model concerning how attractiveness and the weights of the connections between the islands

Table 34: Holm test for TMP.

L	Control Algorithm	i	Algorithm	Rank	P-value	α/i
40	$\mathcal{P}_{\text{gbmm24S}}$ (Rank: 2.45)	4	$\mathcal{GA}_{\text{S}}$	4.88	**1.54E-14**	0.0125
		3	$\mathcal{P}_{\text{Rd24A}}$	2.58	0.6810	0.0167
		2	$\mathcal{P}_{\text{N24S}}$	2.55	0.752	0.025
		1	$\mathcal{P}_{\text{To24A}}$	2.54	0.78	0.05
50	$\mathcal{P}_{\text{gbmm24S}}$ (Rank: 2.08)	4	$\mathcal{GA}_{\text{S}}$	5.0	**2.61E-20**	0.0125
		3	$\mathcal{P}_{\text{To24A}}$	2.74	0.0369	0.0167
		2	$\mathcal{P}_{\text{Rd24A}}$	2.74	0.037	0.025
		1	$\mathcal{P}_{\text{N24S}}$	2.44	0.25	0.05
60	$\mathcal{P}_{\text{gbmm24S}}$ (Rank: 1.76)	4	$\mathcal{GA}_{\text{S}}$	4.80	**7.02E-22**	0.0125
		3	$\mathcal{P}_{\text{Rd24A}}$	3.16	**9.55E-6**	0.0167
		2	$\mathcal{P}_{\text{N24S}}$	2.64	**0.005**	0.025
		1	$\mathcal{P}_{\text{To24A}}$	2.64	**0.01**	0.05
70	$\mathcal{P}_{\text{gbmm24S}}$ (Rank: 1.64)	4	$\mathcal{GA}_{\text{S}}$	5.0	**2.27E-26**	0.0125
		3	$\mathcal{P}_{\text{Rd24A}}$	3.11	**3.34E-6**	0.0167
		2	$\mathcal{P}_{\text{To24A}}$	2.65	**0.001**	0.025
		1	$\mathcal{P}_{\text{N24S}}$	2.60	**0.002**	0.05
80	$\mathcal{P}_{\text{gbmm24S}}$ (Rank: 1.56)	4	$\mathcal{GA}_{\text{S}}$	4.99	**2.07E-27**	0.0125
		3	$\mathcal{P}_{\text{Rd24A}}$	3.55	**3.12E-10**	0.0167
		2	$\mathcal{P}_{\text{To24A}}$	2.52	**0.002**	0.025
		1	$\mathcal{P}_{\text{N24S}}$	2.38	**0.01**	0.05

Table 35: Holm test for N-Queens.

N	Control Algorithm	i	Algorithm	Rank	P-value	α/i
50 to 246	$\mathcal{P}_{\simeq\text{12A}}$ (rank: 2.67)	5	$\mathcal{GA}_{\text{S}}$	6.00	**5.59E-19**	0.01
		4	$\mathcal{P}_{\text{C24A}}$	3.50	0.0265	0.0125
		3	$\mathcal{P}_{\simeq\text{12S}}$	3.10	0.2505	0.0167
		2	$\mathcal{P}_{\text{Tr24S}}$	2.89	0.557	0.025
		1	$\mathcal{P}_{\text{Rd12S}}$	2.84	0.65	0.05

are computed. They considered the differences for the migration strategies previously proposed in [17] and [18], focusing on the operations and information used to define the attractiveness and the methodology for adjusting the weights of the connections between islands. The adjustment operations were inspired by the natural phenomenon known as stigmergy [11], and the experiments were conducted with fifteen problems reported in [41]. Results showed that the new strategy provides competitive quality solutions.

A new PIM concept with a unidirectional ring (asynchronous) communication topology based on probability models was presented by Jaros and Schwarz in [38]. The idea is to replace migrations of individuals by the transference of a probability model that is responsible for producing the new population. The transfer takes into account the probabilistic model of emigration and the resident model that is the receiver and will be modified by the emigration model through rules of adaptive learning involving probability statistics ([5], [46]). The proposed PIM was compared with its sequential version, a PIM containing migration of individuals, and two other sequential algorithms that use probability models. Authors conclude that such PIM with probability is better than the PIM with migration of individuals, and in addition that migrating the best individuals favors the PIM with probability. On the other hand, the PIM with probability does not exceed its sequential version, and depending on the input size, provides inferior results. Concerning running time, both algorithms have reached speed-up close to eight.

Sudholt in [60] surveyed the design and analysis of parallel EAs explaining when and how it is possible to obtain accelerations on sequential EAs. Sudholt explains the challenges encountered when dealing with PIMs, as they represent interactive stochastic processes, where dynamics are determined by various design parameters, such that choices can drastically affect their performance and accuracy.

in [55], Skolicki and De Jong studied the influence of *migration interval* together with *number of immigrants and emigrants* for PIMs. One observation from the experiments is that the migration interval seems to be a dominant factor, with number of immigrants and emigrants generally playing a secondary role. Experiments also show that frequent migrations lead islands to dominate others and to lose global diversity before they can exchange genetic material. On the other hand, migrations rarely caused degraded performance due to slow convergence.

A study analyzing *synchronization policy* aspects in terms of the advantages and disadvantages of synchronous and asynchronous migration in PIMs was given by Fernández et al. in [24]. It is suggested that asynchronous migration is advantageous with respect to its synchronous counterpart in terms of accuracy. Izzo et al. [36] also advocated the asynchronous migration policy and proposed a heterogeneous PIM from variations of the differential evolution algorithm using asynchronous migration. The asynchronous migration was justified because it is more intuitive and suitable for distributed computing over TCP/IP, where resources might become available/unavailable at any time. From the experiments, the proposed PIMs obtained better performance and accuracy when compared to their sequential versions. In [2], Alba and Troya studied synchronous and asynchronous models with GA to explore the migration interval with an elitist policy where the best emigrants are sent to replace the worst individuals on the target islands. From experiments, it was observed that PIMs are superior to their sequential version regarding the quality of solutions and that asynchronous models present speed-ups superior to synchronous models, conserving the quality of the solutions.

Liu and Layland [43] analyzed a PIM for the *dynamic Maze problem*. The authors proposed a PIM with communication occurring at regular intervals. The number of islands and migration interval in the PIM were carefully studied. Experiments were performed with complete graph and ring topologies, and the best performances were obtained with the ring topology. It is worth noting that in the extreme case in which migration occurs in each generation, the proposed PIM cannot explore the search space efficiently, providing solutions with poor quality. On the other hand, when migration occurs with low frequency, the proposed PIM can avoid premature convergence having consequently a broader exploration of the search space, reaching good results.

Authors of the current work have been using island models from GA exploring the migration policy to solve complex problems in the genetic area. In [50], the authors proposed a PIM for a sequential GA introduced in [52] and applied to solve UTD problem. 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, the accuracy provided by these PIMs was not better that the one of the sequential GA. Later, in [53], authors worked out the calibration of the parameters related to migration policies and breeding cycle over PIMs considering other static topologies such as torus, complete graph, tree, and net, and a migration dynamics which explores the characteristics of individuals. As result, the outputs obtained improved both ac-

curacy and running time. After that, the proposed PIMs were applied to the URD problem in [13], improving accuracy and performance of the best-known GA.

Several recent works develop related research. In [54], four of the authors of the current paper analyzed PIMs for GA and two additional metaheuristics: self-adjusting Particle Swarm Optimization (PSO) and Social Spider Algorithm (SSA), but restricted to the synchronous PIMs and performing experiments only for the case-study of URD. The experiments were conducted with islands with small populations of n individuals, for inputs of length n with different migration policies. Results showed that in general, static PIMs for PSO output the best solutions, while PIMs for GA are competitive and PIMs for SSA present the worst solutions regarding their sequential versions. In compensation PIMs for SSA provided the best speed-ups when compared to the SSA and GA models.

In [49], Saito et al. presented a method for parallel speed-up of decomposition-based multi-objective evolutionary algorithms (MOEA/D). The method uses many-core environments such as GPUs and seeks to prevent degradation in the accuracy of solutions. To avoid degradation a virtual overlapping zone is defined between partitions, and individuals are selected for mating and migration by evaluating individuals in this zone using weight vectors of adjacent partitions. The method is compared with sequential MOEA/D, no-migration parallel MOEA/D, and a parallel MOEA/D with standard migration in which at a certain interval, the worst five solutions of each island are replaced by the five best solutions from the neighboring islands. The case-study was a two-objective knapsack problem with constraints, and experiments were conducted in various scenarios, varying the number of islands, population size and number of evolutionary cycles. The results showed that the proposed method is effective in improving diversity of solutions and running time.

Federici et al. in [23], presented the Evolutionary Optimization at Sapienza (EOS) that solves real-world (unconstrained and constrained) problems of space trajectory design and implements a synchronous heterogeneous PIM paradigm. The heterogeneity is achieved by varying the mutation method distributed in a spiral archipelago in which the islands communicate by trails (analogous to the trail on a hard disk) using a ring topology. The adopted migration policy replaces the worst natives on the target islands by the best immigrant individuals. Authors report that EOS succeeds to deal with many variables and the presence of multiple local minimums, as in the case of multiple gravity-assist trajectories, the ascent trajectory of a rocket, and multi-target rendezvous missions.

A practical study to avoid failures of refrigerated showcases placed in convenience stores and supermarkets was proposed by Otaka et al. in [47]. This paper proposes a fault detection method by pasting based artificial neural networks using parallel multi-population modified brain storm optimization and correntropy (see [45]). Variations with static ring, trigonal pyramid and cube topologies were used with an elitist migration policy where the best immigrants replace the worst natives every ten generations. The experiments were conducted with a small number of islands (2, 4 and 8). Statistical tests showed the superiority of the proposed model using the trigonal pyramid topology with four islands indicating a success rate of approximately 99.4% of the samples presented.

7 Conclusions and Future Work

To the best of our knowledge, there are no previous works addressing our methodology in terms of carefully analyzing each parameter involved in migration policies for synchronous and asynchronous PIMs. We use an exhaustive experimental approach exploring the potential of each migration policy. From this mindset, we proposed and analyzed homogeneous PIMs for four case-studies: *Unsigned Reversal Distance*, *Unsigned Translocation Distance*, *Scheduling Task Mapping Problem*, and *N-Queens*. For each case-study, synchronous and asynchronous 12- and 24-island PIMs were implemented using static topologies: net, torus, tree, ring, and complete graph, and three dynamic topologies based on the diversity and genotype of the native individuals. To obtain good quality results and solid feedback regarding the adequacy of each different PIM, parameters were properly and separately calibrated for each model.

From the experiments, it is observed that each PIM requires specific breeding cycle and migration parameters to achieve satisfactory accuracy having an impact in its run-time; in particular, the setup of migration parameters resulted in very different values. But for all PIMs the adequate parameter setup provided satisfactory speed-ups and always better accuracy than the one provided by the sequential GA, which implies that the way in which both the breeding cycle and the migration process are integrated is crucial to successfully adapt each PIM to each case-study.

The best accuracies for TMP were achieved by synchronous dynamic 24-island PIM, while the best accuracies for URD and UTD by synchronous static and dynamic 12-island PIMs, on the opposite side we have N-Queens where the best results were provided by an asynchronous dynamic 12-island model. For URD, UTD and N-Queens, the speed-up of the majority of asynchronous PIMs were better than those of their synchronous versions. The contrary happens for TMP. However, for URD and TMP the best speed-ups were provided by synchronous PIMs. For URD and UTD, the majority of 24-island PIMs provided better speed-ups than their 12-island versions. For TMP, the best speed-ups were provided by 24-island PIMs, but in several cases, 12-island PIMs presented better speed-ups than their 24-island versions, while for N-Queens the best speed-up was provided by a 12-island PIM and the majority of 12-island PIMs gave better speed-ups than their 24-island versions.

Current work explores heterogeneous PIMs, in which islands may run different algorithms. There is a variety of EAs that can apply in such heterogeneous PIMs, such as particle swarm optimization [20], artificial bee colony [39], shuffled frog leaping [22], elephant herding behaviour [63], social spider algorithm [37], among others. Such research will require the initial development of sequential versions for these EAs specialized for each different case-study and the subsequent development of heterogeneous PIMs for which migration policies, as well as speed-up and accuracy, need to be investigated. Besides, of course, comparing heterogeneous and homogeneous PIMs.

References

1. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation **6**(5), 443–462 (2002)

2. Alba, E., Troya, J.M.: Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Future Generation Computer Systems **17**(4), 451–465 (2001). Workshop on Bio-inspired Solutions to Parallel Computing problems
3. Baderand, A.D., Moret, B.M., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. Journal of Computational Biology **8**(5), 483–491 (2001)
4. Bafna, V., Pevzner, P.A.: Genome rearrangements and sorting by reversals. In: Proc. of 1993 IEEE 34th Annual Foundations of Computer Science, pp. 148–157 (1993). DOI 10.1109/SFCS.1993.366872
5. Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Tech. Rep. CMU-CS-94-163, CS Dept., Carnegie Mellon University (1994)
6. Bergeron, A., Mixtacki, J., Stoye, J.: On sorting by translocations. J. of Computational Biology **13**(2), 567–578 (2006)
7. Bianchini, R., Brown, M.C.: Parallel Genetic Algorithms on Distributed-Memory Architectures. In: NATUG-6: Proceedings of the Sixth Conference of the North American Transputer Users Group on Transputer Research and Applications 6. IOS Press, Vancouver, British Columbia, Canada (1993)
8. Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis **10**(2), 141–171 (1998)
9. Cantú-Paz, E.: Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms. J. of Heuristics **7**(4), 311–334 (2001). DOI 10.1023/A:1011375326814
10. Caprara, A.: Sorting by reversals is difficult. In: Proceedings of the First Annual International Conference on Computational Molecular Biology, pp. 75–83. Association for Computing Machinery, New York, NY, USA (1997). DOI 10.1145/267521.267531
11. Capriles, P.V.S.Z., Fonseca, L.G., Barbosa, H.J.C., Lemonge, A.C.C.: Rank-based ant colony algorithms for truss weight minimization with discrete variables. Communications in Numerical Methods in Engineering **23**(6), 553–575 (2007). DOI https://doi.org/10.1002/cnm.912
12. Cohoon, J.P., Hegde, S.U., Martin, W.N., Richards, D.: Punctuated equilibria: a parallel genetic algorithm. In: Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms, pp. 434–439. Hillsdale, NJ: L. Erlhaum Associates, 1987., Massachusetts Institute of Technology, Cambridge (1987)
13. da Silveira, L.A., Soncco-Álvarez, J.L., de Lima, T.A., Ayala-Rincón, M.: Parallel multi-island genetic algorithm for sorting unsigned genomes by reversals. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2018). DOI 10.1109/CEC.2018.8477968
14. De Micheli, G., Benini, L.: Networks on chips: 15 years later. Computer **50**(5), 10–11 (2017). DOI 10.1109/MC.2017.140
15. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)
16. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3–18 (2011)
17. Duarte, G., Lemonge, A., Goliatt, L.: A dynamic migration policy to the island model. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1135–1142 (2017). DOI 10.1109/CEC.2017.7969434
18. Duarte, G., Lemonge, A., Goliatt, L.: A New Strategy to Evaluate the Attractiveness in a Dynamic Island Model. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2018). DOI 10.1109/CEC.2018.8477706
19. Duarte, G.R.d.C.L., da Fonseca, A.C., de Lima, L.G., Pires, B.S.L.: An Island Model based on Stigmergy to solve optimization problems. Natural Computing pp. 1–29 (2020). DOI 10.1007/s11047-020-09819-x
20. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of the Sixth International Symposium on Micro Machine and Human Science (MHS), pp. 39–43 (1995). DOI 10.1109/MHS.1995.494215
21. Eiben, A., Smit, S.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation **1**(1), 19–31 (2011). DOI 10.1016/j.swevo.2011.02.001
22. Eusuff, M., Lansey, K.: Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm. Journal of Water Resources Planning and Management **129**(3), 210–225 (2003). DOI 10.1061/(ASCE)0733-9496(2003)129:3(210)

23. Federici, L., Benedikter, B., Zavoli, A.: EOS: a Parallel, Self-Adaptive, Multi-Population Evolutionary Algorithm for Constrained Global Optimization. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–10. IEEE (2020). DOI 10.1109/CEC48606.2020.9185800
24. Fernández, F., Galeano, G., Gómez, J.A.: Comparing Synchronous and Asynchronous Parallel and Distributed Genetic Programming Models. In: 5th European Conference on Genetic Programming, pp. 326–335. Springer (2002)
25. García, S., Herrera, F.: An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. Journal of Machine Learning Research **9**, 2677–2694 (2008)
26. Garey, M.R., Johnson, D.S.: Computers and Intractability: : A Guide to the Theory of NP-Completeness. W.H. Freeman New York (2002)
27. Gent, I.P., Jefferson, C., Nightingale, P.: Complexity of n-Queens Completion. Journal of Artificial Intelligence Research **59**, 815–848 (2017). DOI 10.1613/jair.5512
28. Gould, N., Eldredge, S.J.: Punctuated equilibria: An alternative to phyletic gradualism. Essential readings in evolutionary biology pp. 82–115 (1972)
29. Grosso, P.B.: Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model. Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA (1985). AAI8520908
30. Grusea, S., Labarre, A.: The distribution of cycles in breakpoint graphs of signed permutations. Discrete Applied Mathematics **161**(10), 1448–1466 (2013). DOI https://doi.org/10.1016/j.dam.2013.02.002
31. Hannenhalli, S.: Polynomial-time algorithm for computing translocation distance between genomes. Discrete Applied Mathematics **71**(1), 137–151 (1996)
32. Hannenhalli, S., Pevzner, P.: Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals. In: Proc. of the 27th Annual ACM Symposium on Theory of Computing (STOC), pp. 178–189. ACM (1995)
33. Hesham, S., Rettkowski, J., Goehringer, D., Abd El Ghany, M.A.: Survey on Real-Time Networks-on-Chip. IEEE Transactions on Parallel and Distributed Systems **28**(5), 1500–1517 (2017)
34. Hong, T.P., Lin, W.Y., Liu, S.M., Lin, J.H.: Experimental analysis of dynamic migration intervals on 0/1 knapsack problems. In: 2007 IEEE Congress on Evolutionary Computation (CEC), pp. 1163–1167 (2007). DOI 10.1109/CEC.2007.4424601
35. Indrusiak, L.S.: End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. Journal of Systems Architecture **60**(7), 553–561 (2014). DOI 10.1016/j.sysarc.2014.05.002
36. Izzo, D., Rucinski, M., Ampatzis, C.: Parallel global optimisation meta-heuristics using an asynchronous island-model. In: 2009 IEEE Congress on Evolutionary Computation (CEC), pp. 2301–2308 (2009). DOI 10.1109/CEC.2009.4983227
37. James, J., Li, V.O.: A social spider algorithm for global optimization. Applied Soft Computing **30**, 614–627 (2015)
38. Jaros, J., Schwarz, J.: Parallel BMDA with probability model migration. In: 2007 IEEE Congress on Evolutionary Computation (CEC), pp. 1059–1066 (2007). DOI 10.1109/CEC.2007.4424587
39. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report, Erciyes University p. 10 (2005)
40. Lazarova, M.: Efficiency of parallel genetic algorithm for solving n-queens problem on multicomputer platform. In: the 9th WSEAS International Conference on Evolutionary Computing, pp. 51–56 (2008)
41. Liang, J., Qu, B., Suganthan, P., Chen, Q.: Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. Technical Report, Zhengzhou University (2014)
42. de Lima, T.A., Ayala-Rincón, M.: On the average number of reversals needed to sort signed permutations. Discrete Applied Mathematics **235**(Supplement C), 59–80 (2018)
43. Lissovoi, A., Witt, C.: A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization. Algorithmica **78**(2), 641–659 (2017). DOI 10.1007/s00453-016-0262-4
44. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard-real-time environment. J. of the ACM (JACM) **20**(1), 46–61 (1973)
45. Liu, W., Pokharel, P.P., Principe, J.C.: Correntropy: A localized similarity measure. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 4919–4924 (2006)

46. Mühlenbein, H.: The Equation for Response to Selection and Its Use for Prediction. Evolutionary Computation **5**(3), 303–346 (1997). DOI 10.1162/evco.1997.5.3.303
47. Otaka, N., Y.Fukuyama, Kawamura, Y., Murakami, K., Santana, A.: Refrigerated Showcase Fault Detection by a Pasting based Artificial Neural Networks using Parallel Multipopulation Modified Brain Storm Optimization and Correntropy. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2020). DOI 10.1109/CEC48606.2020.9185511
48. Pettey, C.B., Leuze, M.R., Grefenstette, J.J.: Parallel Genetic Algorithm. In: Proc. of the second International Conference on Genetic Algorithms their applications at the Massachusetts Institute of Technology (1987)
49. Saito, Y., Sato, M., Midtlyng, M., Miyakawa, M.: Parallel and Distributed MOEA/D with Exclusively Evaluated Mating and Migration. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2020). DOI 10.1109/CEC48606.2020.9185559
50. da Silveira, L.A., Soncco-Álvarez, J.L., Ayala-Rincón, M.: Parallel memetic genetic algorithms for sorting unsigned genomes by translocations. In: 2016 IEEE Congress onEvolutionary Computation (CEC), pp. 185–192 (2016)
51. da Silveira, L.A., Soncco-Álvarez, J.L., Ayala-Rincón, M.: Parallel genetic algorithms with sharing of individuals for sorting unsigned genomes by reversals. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 741–748 (2017). DOI 10.1109/CEC.2017.7969384
52. da Silveira, L.A., Soncco-Álvarez, J.L., de Lima, T.A., Ayala-Rincón, M.: Computing translocation distance by a genetic algorithm. In: 2015 Latin American Computing Conference (CLEI), pp. 1–12 (2015). DOI 10.1109/CLEI.2015.7359994
53. da Silveira, L.A., Soncco-Álvarez, J.L., de Lima, T.A., Ayala-Rincón, M.: Parallel Island Model Genetic Algorithms applied in NP-Hard problems. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2019)
54. da Silveira, L.A., Soncco-Álvarez, J.L., de Lima, T.A., Ayala-Rincón, M.: 2020 IEEE Congress on Evolutionary Computation (CEC). In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2020). DOI 10.1109/CEC48606.2020.9185732
55. Skolicki, Z., De Jong, K.: The influence of migration sizes and intervals on island models. In: Proc. of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 1295–1302. ACM (2005). DOI 10.1145/1068009.1068219. URL `http://doi.acm.org/10.1145/1068009.1068219`
56. Soncco-Álvarez, J.L., Muñoz, D.M., Ayala-Rincón, M.: Opposition-based memetic algorithm and hybrid approach for sorting permutations by reversals. Evolutionary computation **27**(2), 229–265 (2019)
57. Starkweather, T., Whitley, D., Mathias, K.: Optimization using distributed genetic algorithms. In: International Conference on Parallel Problem Solving from Nature, pp. 176–185. Springer (1990)
58. Starkweather, T., Whitley, D., Mathias, K.: Optimization using distributed genetic algorithms", pp. 176–185. Springer Berlin Heidelberg, Berlin, Heidelberg (1991). DOI 10.1007/BFb0029750
59. Stone, H.S., Stone, J.M.: Efficient search techniques-an empirical study of the n-queens problem. IBM Journal of Research and Development **31**(4), 464–474 (1987). DOI 10.1147/rd.314.0464
60. Sudholt, D.: Springer Handbook of Computational Intelligence, chap. Parallel Evolutionary Algorithms, pp. 929–959. Springer (2015)
61. Tanese, R.: Distributed Genetic Algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms, pp. 434–439. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)
62. Tanese, R.: Distributed genetic algorithms for function optimization. Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA (1989). AAI9001722
63. Wang, G., Deb, S., Gao, X., Coelho, L.D.S.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. International Journal of Bio-Inspired Computation **8**(6), 394–409 (2016)
64. Whitley, D., Rana, S., heckendorn, R.B.: The island model genetic algorithm: On separability, population size and convergence. J. of Computing and Information Technology **7**(1), 33–47 (1999)
65. Wolf, W., Jerraya, A., Martin, G.: Multiprocessor System-on-Chip (MPSoC) Technology. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**(10), 1701–1713 (2008)

66. Xiaohui Hu, Eberhart, R.C., Yuhui Shi: Swarm intelligence for permutation optimization: a case study of n-queens problem. In: Proc. of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), pp. 243–246 (2003). DOI 10.1109/SIS.2003.1202275
67. Zhu, D., Wang, L.: On the complexity of unsigned translocation distance. Theor. Comp. Sci. **352**(1), 322–328 (2006)