Received: 29 February 2016 | Revised: 27 December 2016 | Accepted: 27 December 2016

DOI: 10.1002/cpe.4087

WILEY

RESEARCH ARTICLE

Hierarchical parallel model for improving performance on differential evolution

María Laura Tardivo[1,2,3] | **Paola Caymes-Scutari**[1,2] | **Germán Bianchini**[1] | **Miguel Méndez-Garabetti**[1,2]

[1]Laboratorio de Investigación en Cómputo Paralelo/Distribuido LICPaD. Departamento de Ingeniería en Sistemas de Información, Facultad Regional Mendoza, Universidad Tecnológica Nacional, Mendoza, M5502AJE, Argentina
[2]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Buenos Aires, Argentina
[3]Departamento de Computación, Universidad Nacional de Río Cuarto, Río Cuarto, X5804BYA, Córdoba, Argentina

Correspondence
María Laura Tardivo, Departamento de Computación, Universidad Nacional de Río Cuarto, Facultad de Ciencias Exactas, Físico-Químicas y Naturales. Ruta Nacional 36 Km 601, X5804BYA Río Cuarto, Córdoba, Argentina.
Email: lauratardivo@dc.exa.unrc.edu.ar

Funding Information
UTN (Argentina), Grant/Award Number: EIUTIME0003939TC, EIUTNME0003952; MEC (Spain), Grant/Award Number: TIN2014-53234-C2-1-R

Summary

This paper presents a parallel distributed model for the Differential Evolution algorithm. The proposed model, Hierarchical Island-Based Model for Differential Evolution, follows a double-hierarchy master-worker scheme and offers two parallelism levels. In this proposal, the processes are associated with certain cooperation hierarchy, allowing them to explore in a more comprehensive way the search space of the problem at hand. A comparative study with other algorithms from the state of art is also presented. The results show that Hierachical Island-Based Model for Differential Evolution is a flexible model and achieves good performance in terms of results quality and computing time.

KEYWORDS

comparative study, Differential Evolution, double hierarchy, high performance computing, island model

1 | INTRODUCTION

Many real scientific and industrial problems can be formulated as numerical optimization problems, but most of them cannot be solved analytically. Examples include the electronic systems design (placement and routing of components,[1] and performance improvement or the manufacture yield of circuits[2]), image processing (image segmentation,[3] and color separation[4]), management problems, and planning or predictive models (air traffic management,[5] electrical substation planning,[6] and wildfire behaviour prediction[7]), among others. Over the past few decades, researchers have focused on the development of algorithmic strategies that attempt to provide approximate solutions to these problems. Among them, evolutionary computation offers a number of advantages such as ease of implementation, reliable performance, robustness, global search capability, parallelism posibilities, etc.[8]

An optimization problem is defined in one study[8] by a couple (S, f), where S represents the set of possible solutions and $f : S \rightarrow \mathbb{R}$ is the objective function to be optimized. The objective function assigns to every solution $s \in S$ of the search space a real number indicating its quality. Hence, the main goal in solving an optimization problem is to find a solution $s^* \in S$, called *global optimum*, which has the best objective function value of all solutions of the search space. The most successful models used in practice to formulate and solve optimization problems are based on mathematical programming.[8] Within it, we find lineal models like Linear Programming or Integer Linear Programming, in which both the objective function and the constraints associated are lineal functions. Nonlinear programming models deal with mathematical programming problems where the objective function and/or the constraints are nonlinear. These models are much more difficult to solve; moreover, some problem properties such as high dimensionality, multimodality, parameter interaction, and nondifferentiability can not be addressed with those traditional approaches. Metaheuristics are good candidates to solve moderate and large instances of this type of problems. They represent a family of approximate optimization techniques and provide "acceptable" solutions in a reasonable time. Unlike

exact optimization algorithms, metaheuristics do not guarantee the optimality of the obtained solutions, but they are designed to obtain good solutions by searching in a clever way over the problem search space.

The Differential Evolution (DE) algorithm is a population-based metaheuristic proposed by Rainer Storn and Kenneth Price[9] in 1995, and it has exhibited an overall excellent performance for a wide range of benchmarks as well as real-world application problems.[10] It has been used in a wide variety of areas, including economy,[11] bioinformatics,[12–14] industrial engineering,[15,16] prediction models,[17,18] image processing,[19] and many others. At present, it is possible to find a large number of research and empirical work describing variants of this algorithm, with the aim of achieving better performance in results quality or computing time. Examples include the calibration of some interest parameters,[20,21] the development of hybrid models that combine and enhance the outstanding features of each methodology,[12,22] or the utilization of other techniques providing new strategies that improve the decision-making process.[23,24]

With the increasing rate of progress in the hardware area, parallel processing has given rise to a world of techniques that attempt to exploit computational resources to solve these and other problems effectively and efficiently. The main motivation for parallel computing is to accelerate the computation by solving the same problem in a lower time, or by solving a broader problem preserving a similar execution time. It has been found that the usage of parallel approaches in metaheuristics can further improve the solutions quality, regarding the sequential algorithm, at a low cost of hardware resources or execution time.[25] There is a particular interest in the parallelization of DE on parallel and distributed general purpose architectures, as they are one of the most popular and available computational platforms.

This paper describes an island parallel model for the DE algorithm. The new proposal, called Hierarchical Island Based Model for Differential Evolution (HIBM-DE), is based on islands and incorporates two information exchange mechanism levels. The processes involved are associated with certain cooperation hierarchy, allowing them to explore in a more comprehensive way the search space of the problem at hand and favouring the maintenance of the population diversity. This model is inspired by two parallel approaches, called Subpopulation Island Based Model (SIBM-DE) and Classic Island Based Model (CIBM-DE),[26] inheriting their main characteristics and combining their main advantages.

The remainder of the paper is structured as follows. Section 2 briefly introduces the classical DE algorithm. Section 3 provides a summarized description about some related work on parallel DE and presents 2 parallel approaches named SIBM-DE and CIBM-DE. Section 4 gives the details of the HIBM-DE method and differentiates it from other parallel approaches. Section 5 presents the obtained results as well as the performance comparisons among HIBM-DE and other parallel/distributed DE algorithms. Finally, Section 6 summarizes the main conclusions and future work.

2 | DIFFERENTIAL EVOLUTION

The DE algorithm is a population-based stochastic optimizer proposed by Price and Storn[9] in 1995. It starts to explore the search space by generating a population of individuals. An individual is defined as a D-dimensional vector, whose initial values are randomly obtained based on the limits defined by the user and according to the problem nature.

Each individual represents a possible solution of the problem and belongs to a generation g, ie, let $X_{i,g} = (x^1_{i,g}, \ldots, x^D_{i,g})$ an individual of the population, with $i = 1, \ldots, N$ where the index i denotes *i-th* population individual, g determines the generation to which the individual belongs, D is the problem dimension, and N is the total number of individuals in the population.

The main idea of the method is to use vectors difference[27] to modify the population vector. After initialization, DE performs in sequence three vector operations: mutation, crossover, and selection. These operations are repeated from generation to generation until the specified termination criterion is satisfied. This criterion could be finding a predefined minimal error, reaching a certain number of iterations, or reaching a predefined number of objective function evaluations. A notation is adopted to represent the different strategies used in the vector operators, namely, *DE/x/y/z*. *x* is a string representing the vector to be perturbed, *y* is the number of difference vectors used in the mutation phase, and *z* is the crossover type used. Following, a description of the strategy *DE/best/1/bin* is presented.

2.1 | Mutation

The mutation operation applies vectors differences among the existing population members for determining both the degree and direction of the new individuals produced. The mutation process begins in each generation by selecting random individuals $X_{r_1,g}, X_{r_2,g}$. A mutant vector $V_{i,g+1}$ is obtained using the strategy of the Equation 1, where the indexes i, r_1, and r_2 are integers numbers different from each other, randomly generated in the range $[1, N]$.

$$V_{i,g+1} = X_{best,g} + (X_{r_1,g} - X_{r_2,g})F \tag{1}$$

The constant F, commonly known as *scaling factor* or *amplification* factor, is a positive number usually less than 1.0 that controls the amplification difference between individuals r_1 and r_2, and it is also used to avoid stagnation in the search process. $X_{best,g}$ is the best individual, ie, it has the best objective function evaluation value among all individuals of the current generation g. Differential Evolution proposes an alternative mutation strategy. Instead of using the best individual (such as in Equation 1), another individual randomly selected can be used. The Equation 1 is one of the most popular mutation strategies. Additionally, DE proposes three alternative strategies.[10]

2.2 | Crossover

After mutation phase, the mutant vector $V_{i,g+1} = (v^1_{i,g+1}, \ldots, v^D_{i,g+1})$ and the current population individual $X_{i,g} = (x^1_{i,g}, \ldots, x^D_{i,g})$ are involved in the crossover operation, generating a new vector $U_{i,g+1} = (u^1_{i,g+1}, \ldots, u^D_{i,g+1})$, denominated "trial vector." There are two crossing operators that can be applied: binomial or exponential. For the binomial crossover operator, the Equation 2 is used, and the changed coordinates are dispersed randomly over the dimensions $\{1, \ldots, D\}$. This crossover type copies the jth parameter value from the mutant vector $V_{i,g+1}$ to the corresponding element in the trial vector $U_{i,g+1}$ if $rand_j \leq Cr$ or $j = k$.

Otherwise, it is copied from the corresponding target (or parent) vector $X_{i,g}$.

$$U^j_{i,g+1} = \begin{cases} v^j_{i,g+1} & \text{if } rand_j \le Cr \text{ or } j = k \\ x^j_{i,g} & \text{in other case} \end{cases} \quad (2)$$

The indexes j and k belongs to $\{1, \ldots, D\}$. The constant $Cr \in [0,1]$, denominated "crossover factor," is an algorithm parameter defined by the user. Cr is used to control the values fraction that are copied from the mutant vector V. The value $rand_j \in \{1, \ldots, N\}$ is the output of a uniformly distributed random number generator, and it is calculated in each comparison made on the vector components. The value k is a randomly generated index chosen for each individual, and it is used to ensure that the trial vector is not exactly equal to its source vector $X_{i,g}$, in which case, the vector component for the index is taken from the mutated vector.

To complete the notation, when the binomial crossover is utilized, the method is named "*DE///bin*," whilst "*DE///exp*" is denoted when the exponential crossover is used.

2.3 | Selection

This phase determines which individuals will be part of the next generation. The objective function of each trial vector $U_{i,g+1}$ is evaluated and compared with the objective function value for its counterpart $X_{i,g}$ in the current population. If the trial vector has less or equal objective function target value (for minimization problems), it will replace the vector $X_{i,g}$ in the population of the next generation (Equation 3). As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation.

$$X_{i,g+1} = \begin{cases} U_{i,g+1} & \text{if } f(U_{i,g+1}) \le f(X_{i,g}) \\ X_{i,g} & \text{in other case} \end{cases} \quad (3)$$

3 | PARALLEL MODELS

In evolutionary computing, the concept of "island" arises from the development of hybrid parallel models, which involve several self-contained algorithms performing a search in parallel and cooperating to find an optimum.[8] In general, an island is considered to be a logic entity integrated by a population or subpopulation, which is evolved by a single process or a group of processes. Periodically, some individuals may move from one island to another to integrate the target population. This communication among islands is usually called "migration phase," and its occurrence is established by the migration rate parameter (Mr), which determines how frequently the migration take place, and by the migration percentage parameter (Mp), which defines the amount of individuals that will migrate.

An algorithm involving islands and migration is said to be an algorithm that follows the *Island Model*. In general terms, when the island model is implemented in a parallel environment, it can be mapped over a *master/worker* pattern.[28] It is characterized by a coordinator process, usually called *master*, and other processes, called *workers*, that collaborate to obtain some final result, making some tasks and exchanging information with the master, and possibly with each other. Usually, each process is located in a separate computing unit.

According to the literature,[8,10] there are different configuration possibilities for the island model depending on the purpose to be achieved. For example, a master process may be in charge of the initialization and distribution of the population among the worker processes (management of a unique population subdivided into subpopulations), or the master can adopt a control position, indicating each worker to manage its own population (the use of multiple populations). Also, the master can help workers to evolve a subpopulation, to take advantage of all the computational resources. Also, the islands may be initialized with different populations or may use different parameter settings for DE, such as the size of the population in each island, the mutation probability, crossover factor, mutation strategy, and so on.

The rest of the section is organized as follows. Section 3.1 summarizes some related work on other parallel/distributed DE algorithms. Sections 3.2 and 3.3 describe SIBM-DE and CIBM-DE, 2 of the different possibilities for parallelizing DE by using subpopulations and multiple populations, respectively.

3.1 | Previous related work

Several existing research in parallel DE follow the island model scheme.[8] Most of them are framed within distributed memory models. Zaharie and Petcu[29] presented a proposal for solving a multiobjective optimization problem. An individual in the population can be migrated with a certain probability to a random position in a random subpopulation. Tasoulis et al[30] developed a parallel DE version, named PDE, using a ring interconnection topology and a random migration rate controlled by a parameter of the algorithm. The aim of that work was to study the implications of a controlled migration constant. Kozlov and Samsonov[31] proposed a parallel DE algorithm and applied it to solve a biological data fitting problem. It also follows a ring interconnection topology, where the replacement strategy in the migration phase is to substitute the oldest member in the target subpopulation (the one who has been the longest in the population without being replaced) by the best individual in the source subpopulation. The analysis was performed with different migration rates, and they analysed the dependency between the communication frequency and the accuracy of the optimization. Apolloni et al[32] designed a modified version of the PDE algorithm in a generic way, namely, island-based distributed differential evolution, integrating a set of 5 parameters in the migration phase. Their experimentation was based on 2 and 4 islands, and they demonstrated a higher search capacity compared with the classical DE algorithm. Zhang et al[33] explored the DE parallel capabilities with a hybrid version, named Distributed Memetic Differental Evolution (DMDE), which enhances exploration and exploitation capabilities, combining a parallel distributed DE method with two local search approaches. The subpopulations are arranged following a Von Newman topology. In the migration phase, the best individual replaces the worst neighbour. This proposal has been compared with other parallel versions (including the Apolloni's one), and it has demonstrated to achieve good per-

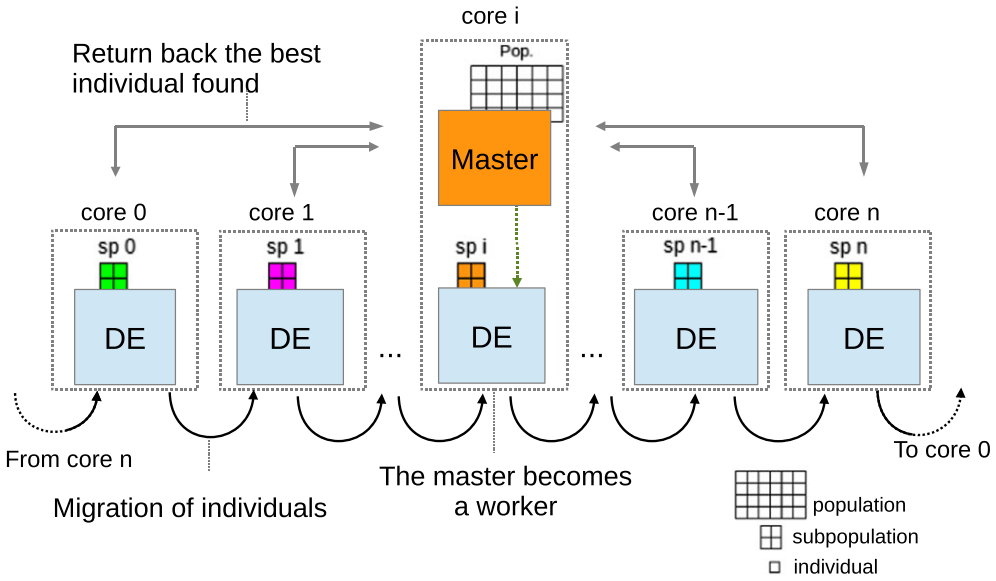

FIGURE 1 **Subpopulation Island-Based Model.** The master initializes and distributes the population. To take advantage of all the computational resources, the Master becomes a worker and collaborates in the evolutionary process

formance, generally better than the other proposals, in computing quality and convergence speed.[33] These characteristics make DMDE a competitive algorithm from the state of the art. For this reason, in Section 5.2.2, a comparison of HIBM-DE against the proposal of Zhang et al is presented.

In the following, we describe two parallel models for Differential Evolution that are based on the island model. The description of them will be helpful to better understand the HIBM-DE behaviour. Each model exploits parallelism with a specific purpose; their main characteristics and differences are highlighted.

3.2 | Subpopulation Island-Based Model for Differential Evolution

In the SIBM-DE,[26] a unique population is managed, which is initialized by a master process, and it is distributed among the worker processes, which evolve the subpopulations until reaching a certain termination condition. In this model, an island is constituted by a worker process, which evolves a subpopulation. Figure 1 describes this model. The boxes that represent an island are framed within a doted-line box, symbolizing a single computing unit. As it can be seen, the master process distributes the subpopulations among the workers. To take advantage of all the computational resources, the master retains one subpopulation and turns into a worker. Then, the workers and the master start evolving the subpopulation until a certain predefined termination condition. Every certain generations number, the processes communicate to each other following a ring topology to share some individuals from their subpopulations. The main goal of this model is to achieve a computing time reduction with regard to the sequential version. This goal is achieved precisely because the original population is partitioned in smaller parts. In consequence, each island has a subpopulation composed by a few individuals, and the workers collaborate evolving them. Although this model achieves a considerable execution time reduction, it has been shown that it does not achieve good results quality for some functions compared with the sequential version or with the CIBM-DE model, and the results quality gets worse as the subpopulation size gets smaller.[26]

3.3 | Classic Island-Based Model for Differential Evolution

In the CIBM-DE,[26] the master process does not manage any population, but it has a controlling role, indicating each worker to initialize and evolve a whole population. The workers have a harder task compared with the previous model, because each island manages a larger number of individuals (a complete population). A worker unit evolving a complete population is considered to be an island (see Figure 2). The aim of this model is to explore the search space in a more comprehensive way. This goal is reached because each island is initialized with a different seed. The CIBM-DE can be viewed as launching in parallel multiple DE sequential version instances, together with communication process among workers provided by the migration operation. In the CIBM-DE, the migration process follows a ring topology. Although this model obtains better results quality, thanks to a broader an deeper search, the migration and coordination of the processes introduce some overhead in the algorithm. In consequence, the CIMB-DE execution time is similar or even higher to the sequential version execution time, considering each single population with the same size as the sequential version.

4 | HIERARCHICAL ISLAND-BASED MODEL FOR DIFFERENTIAL EVOLUTION

In the following, we present HIBM-DE. Its processing scheme is framed within the hierarchical parallel algorithms,[8,34] and it provides two parallelism levels. One level can enhance the computational speed, whilst the other can lead to a broader problem domain coverage. These

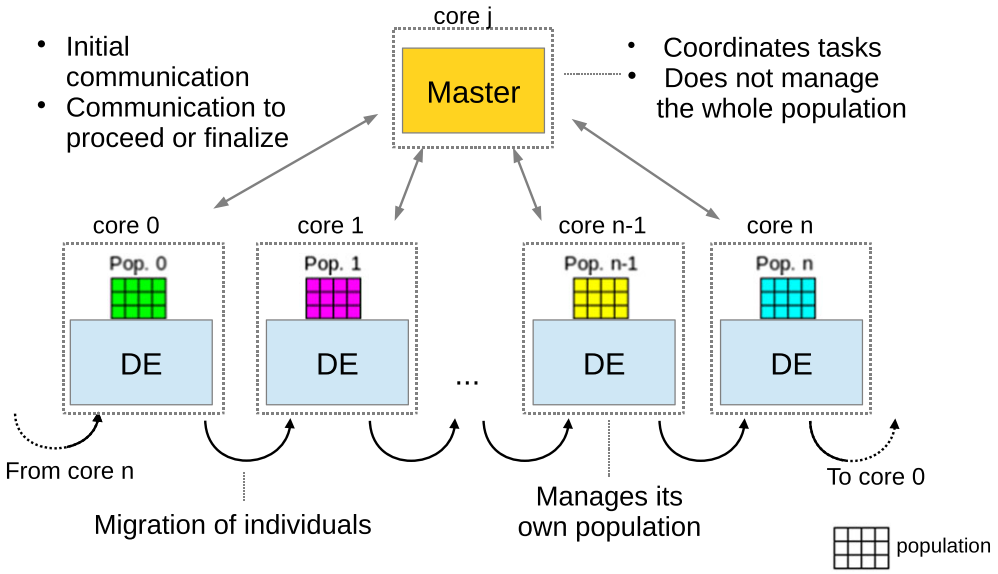

FIGURE 2 **Classic Island-Based Model.** The Master does not manage the population. Each worker initializes and handles a whole population. After each migration, the workers inform to the master the best individual found and wait for a response to proceed or finalize the evolutionary process

characteristics are inherited from the methods benefits mentioned in Sections 3.2 and 3.3, respectively.

The model comprises the CIBM-DE characteristics, considering that multiple populations are involved in the evolutionary process, and each island has a whole population initialized with a different seed. However, instead of having a unique worker responsible for evolving the population, HIBM-DE proposes a group of processes collaborating in this task, such as in SIMB-DE, with the aim of accelerating the evolutionary process in each island. In consequence, they are organized in a way that one process, named the island master, will coordinate the population evolution performed by the workers of that island.

Figure 3 describes HIBM-DE. The evolutionary process begins with the master of each island, which initializes its own population (like CIMB-DE) and distributes it among the worker processes (like SIBM-DE). The workers are related following a predefined topological order. In HIBM-DE, the order selected is a ring topological one. Each worker is in charge of evolving its subpopulation, applying all the DE operators over the individuals (mutation, crossover, and selection). A high level description of the HIBM-DE parallel algorithmic scheme follows

At one master of an island:

1. Initialize the population with seed S.
2. Distribute the population over the workers of the island.
3. repeat
4. if g is multiple of Inter-Mr /* inter island migration rate */
5. Receive individuals from all the workers of the island.
6. Send the individuals received to the master of next island
7. Receive individuals from the master of previous island.
8. Distribute the individuals received among the workers of the island.
9. end
10. until finalization condition met
11. select the best individual of the last inter-migration /*the best individual of island*/
12. send the best of the island to monitor

At one worker of an island:

1. Receive sub-population from master of island
2. Repeat
3. perform a DE step over the sub-population (mutation, crossover, selection)
4. if g is multiple of Intra-Mr
5. /* intra-island migration */
6. select the individuals that will migrate to a sub-population
7. send the individuals selected to next worker
8. choose the individuals to be replaced
9. receive the individuals from previous worker
10. integrate the individuals in sub-population
11.
12. /* send individuals to the island master (initiate the inter-island migration) */
13. select the individuals that will migrate to another island
14. send the individuals to the island master
15. end
16. if g is multiple of Inter-Mr
17. /* inter-island reception: reception of individuals from another island
18. is delayed from the send action */
19. receive the individuals from master
20. integrate the individuals in sub-population
21. end
22. Until finalization condition met

At the monitor:

1. receive the best individuals from each island
2. select the best individual of all the islands
3. output results

Every certain number of generations, and considering the topology, an intra-island migration phase is launched, in which the workers communicate to each other to send certain individuals from their subpopulations (lines 4 to 10 from worker pseudocode). The intra-island migration rate (Intra-Mr) determines the frequency at which this communication phase occurs. The amount of individuals that migrates is a certain percentage of the population; the resulting number is proportionally divided among the workers from the same island. This percentage, named Mp, is a parameter of the algorithm.

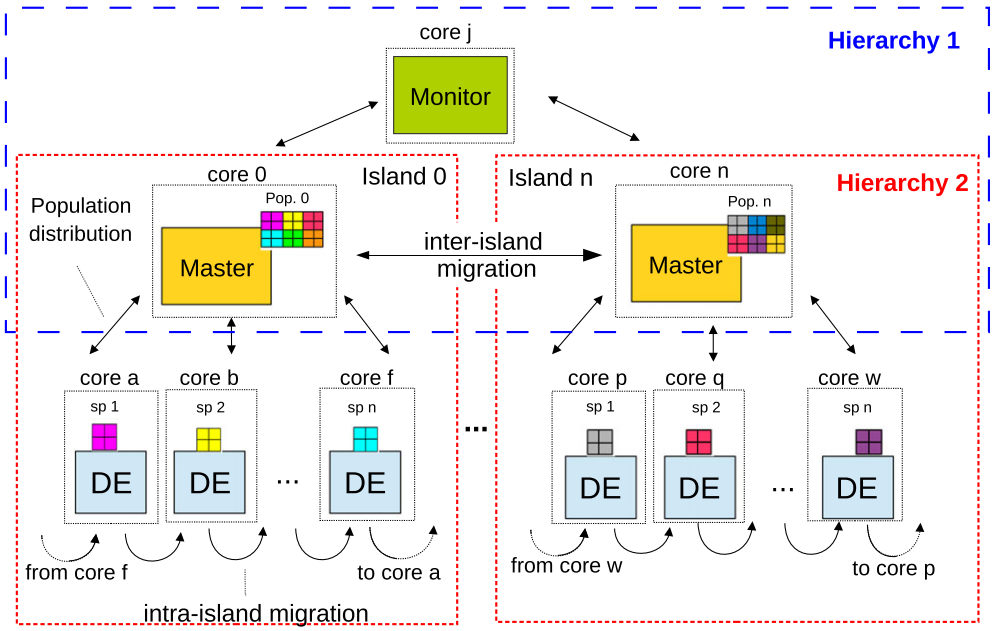

FIGURE 3 **Hierarchical Island Based Model.** Two levels of hierarchy. At bottom level, the workers evolve the islands local subpopulations. At top level, the masters communicate following a ring interconnection topology, to share global information

Like CIBM-DE and SIBM-DE, the HIBM-DE replacement strategy is semielitist, because the individuals to be migrated are the best subpopulation member (the one who has the best fitness value), and the rest is completed by means of a random selection. While the selected individuals are migrating, the worker organizes its subpopulation to determine which will be the candidates to be replaced by the newly received ones (line 8 from worker pseudocode). The received individuals will replace the worst individuals of the target subpopulation, whenever they have better objective function value than the individuals to be replaced (lines 9 and 10 from worker pseudocode).

After this intra-island communication with neighbours workers, the inter-island migration phase is launched, in which each worker communicates with the master of their island to send it to some individuals (lines 13 and 14 from worker pseudocode). The number of individuals to be sent by each worker is calculated based on the global percentage of individuals to be migrated (Mp) divided by the number of workers in each island. Like the intra-island migration, each worker selects the best individual, and the rest is completed by means of a random selection. Once this inter-island migration phase is launched, each worker resumes the evolutionary process until the master is ready for sending the individuals that arrive from another island.

The master collects all the individuals received and sends them to the master of the next island in the topological order (lines 5 and 6 from master code). Once the master has sent the individuals, it receives the same amount of individuals from the master of the predecessor island and distributes them among the workers of its island (lines 7 and 8 from master code).

It can be noticed that the inter-island migration actions that take place among the masters are performed concurrently with the workers evolutionary actions, in the sense that once the workers have sent the selected individuals to the master, they resume the cycle applying the DE operators. In this way, the workers remain active, evolving their subpopulations and minimizing idle periods. This simultaneity between workers and masters is conditioned by an appropriate setting of Inter-Mr and Intra-Mr parameters, taking into account the network speed and the amount of information to be transmitted in each migration. The amount of individuals selected for the migration process will determinate the model convergence degree. If the immigrants number is very low, the information exchange process is less efficient and the islands (and subpopulations) tend to evolve independently.[8] Instead, if there are many immigrants (Mr too high), the islands tend to converge to the same solution.[10] In our implementation, we use empirical values for all this parameters, according to the execution environment.

Figure 4 describes the collaboration actions among all the entities in the model, represented by a sequence diagram. The white boxes in the workers lifeline represent the subpopulation evolution (DE). The grey part of the box represents a communication phase: the light grey boxes symbolize the involvement in the intra-island migration whilst the dark grey boxes denote the involvement in the inter-island migration. The R boxes in the masters lifeline represent the actions needed to receive the individuals from the workers. The M part of these boxes denotes the migration of the collected individuals to another island. In the workers lifeline, R symbolizes the reception of individuals (immigrant from other island) sent by the master. In this scenario, the HIBM-DE was configured with 2 islands and 2 workers in each island. As it can be seen in the figure that both the intra-island migration and the inter-island migration occurs at a migration rate of 500 generations. However, the inter-island migration is firstly applied at the 750 generation, and in consequence, the inter-island reception in the workers is synchronized in an intermediate generation between 2 inter-island migration phases. As it can be observed from Figure 4, when the workers arrive to a generation in which the intra-island migration must occur, they exchange the individuals with another neighbour worker in the topological order. Then, they send some individuals to the island master, and they continue evolving the subpopulation until the synchronized reception of new individuals that come from another island.

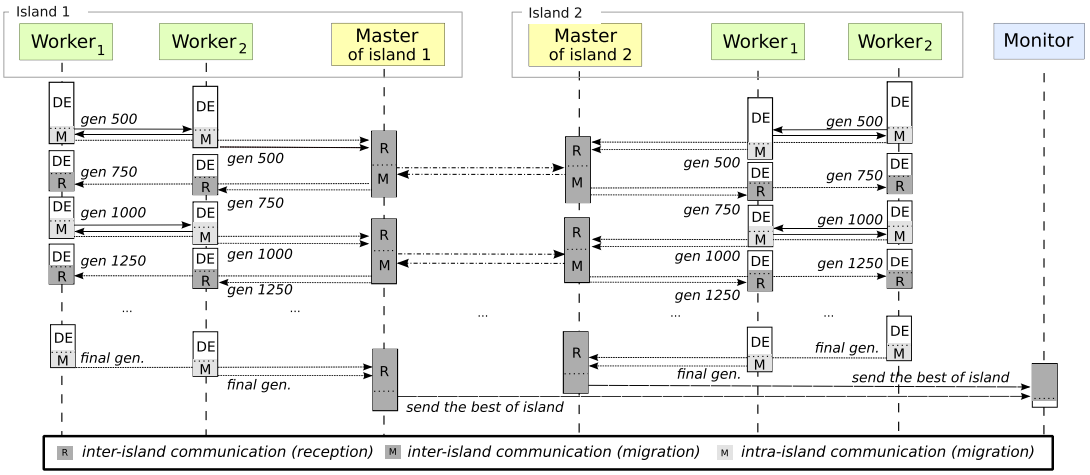

FIGURE 4 Hierarchical Island Based Model for Differential Evolution: sequence diagram example. Migration rate Intra-Mr = 500 generations. The inter-island migration is firstly applied at the 750 generation, to overlap the inter-island communication with the intra-island processing

The evolutionary process finalizes when all the islands have reached a predefined termination condition. In the HIBM-DE, this condition could be reaching a predefined number of generations or reaching certain error value. To collect the best individual found through all the islands, the model includes another process called monitor. After all the workers finish evolving their subpopulations, the master of each island receives the last group of individuals from its workers. Then, it selects the best member in the group and sends it to the monitor process (lines 11 and 12 from master pseudocode). Finally, the monitor selects the best of all the individuals received, and the evolutionary process finalises (lines 1 to 3 from monitor pseudocode).

Since HIBM-DE is based on a parallel/distributed computing model, it is possible to arrange the available computing units in several ways, depending on the purpose to be achieved. Even fixing a certain number of computing units makes it possible to adjust HIBM-DE to different islands organizations. As an example, having 16 computing units, it is possible to configure 3 islands composed of 4 workers in each island or 5 islands composed of 2 workers in each island.

In summary, this model proposes a two-level information exchange mechanism. On the one hand, the intra-island communication allows the evolution of the population to speed up by the jobs distribution among the workers. On the other hand, the inter-island communication gives the possibility to share information from other search spaces, since each island is initialized with a different seed.

The next section presents a comparative study of the three models described in this section and includes a comparison with other algorithm from the state of the art. Different performance metrics are used to analyse the obtained results.

5 | EXPERIMENTAL ANALYSIS

In this section we present a set of experiments to document the HIBM-DE analysis. The performance is analysed through different aspects. First, an overall study of the model inherent characteristics is presented, based on scalability aspects, resources consumption, and execution time. Later, a comparison against other distributed DE algorithms is presented. The tests were performed on a cluster of 11 computers having 64 bits Intel Q9550 Quad Core 2.83GHz processors and 4 Gb DDR3 1333Mz RAM memory. Each experiment used the number of computers needed so as to assign a single core to each logical unit (master, worker, or monitor). The base software on the cluster includes a 64 bits Debian 5 Lenny Operating System. Our HIBM-DE algorithmic version is based on the DE sequential version[10] and has been redesigned using the object oriented paradigm under an implementation using the C++ programming language and MPICH library[35] for message passing communication. In our implementation, each HIBM-DE component (master, worker, or monitor) is located into a separate core. In this way, each of them can be exclusively focused to the task to which it has been assigned.

5.1 | Hierarchical island-based model for differential evolution analysis

The algorithm performance was tested with a set of scalable functions.[36] Sphere and Schwefel are unimodal functions, and Rosenbrock, Rastrigrin, Griewank, and Ackley are multimodal functions.

To prevent exploitation of the search space symmetry and of the typical zero value associated with the global optimum, the global optimum is shifted to a value different from zero, and the function values of the global optimum are nonzero. The average error is defined as the difference between the global optimum value and the algorithm obtained value. A zero error indicates that the algorithm has found the global optimal solution. For the problems considered, the best results are those that are closer to zero error.

5.1.1 | The configuration

The termination criterion for all the executions was to reach 6000 generations. The SIBM-DE, CIBM-DE, and HIBM-DE were tested in previous experimentations with the aim of tuning some sensitive parameters. In the calibration, we focused on the mutation strategy, the crossover probability (Cr), and the mutation factor (F). For HIBM-DE,

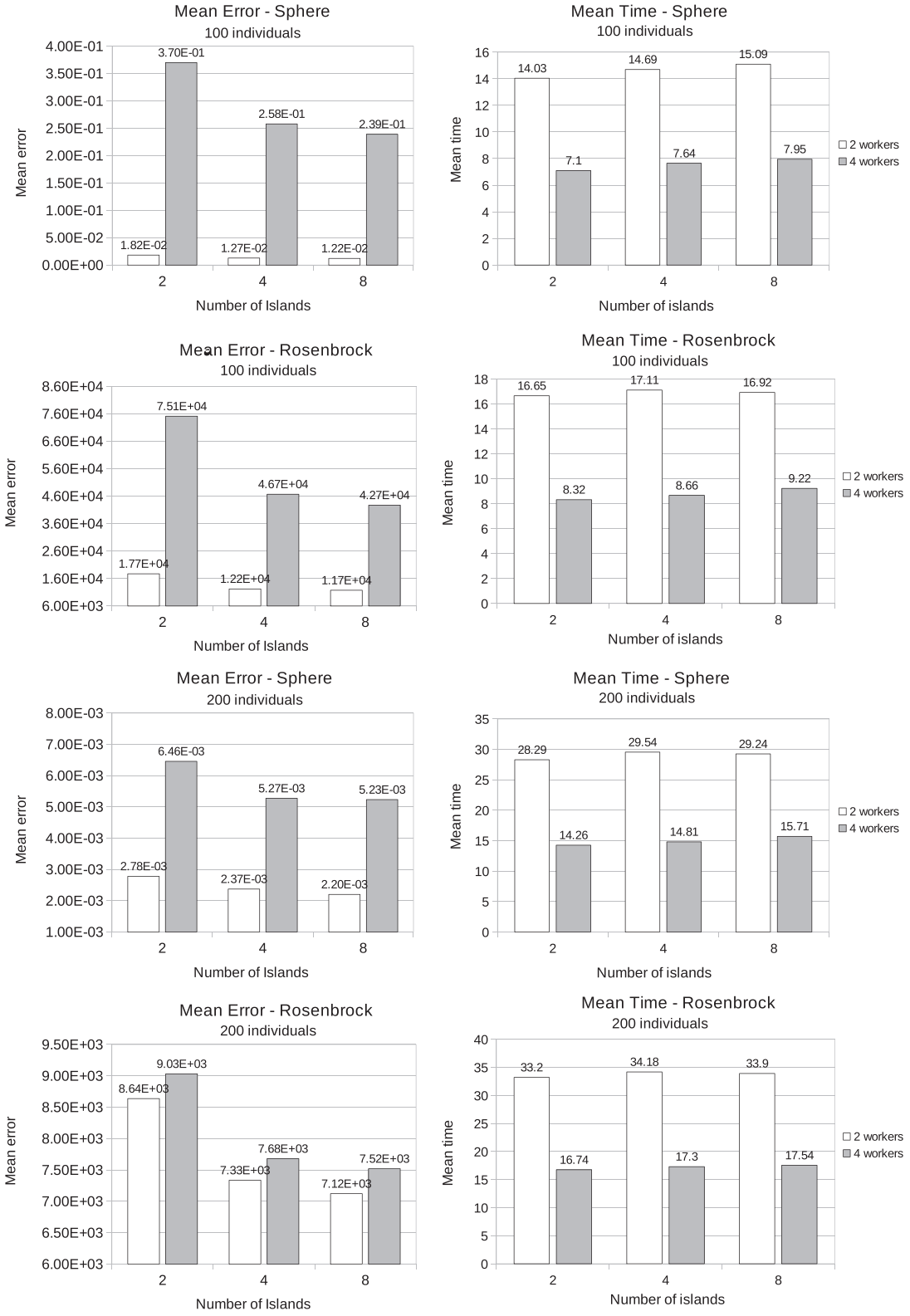

FIGURE 5 Comparison between 2, 4, and 8 islands. Mean error (left graphics) and mean execution time (right graphics) from 30 independent runs for Sphere and Rosenbrock functions. Population with 100 and 200 individuals

the strategies tested were *DE/rand/1/bin* and *DE/best/1/bin*, and better error values were obtained using the second one.

The parameters *Cr* and *F* were calibrated to obtain the best possible results for the functions considered. A static calibration for SIBM-DE and CIBM-DE was previously performed.[22] From the HIBM-DE calibration, it was found that setting F=0.5 and Cr=0.3 provides, in general, higher solutions quality for the test suite. The Intra-Mr parameter was established to 500 generations. Also, different experimental

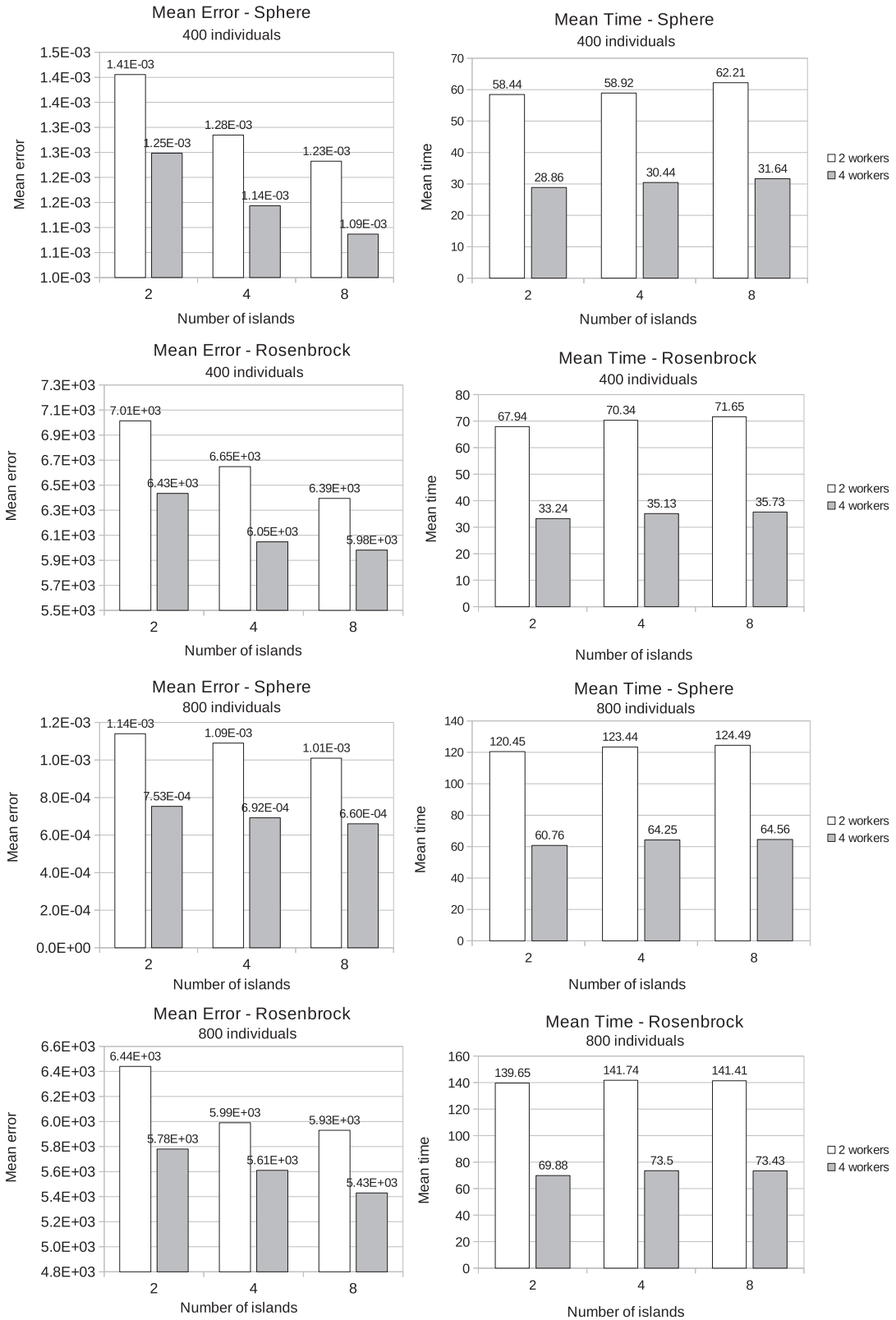

FIGURE 6 Comparison between 2, 4, and 8 islands. Mean error (left graphics) and mean execution time (right graphics) from 30 independent runs for Sphere and Rosenbrock functions. Population with 400 and 800 individuals

tests were performed, with the aim of analysing the CIBM-DE behaviour when each island is configured with different parameters.[26] However, this heterogeneous setting does not produce better results than using the same settings for all the islands. In consequence, all the islands in HIBM-DE were configured with the same parameters values.

5.1.2 | Obtained results

The following results show the HIBM-DE behaviour with different population sizes and different number of islands and workers in each island. The problems dimension considered was 1000 with a population made up with 100, 200, 400, and 800 individuals. This configuration setting is appropriate to test the method potential for solving highly complex problems. In this experimentation we present the obtained results with two functions from the test suit. They are Sphere (unimodal, search range between [−100,100]) and Rosenbrock (multimodal, search range between [−100,100]). Both were selected to have one unimodal function and one multimodal function.

Figures 5 and 6 show the mean error and mean execution time obtained with HIBM-DE, scaling the islands number in 2, 4, and 8 islands. Figure 5 shows the results having a population made up with 100 and 200 individuals. Figure 6 shows the values obtained having a population of 400 and 800 individuals. Thirty independent runs were performed for each function and model. The lowest obtained value (minimum), the average (mean), and the standard deviation of the objective values found (std-dev) are shown in Tables 1 and 2.

Perhaps the most popular use of the scalability concept reflects the ability of a parallel system to use an increasing number of processing units, trying to achieve an execution time reduction. This characteristic, applied to metaheuristics, can be seen as speeding up the search

TABLE 1 Results obtained with HIBM-DE, having a population of 100 and 200 individuals

NP (ind.)	Model	Metric	Sphere	Rosenbrock
100	HIBM-DE 2-2	minimum	1.54e-02	1.49e+04
		mean	1.82e-02	1.77e+04
		std-dev	± 2.27e-03	± 1.89e+03
	HIBM-DE 2-4	minimum	3.05e-01	6.09e+04
		mean	3.70e-01	7.51e+04
		std-dev	± 4.28e-02	± 8.19e+03
	HIBM-DE 4-2	minimum	1.08e-02	1.05e+04
		mean	1.27e-02	1.22e+04
		std-dev	± 1.19e-03	± 7.99e+02
	HIBM-DE 4-4	minimum	2.15e-01	3.72e+04
		mean	2.58e-01	4.67e+04
		std-dev	± 1.96e-02	± 3.45e+03
	HIBM-DE 8-2	minimum	1.02e-02	1.06e+04
		mean	1-22e+02	1.17e+04
		std-dev	± 9.69e-04	± 8.39e+02
	HIBM-DE 8-4	minimum	2.10e-01	3.67e+04
		mean	2.39e+01	4.27e+04
		std-dev	± 1.35e-02	± 3.21e+03
200	HIBM-DE 2-2	minimum	2.32e-03	7.70e+03
		mean	2.78e-03	8.64e+03
		std-dev	± 2.90e-04	± 6.70e+02
	HIBM-DE 2-4	minimum	5.12e-03	8.09e+03
		mean	6.46e+03	9.03e+03
		std-dev	± 5.34e-04	± 1.12e+03
	HIBM-DE 4-2	minimum	1.87e-03	6.61e+03
		mean	2.37e-03	7.33e+03
		std-dev	± 2.28e-04	± 2.92e+02
	HIBM-DE 4-4	minimum	4.50e-03	7.08e+03
		mean	5.27e-03	7.68e+03
		std-dev	± 3.21e-04	± 3.50e+02
	HIBM-DE 8-2	minimum	1.95e-03	6.53e+03
		mean	2.20e-03	7.12e+03
		std-dev	± 1.35e-04	± 2.51e+02
	HIBM-DE 8-4	minimum	4.70e-03	6.85e+03
		mean	5.23e-03	7.52e+03
		std-dev	± 2.51e-04	± 2.69e+02

Lowest obtained value (minimum), average (mean), and standard deviation of the objective values found (std-dev), obtained for 30 independent runs

TABLE 2 Results obtained with HIBM-DE, having a population of 400 and 800 individuals

NP (ind.)	Model	Metric	Sphere	Rosenbrock
400	HIBM-DE 2-2	minimum	1.21e-03	6.40e+03
		mean	1.41e+03	7.01e+03
		std-dev	± 1.47e-04	± 4.58e+02
	HIBM-DE 2-4	minimum	1.12e-03	5.83e+03
		mean	1.25e-03	6.43e+03
		std-dev	± 7.27e-05	± 3.29e+02
	HIBM-DE 4-2	minimum	1.07e-03	6.14e+03
		mean	1.28e-03	6.65e+03
		std-dev	± 8.41e-05	± 3.40e+02
	HIBM-DE 4-4	minimum	9.99e-04	5.62e+03
		mean	1.14e-03	6.05e+03
		std-dev	± 7.90e-05	± 2.17e+02
	HIBM-DE 8-2	minimum	8.91e-04	5.86e+03
		mean	1.23e-03	6.39e+03
		std-dev	± 8.81e-05	± 2.28e+02
	HIBM-DE 8-4	minimum	9.77e-04	5.62e+03
		mean	1.09e-03	5.983+03
		std-dev	± 6.07e-05	± 2.12e+02
800	HIBM-DE 2-2	minimum	8.70e-04	5.47e+03
		mean	1.14e-03	6.44e+03
		std-dev	± 1.25e-04	± 7.01e+02
	HIBM-DE 2-4	minimum	6.45e-04	5.29e+03
		mean	7.53e-04	5.78e+03
		std-dev	± 6.83e-05	± 2.68e+02
	HIBM-DE 4-2	minimum	8.49e-04	5.46e+03
		mean	1.09e-03	5.99e+03
		std-dev	± 1.34e-04	± 2.30e+02
	HIBM-DE 4-4	minimum	5.94e-04	4.91e+03
		mean	6.92e-04	5.61e+03
		std-dev	± 4.59e-05	± 3.27e+02
	HIBM-DE 8-2	minimum	8.53e-04	5.24e+03
		mean	1.01e-03	5.93e+03
		std-dev	± 7.05e-05	± 2.07e+02
	HIBM-DE 8-4	minimum	5.83e-04	5.13e+03
		mean	6.60e-04	5.43e+03
		std-dev	± 4.33e-05	± 1.66e+02

Lowest obtained value (minimum), average (mean), and standard deviation of the objective values found (std-dev), obtained for 30 independent runs

process. But another important use of this term is related to the effective utilization of the computational resources, trying to solve in parallel a larger instance of the problem, at a proportional or similar execution time. One aspect to consider in this analysis concerns the scalability of increasing the number of islands in the model. The goal is to explore a broader search area, with the expectation of obtaining better results quality. From this point of view, it can be seen from each graphic of Figures 5 and 6 that the increment of islands produces better results quality. This is because each island incorporates a new search space, since each population is configured with a different seed. In reference to this, it can be noticed a slight increase in the average execution time. This fact is associated to the overhead produced by the communications needed to exchange information among the islands, but this increment can be considered negligible with respect to the gain in the solution quality.

Comparing the results considering the population size, it is possible to see that the results quality is better when the search space is broader, although at a higher execution time. It can be seen that better mean error is obtained with populations made by 800 individuals (Figure 6).

Analysing the behaviour of the model with respect to scalability of workers per island (see the mean time graphics of Figure 5 or Figure 6), it can be seen that doubling the number of workers per island leads to a reduction of the execution time. This reduction is approximately half the time of execution. This is expected because of the problem subdivision and the intra cooperation among the involved workers.

As it can be observed from the mean error graphics, based on a comparison between Figure 5 and Figure 6, the configuration of the number of workers per island is a parameter sensitive to the population size. When the populations are small, better results are found having a configuration of few workers per island. Figure 5 considers populations made up with 100 and 200 individuals, and it is possible to see that better results are found with 2 workers per island. Adding more workers per island produces too small subpopulations, causing the reduction of the search space managed by each worker. This negatively affects the diversity or variability within a subpopulation and produces lower results quality, since there is a grater probability that the search process gets stagnated. Also, lower standard deviation values are found when using 2 workers per island (see Table 1). Therefore, in this case, using a small number of workers per island is the best choice. In addition, when the population size is increased, better results are found with more workers per island, since each worker will manage larger subpopulation. Figure 6 shows that for a population made up with 400 and 800 individuals, better error values are obtained using 4 workers per island. Also, lower standard deviation values are found in this case (see Table 2). Having a greater amount of individuals implies a broader search space area and leads to better results quality.

The above analysis suggests that HIBM-DE offers configuration flexibility, providing benefits according to what is sought, that is, obtaining the best results quality, minimizing the computing time, or maybe achieving a balance between both characteristics. An appropriate tuning of these parameters may allow to fully exploit the potential of the model. The study of tuning techniques is part of our immediate future work, where different static and dynamic strategies will be considered for automatic tuning of these features, with the aim of incorporating them to the HIBM-DE implementation.

5.2 | Comparison with other distributed DE algorithms

This section presents the comparative study of HIBM-DE and other DE parallel distributed algorithms from the state of the art, and with the sequential version of DE. Section 5.2.1 presents a performance analysis of HIBM-DE against two parallel models for DE, which have been previously described in Sections 3.2 and 3.3. Section 5.2.2 presents the comparison of HIBM-DE with the DMDE algorithm, briefly described in Section 3.1. The analysis was performed considering different configuration possibilities for HIBM-DE. A discussion of the obtained results for each comparison is also presented.

5.2.1 | Comparison of HIBM-DE, CIBM-DE, and SIBM-DE

In this subsection, we present a comparison of HIBM-DE against SIBM-DE, CIBM-DE, and the sequential version of DE. We propose two types of experiments: considering that HIBM-DE is a 2-level hierarchical model, it is possible to make a comparison taking into account the amount of processing units of both levels or considering the amount of workers when it is compared with other 1-level hierarchical models, such as SIBM-DE or CIBM-DE. The first experiment considers the same number of computing units. This comparison will allow to analyse the relation between HIBM-DE and the other models in quality and execution time gains, when using the same amount of CPUs. The second experiment presents a comparison of HIBM-DE and CIBM-DE considering both models with the same amount of workers and having a population with the same number of individuals in each island. This comparison is useful to analyse the gains of HIBM-DE against a 1-level hierarchical model like CIMB-DE. In these experiments, we use 6 functions from. one study.[36] For each function, the global optimum is shifted to a value different from zero, and the function values of the global optimum are nonzero.

Experiment 1. The first experiment is reflected in the graphics of Figures 7 and 8. The configuration parameters for these experiments were the same as those mentioned in Section 5.1. We present the execution of the models fixing the amount of computing units to 8 and 16 CPUs.

Figure 7 shows the results obtained with the models using 16 computing units. It shows the mean error (left graphics) and mean execution time (right graphics) obtained from 30 independent runs. Table 3 includes the minimum obtained value, the average, and the standard deviation values obtained from the objective function values. Each population was made up with 400 individuals. The sequential version of DE (represented by "DE" in Figure 7) was executed on a single CPU. There are 2 possible ways of configuring HIMB-DE using 16 computing units, recalling that an island is composed by a specific number of workers controlled by a master process, and all the masters are commanded by a unique monitor process. The first test case considers 3 islands, composed of 4 workers each island (HIBM-DE 3-4). The second test case considers 5 islands, composed of 3 workers each island (HIBM-DE 5-2). The best mean error value of each graphic is highlighted in bold type.

As it can be seen from Figure 7, HIBM-DE obtains better results quality compared to CIBM-DE, SIBM-DE, and the sequential version for the Sphere, Rosenbrock, Griewank, and Ackley functions and obtains

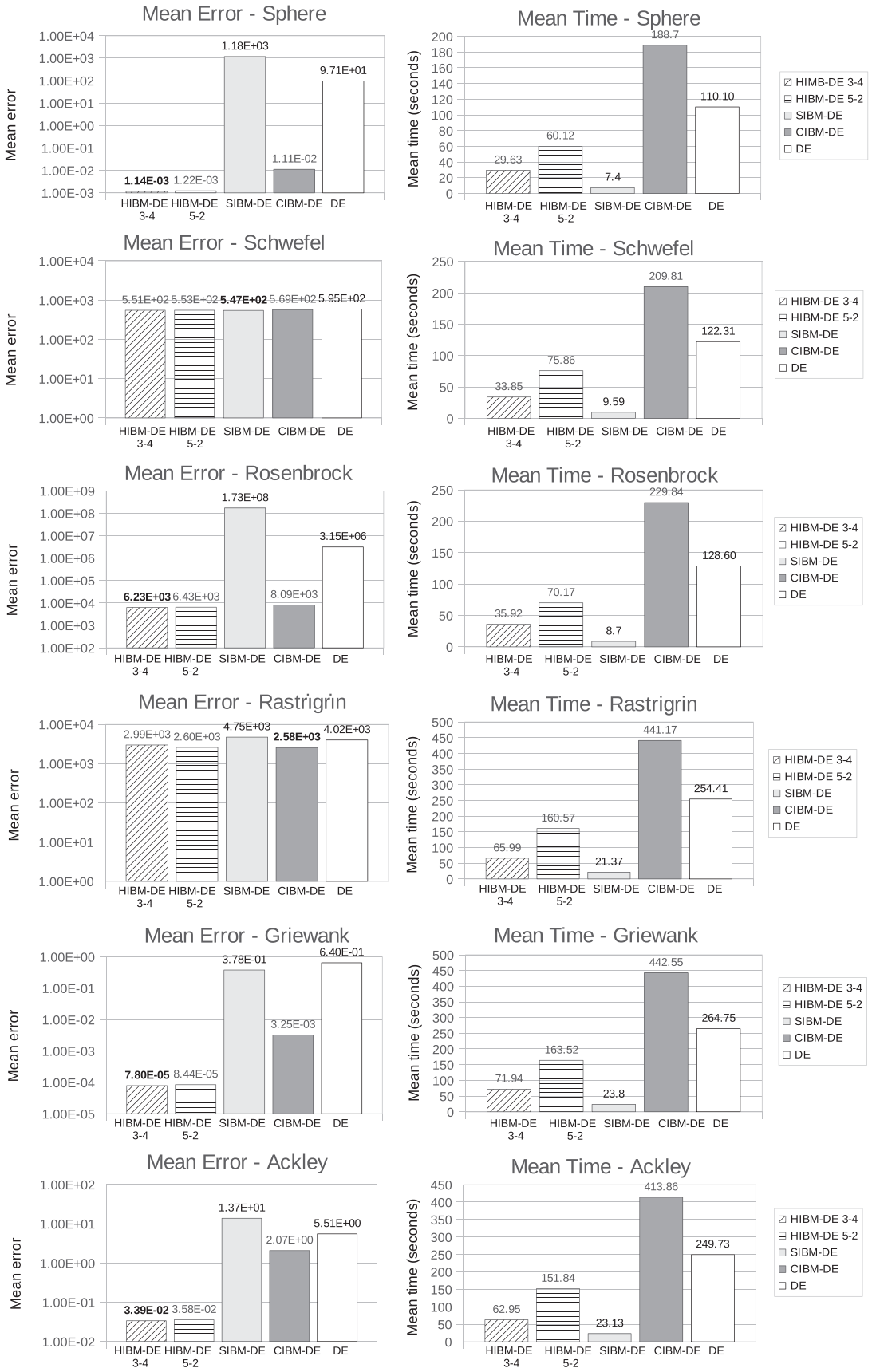

FIGURE 7 Sixteen computing units in each test case for the parallel models. Mean error (left -logarithmic scale) and mean execution time (right) from 30 independent runs. HIBM-DE was configured with 3 islands and 4 workers per island (HIBM-DE 3-4) and with 5 islands and 2 workers per island (HIBM-DE 5-2). Both CIBM-DE and SIBM-DE were configured with 16 computing units

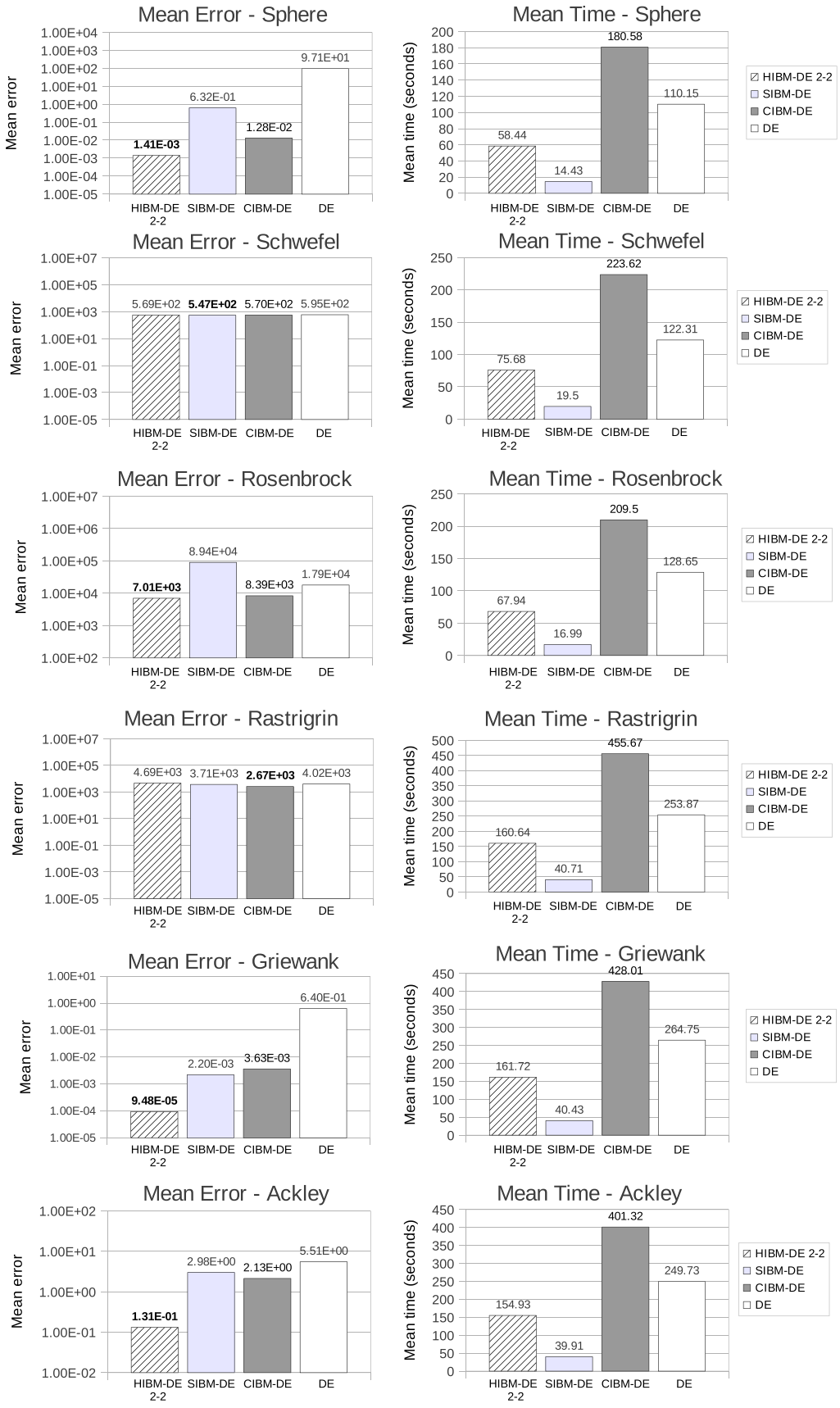

FIGURE 8 Eight computing units in each test case for the parallel models. Mean error (left - logarithmic scale) and mean execution time (right) from 30 independent runs. HIMB-DE was configured with 2 islands and 2 workers per island (HIBM-DE 2-2). Both CIBM-DE and SIBM-DE were configured with 8 computing units

TABLE 3 Sixteen islands in each model

		Sphere	Schwefel	Rosenbrock	Rastrigrin	Griewank	Ackley
HIBM-DE 3-4	minimum	9.73e-04	5.44e+02	5.79e+03	2.27e+03	6.45e-05	2.17e-02
	mean	1.14e-03	5.51e+02	6.23e+03	2.99e+03	7.80e-05	3.39e-02
	std-dev	± 8.18e-05	± 2.56e+00	± 3.99e+02	± 3.67e+02	± 5.65e-06	± 9.32e-03
HIBM-DE 5-2	minimum	1.06e-02	5.49e+02	5.98e+03	1.87e+03	7.21e-05	1.98e-02
	mean	1.22e-03	5.53e+02	6.43e+03	2.60e+03	8.44e-05	3.58e-02
	std-dev	± 1.10e-04	± 2.31e+00	± 1.92e+02	± 6.28e+02	± 8.34e-06	± 7.21e-03
SIBM-DE	minimum	9.94e+02	5.46e+02	1.10e+08	4.28e+03	2.87e-01	8.78e+00
	mean	1.18e+03	5.47e+02	1.73e+08	4.75e+03	3.78e-01	1.37e+01
	std-dev	± 9.31e+01	± 2.41e-01	± 3.07e+07	± 2.22e+02	± 5.25e-02	± 2.72e+00
CIBM-DE	minimum	8.11e-03	5.63e+02	7.07e+03	2.31e+03	2.25e-03	1.86e+00
	mean	1.11e-02	5.69e+02	8.09e+03	2.58e+03	3.25e-03	2.07e+00
	std-dev	± 2.04e-03	± 2.61e+00	± 5.70e+02	± 1.25e+02	± 5.82e-04	± 1.08e-01
DE	minimum	4.38e+00	5.88e+02	5.58e+03	3.44e+03	2.11e-01	4.31e+00
	mean	9.71e+01	5.95e+02	3.15e+06	4.02e+03	6.40e-01	5.51e+00
	std-dev	± 4.40e+02	± 2.95e+00	± 2.75e+04	± 3.23e+02	± 2.61e-01	± 7.61e-01

Lowest obtained value (minimum), average (mean), and standard deviation of the objective values found (std-dev), obtained for 30 independent runs

TABLE 4 Eight computing units

		Sphere	Schwefel	Rosenbrock	Rastrigrin	Griewank	Ackley
HIBM-DE 2-2	minimum	1.21e-03	5.64e+02	6.40e+03	2.13e+03	7.53e-05	2.14e-02
	mean	1.41e-03	5.69e+02	7.01e+03	4.69e+03	9.48e-05	1.31e-01
	std-dev	± 1.47e-04	± 2.99e+00	± 4.58e+02	± 3.26e+03	± 7.63e-06	± 2.54e-01
SIBM-DE	minimum	1.43e+01	1.44e+01	1.70e+01	3.10e+01	3.23e+01	3.21e+01
	mean	6.32e-01	5.47e+02	8.94e+04	3.71e+03	2.20e-03	2.98e+00
	std-dev	± 2.72e-01	± 2.25e+00	± 4.34e-02	± 6.08e+00	± 6.49e+00	± 6.13e+00
CIBM-DE	minimum	9.45e-03	5.63e+02	7.35e+03	2.29e+03	2.60e-03	1.73e+00
	mean	1.28e-02	5.70e+02	8.39e+03	2.67e+03	3.63e-03	2.13e+00
	std-dev	± 1.69e-03	± 3.15e+00	± 4.82e+02	± 1.53e+02	± 4.83e-04	± 1.36e-01
DE	minimum	4.38e+00	5.88e+02	5.58e+03	3.44e+03	2.11e-01	4.31e+00
	mean	9.71e+01	5.95e+02	1.79e+04	4.02e+03	6.40e-01	5.51e+00
	std-dev	± 4.40e+02	± 2.95e+00	± 2.75e+04	± 3.23e+02	± 2.61e-01	± 7.61e-01

Lowest obtained value (minimum), average (mean) and standard deviation of the objective values found (std-dev), obtained for 30 independent runs

results from the same order for the rest of the functions. In addition, HIBM-DE obtains, in most cases, lower standard deviation values than the other models (see Table 3). One important issue to consider is the execution time of the model, gaining several minutes compared to CIBM-DE. Moreover, from the graphics, it is possible to see that HIBM-DE configured with 3 islands, and 4 workers in each island obtains a significant gain in the results quality and in the execution times.

Figure 8 shows the mean error (left graphics) and mean execution time (right graphics) obtained values. In this experiment, each population has 400 individuals, and 8 computing units were used. The HIBM-DE was configured with 2 islands and 2 workers per island (HIBM-DE 2-2). In computing units, this configuration is the one closer to the 8 CPUs used for the other models, with a total of 7 computational units (3 units in each island, plus the monitor). As it can be seen, the results are similar to those arising from the experiments with 16 computing units (Figure 7, Table 4).

Experiment 2. The second experiment was performed using the 6 functions under consideration and involves a comparison of HIBM-DE with a 1-hierarchy model. This experiment was made taking into account the same amount of workers in each model. The HIBM-DE was configured with 4 islands and 4 workers per island, making a total of 16 workers. Consequently, CIBM-DE was configured with 16 islands, hence, having a total of 16 workers. We selected CIBM-DE as the 1-level hierarchy model to be compared, because this model has the characteristic of finding good results quality. Figure 9 shows mean error (graphic above) and execution time (graphic below) for the 6 benchmark functions. Table 5 includes the lowest obtained value (minimum), average (mean), and the standard deviation (std-dev) of the objective values found for 30 independent runs. It is important to highlight the trade off between the error quality and the execution time: The mean time graphic shows a significant difference in the execution times of both models. The HIBM-DE achieves a reduction of more than 80% in the execution time for all the functions compared against CIMB-DE,

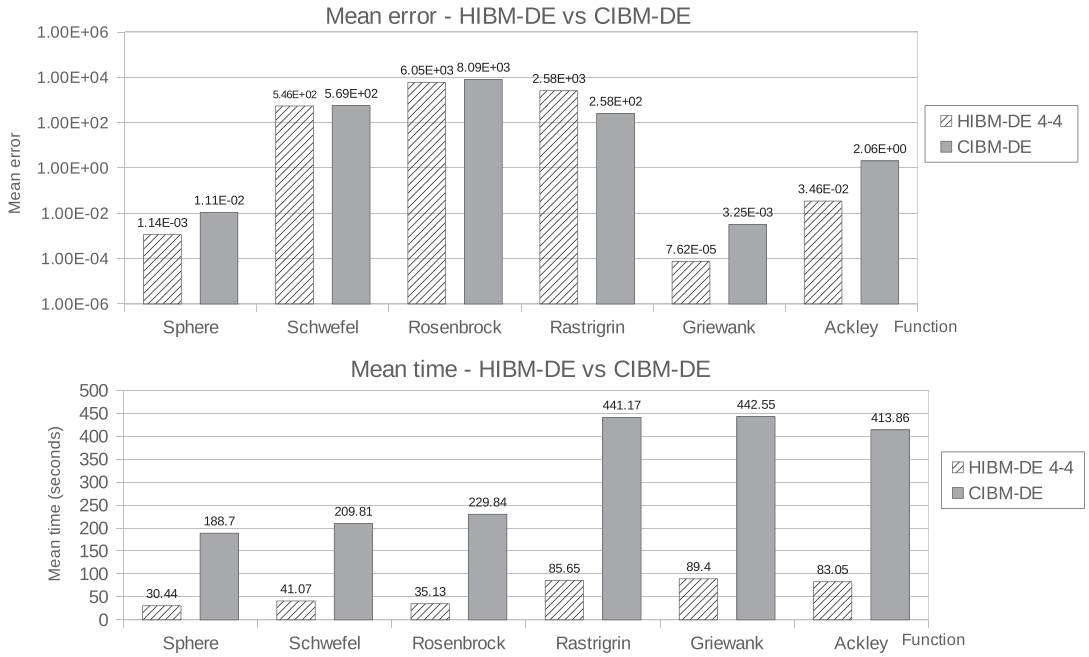

FIGURE 9 Comparison of Hierarchical Island-Based Model for Differential Evolution (HIBM-DE 4-4) and Classic Island-Based Model for Differential Evolution (CIBM-DE) with 16 islands.

TABLE 5 Sixteen islands in each model

		Sphere	Schwefel	Rosenbrock	Rastrigrin	Griewank	Ackley
HIBM-DE 4-4	minimum	9.99e-04	5.41e+02	5.62e+03	2.14e+03	6.48e-05	2.26e-02
	mean	1.14e-03	5.46e+02	6.05e+03	2.58e03	7.62e-05	3.46e-02
	std-dev	± 7.90e-05	± 2.65e+00	± 2.17e+02	± 2.61e+02	± 6.05e-06	± 8.01e-03
CIBM-DE	minimum	8.11e-03	5.63e+02	7.07e+03	2.31e+03	2.25e-03	1.86e+00
	mean	1.11e-02	5.69e+02	8.09e+03	2.58e+02	3.25e-03	2.06e+00
	std-dev	± 2.04e-03	± 2.61e+00	± 5.70e+02	± 1.25e+02	± 5.82e-04	± 1.08e-01

Lowest obtained value (minimum), average (mean), and standard deviation of the objective values found (std-dev), obtained for 30 independent runs

with similar or better error accuracy, even when HIBM-DE involves 4 islands (against the 16 islands of CIMB-DE). In addition, HIBM-DE obtains, in general, lower standard deviation values for the test functions under study (see 5).

5.2.2 | Comparison of HIBM-DE and DMDE

In this subsection, we present the comparative study of HIBM-DE and DMDE. The main characteristics of DMDE have been previously described in Section 3.1. We selected DMDE to perform the comparison because this algorithm has already been compared against other distributed DE algorithms, and it has demonstrated an excellent performance of solutions quality and convergence speed for different problems.[33]

The experiments performed try to follow the same configuration as those proposed for DMDE in one study.[33] The global optimum of the functions used in the experiments is zero, and the function values of the global optimum are zero. The dimension of the problems was set to 50. The method used for DE is *DE/rand/1/bin*. Thirty independent runs were performed for each function.

The termination condition was to reach 10^6 number of function evaluations (NFE). The population size was set to 400 individuals, and the model was configured with 4 islands and 5 workers per island. Thus, each worker manages 80 individuals, such as in DMDE. The subpopulations exchange a percentage of individuals every 100 generations. This percentage was established through the experimental analysis based on different values, and the best results led to choose the 15% as the migration percentage, ie, the individuals were exchanged among the islands at a Mr of 15% every 100 iterations. The inter-island migration

TABLE 6 Hierarchical Island-Based Model for Differential Evolution: Crossover probability (*Cr*) and mutation factor (*F*) values used with each function

Function	*Cr*	*F*
Sphere	2.0e-01	1.0e-01
Rosenbrock	9.0e-01	3.0e-01
Rastrigrin	1.0e-02	1.0e-01
Griewank	2.0e-01	5.0e-01
Ackley	2.0e-01	1.0e-01

frequency was also configured at 100 generations. The parameters *Cr* and *F* were adjusted to each function according to Table 6.

Figure 10 shows the mean error values and mean execution time obtained in the experimentation. Table 7 presents the obtained results of the lowest obtained value (minimum), average (mean), and the standard deviation of the objective values found (std-dev), obtained from 30 independent runs.

It can be seen from the results that HIBM-DE obtains a very good performance for these test problems. As it can be observed in Table 7, HIBM-DE can find the global minimum in two of the functions (Sphere and Griewank, particularly, it is important to note that HIBM-DE reaches the global minimum for Sphere, whilst DMDE only reaches a minimum in the order of e-25). Also, HIBM-DE provides the best minimal and average error values for the other 3 functions, reducing in 7 orders of magnitude the mean error for the Rastrigrin function (see Figure 10).

Figure 11 shows the average convergence trends of the algorithms for Ackley and Rastrigrin functions. The algorithms were run 30 times, and 6 points were recorded every time. For every point, the results were obtained by averaging the fitness value. As it can be

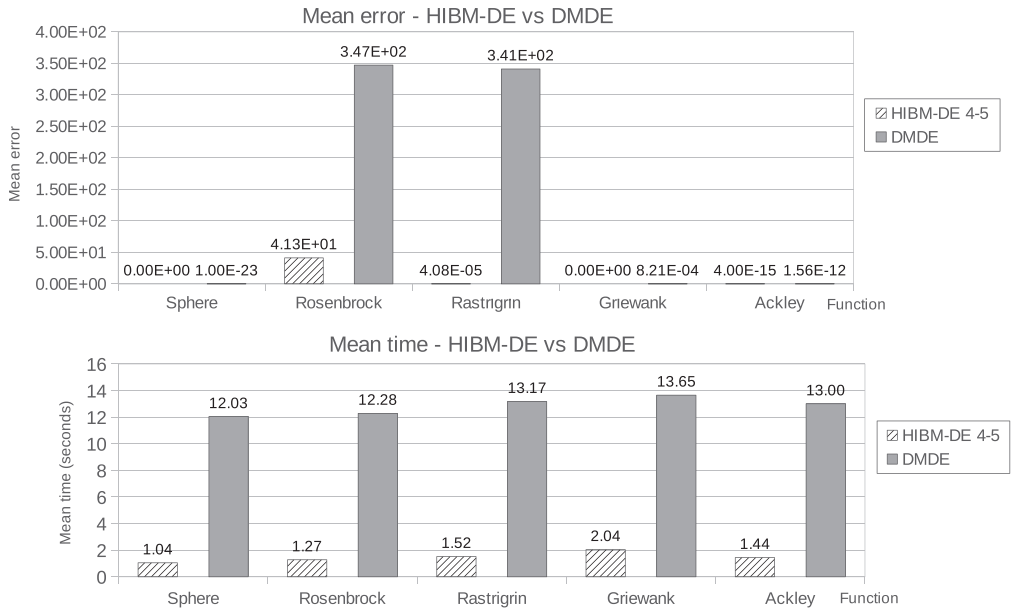

FIGURE 10 Comparison of Hierarchical Island-Based Model for Differential Evolution (HIBM-DE 4-5) and Distributed Memetic Differential Evolution (DMDE)

TABLE 7 Comparison between DMDE and HIBM-DE in terms of lowest obtained value (minimum), average (mean) and standard deviation of the objective obtained values (std-dev) for 30 independent runs

		Sphere	Rosenbrock	Rastrigrin	Griewank	Ackley
DMDE	minimum	4.93e-25	2.84e+02	2.74e+02	0.00e+00	6.55e-13
	mean	1.00e-23	3.47e+02	3.41e+02	8.21e-04	1.56e-12
	std-dev	± 2.00e-24	± 3.51e+00	± 3.93e+00	± 4.71e-04	± 1.34e-13
HIBM-DE	minimum	0.00e+00	5.08e-02	5.93e-10	0.00e+00	4.00e-15
	mean	0.00e+00	4.13e+01	4.08e-05	0.00e+00	4.00e-15
	std-dev	± 0.00e+00	± 2.31e+01	± 1.01e-04	± 0.00e+00	± 0.00e+00

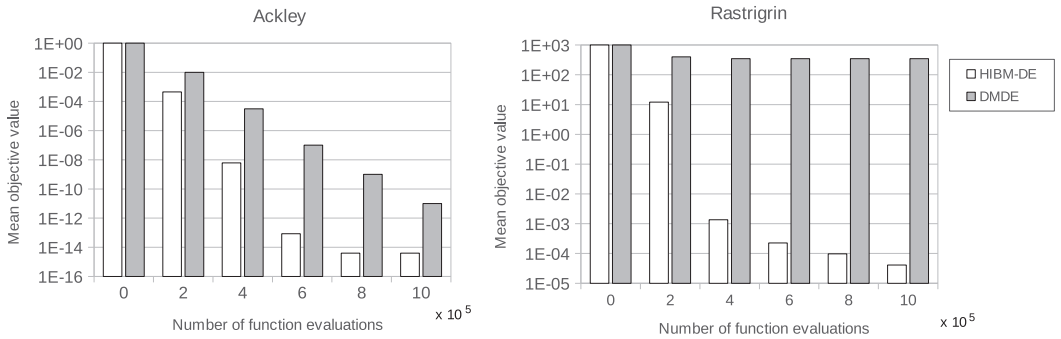

FIGURE 11 **Comparison DMDE and HIBM-DE.** Mean objective function value obtained. For every point, the results were obtained averaging the fitness value of 30 times

observed, HIBM-DE achieves good convergence speed, superior to DMDE. Beyond the gains obtained with HIBM-DE in quality, we also highlight that the mean execution time of HIBM-DE is significantly lower, achieving a reduction, on average, of more than the 92% compared against DMDE.

6 | CONCLUSIONS AND FUTURE WORK

A parallel approach to the DE algorithm named HIBM-DE was presented in this paper. The proposed method is based on an island model, following a double-hierarchy master-worker scheme. On the one hand, an intra-island migration is performed by the workers, which periodically communicate to each other to share individuals from their subpopulations. On the other hand, an inter-island migration is performed by the masters of each island, which promotes the population exchange among different search spaces.

The experimental results show that HIBM-DE provides a flexible configuration interface in which the islands can be composed by the amount of workers needed to achieve the time reduction desired, and the user may instantiate certain number of islands to achieve the results quality needed. Moreover, the model has demonstrated to be competitive with respect to other well-known parallel implementations of DE, exhibiting good quality results and at lower execution time, achieving a reduction of about 90% compared against other algorithms from the state of the art. It is important to emphasise that the versatility and adaptability of the model make it suitable for hard problems dealing with time and/or search space scalability measures.

This type of hierarchical models have several parameters that must be calibrated and adapted to the characteristics of the problem to be solved. Although these hierarchical structures are flexible in the configuration of the components, there are inherent characteristics related to the communication mechanism that must be synchronized and fixed in a suitable way to exploit the method potentialities. As future works, we plan to study different static and dynamic techniques for automatic tuning of these characteristics, in order to facilitate the configuration task to final users of HIBM-DE.

ACKNOWLEDGMENTS

This work has been supported by UTN (Argentina) under project EIUTIME0003939TC and EIUTNME0003952, and by MEC (Spain) under project TIN2014-53234-C2-1-R. The first author would like to thank CONICET for the PhD Grant provided.

REFERENCES

1. Neelima S, Subramanyam PS. Optimal capacitor placement in distribution networks using differential evolution incorporating dimension reducing power flow method. *ISGT*. 2011;396(401): 1–3.
2. Bo L, Fernandez FV, Gielen GGE. Efficient and accurate statistical analog yield optimization and variation-aware circuit sizing based on computational intelligence techniques. *IEEE T Comput Aid D*. 2011;30(6): 793–805.
3. Zhenkui P, Yanli Z, Zhen L. Image segmentation based on differential evolution algorithm. *Int. Conf. IASP*, Taizhou, China; 2009;48–51.
4. Mushtaq H, Rahnamayan S, Siddiqi A. Color separation in forensic image processing using interactive differential evolution. *J Forensic Sci*. 2015;60(1): 212–218.
5. Amin R, Jiangjun T, Ellejmi M, Kirby S, Abbass HA. Trading-off simulation fidelity and optimization accuracy in air-traffic experiments using differential evolution. *IEEE C Evol Computat*. 2014;575(482): 6–11.
6. Wang Z, Zhang X, Liu W, Liu B. Substation planning based on geographic information and differential evolution algorithm. *Int. Conf SUPERGEN*, Nanjing, China; 2009:1–7.
7. Méndez Garabetti M, Tardivo ML, Bianchini G, Caymes Scutari P. Predicción del Comportamiento de Incendios Forestales mediante un Método de Reducción de Incertidumbre basado en HPC y Evolución Diferencial. *XVI Workshop de Investigadores en Ciencias de la Computación*, Ushuaia, Tierra del Fuego, Argentina; 2014:690–694.
8. Talbi EG. *Metaheuristics: From Design to Implementation*. Hoboken, New Jersey: John Wiley & Sons; 2009.
9. Storn R, Price K. Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA; 1995.
10. Price K, Storn R, Lampinen J. *Differential Evolution: A Practical Approach to Global Optimization*. New York: Springer; 2005.
11. Zhang W, He G, Kong L. Grey Compertz prediction model based on hybrid differential evolution algorithm. *Int. Conf. Emerging research in artificial intelligence and computational intelligence*, Chegdu, China; 2012:33–40.
12. Ko GM, Reddy AS, Kumar S, Garg R, Bailey BA, Hadaegh AR. Differential evolution-binary particle swarm optimization algorithm for the analysis of Aryl Beta-Diketo Acids for HIV-1 integrase inhibition. *IEEE Congress on Evolutionary Computation*, Brisbane, Austalia; 2012:1–7.
13. Pohlmann A, Lesmann M, Hameyer K. Comparative study on optimization methods for a motor-drive of artificial hearts. *Int. Conference on Electrical Machines and Systems*, Incheon, South Korea; 2010:1754–1758.
14. Wong GY, Leung FHF, Ling S. Predicting protein-ligand binding site with differential evolution and support vector machine. *The 2012 International Joint Conference on Neural Networks*, Brisbane, Australia; 2012:1–6.
15. Amjady N, Keynia F, Zareipour H. Short-term load forecast of microgrids by a new bilevel prediction strategy. *IEEE Trans Smart Grid*. 2010;1(3):286–294.
16. Chen ML, Wang FS. Fuzzy optimization for a batch simultaneous saccharification and co-fermentation process by hybrid differential evolution. *IEEE C Evol Computat*. Brisbane, Australia; 2012:1–8.
17. Qu J, Cao L, Zhou J. Differential evolution-optimized general regression neural network and application to forecasting water demand in yellow river Basin. *International Conference on Information Science and Engineering*, Hangzhou, China; 2010:1129–1132.
18. Zhao Z, Wang J, Zhao J, Su Z. Using a Grey model optimized by differential evolution algorithm to forecast the per capita annual net income of rural households in China. *Int J Manag Sci*. 2012;40(5):525–532.
19. Lee M, Cho S. Interactive differential evolution for image enhancement application in smart phone. *IEEE C Evol Computat*, Brisbane, Australia; 2012:1–6.
20. Brest J, Zamuda A, Bokovie B, Maucec MS, Zumer V. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. *IEEE C Evol Computat*, Hong Kong; 2008:2032–2039.
21. Yang Z, Tang K, Yao X. Self-adaptative differential evolution with neighborhood search. *IEEE C Evol Computat*, Hong Kong; 2008:1110–1116.
22. Tardivo ML, Cagnina L, Leguizamón G. A hybrid metaheuristic based on differential evolution and local search with quadratic interpolation. *Proc. of the XVIII Congreso Argentino de Ciencias de la Computación*, Bahía Blanca, Argentina; 2012:100–109.
23. García-Martínez C, Rodríguez FJ, Lozano M. Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. *J Soft Comput, Springer*. 2010;15(11): 2109–2126.
24. Zamuda A, Brest J, Bokovie B, Zumer V. Large scale global optimization using differential evolution with self adaptation and cooperative co-evolution. *IEEE C Evol Computat*, Hong Kong; 2008:3718–3725.

25. Izzo D, Rucinski M, Ampatzis C. Parallel global optimization meta-heuristics using an asynchronous island-model. *IEEE Congress on Evolutionary Computation*, Trondheim, Norway; 2009:2301–2308.
26. Tardivo ML, Caymes-Scutari P, Méndez-Garabetti M, Bianchini G. Two models for parallel differential evolution. *Proc. of the High Performance Computing Latin American Symposium*, Mendoza, Argentina; 2013:26–36.
27. Kemeny JG, Snell JL, Thompson GL. *Introduction to Finite Mathematics*. 3rd ed., chapter 4. New Jersey: Prentice Hall; 1974.
28. Mattson T, Sanders B, Massingill B. *Patterns for Parallel Programming*, chapter 5: Addison-Wesley Professional, Boston; 2004:143–152.
29. Zaharie D, Petcu D. Adaptive Pareto differential evolution and its parallelization. In R Wyrzykowski, J Dongarra, M Paprzycki, & J Waśniewski, eds., *5th International Conference PPAM 2003*, Czestochowa, Poland: Springer Berlin Heidelberg; 2003:261–268.
30. Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN. Parallel differential evolution. *Proc C Evol Computat*. 2004;2: 2023–2029.
31. Kozlov KN, Samsonov AM. New migration scheme for parallel differential evolution. *Proceedings of the Fifth International Conference on Bioinformatics of Genome Regulation and Structure*, Novosibirsk, Russia; 2006:141–144.
32. Apolloni J, Leguizamón G, García Nieto J, Alba E. Island based distributed differential evolution: an experiment study on hybrid testbeds. *Eighth International Conference on Hybrid Intelligent Systems*, Barcelona, España; 2008:696–701.
33. Zhang C, Chen J, Xin B. Distributed memetic differential evolution with the synergy of Lamarchian and Baldwinian learning. *J Appl Soft Comput*. 2013;13: 2947–2959.
34. Cantú-Paz E. A survey of parallel genetic algorithms. In Hermes Science Publications, ed. *Calculateurs Parallèles Réseaux et Systèmes Répartis*. 1998;10(2): 141–171.
35. MPICH Message Passing Interface. http://www.mpich.org/. Accessed 2 February 2016
36. Tang K, Yao Z, Suganthan PN, et al. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical Report, China, Nature Inspired Computation and Applications Laboratory, USTC; 2007.

How to cite this article: Tardivo ML, Caymes-Scutari P, Bianchini G, Méndez-Garabetti M. Hierarchical parallel model for improving performance on differential evolution. *Concurrency Computat: Pract Exper*. 2017;29:e4087. https://doi.org/10.1002/cpe.4087