# Parameters Calibration for Parallel Differential Evolution based on Islands*

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

[1] Departamento de Computación, Universidad Nacional de Río Cuarto. (X5804BYA) Río Cuarto, Córdoba, Argentina
lauratardivo@dc.exa.unrc.edu.ar

[2] Laboratorio de Investigación en Cómputo Paralelo/Distribuido (LICPaD) Departamento de Ingeniería en Sistemas de Información, Facultad Regional Mendoza Universidad Tecnológica Nacional. (M5502AJE) Mendoza, Argentina
{pcaymesscutari,gbianchini}@frm.utn.edu.ar
miguelmendezgarabetti@gmail.com

[3] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

**Abstract.** We are studying different alternatives to obtain a version of the Differential Evolution (DE) algorithm that improves the solutions quality properties. One of the parallel alternatives, named Island Model, follows a Master/Worker scheme. With this model, multiple instances of DE are executed in parallel on various computing nodes or *islands*, each of them considering a different population of individuals. Each worker makes the search process, and communicates with each other to exchange information with certain frequency. This model significantly promote the exploration of a larger search space, which leads to good solutions quality. The aim of this paper is to analyse the behaviour of this model, when setting each island with different input parameters. We apply some input configuration tests for the islands, in order to analyse the impact in the solutions quality and the execution time, taking into account the crossover probability and mutation factor, and the crossing type. These parameters are crucial to guide the search towards certain areas of the search space.

## 1 Introduction

The interest in solving combinatorial optimization problems has gained popularity between the scientific and industrial community [15] [4]. Between the strategies developed to solve these problems we find specific heuristics and metaheuristics as popular techniques. The specific heuristics are problem dependent and are designed in a particular way to solve a given problem. Meanwhile, metaheuritics represent a more general set of solutions that can be applied to a large

* This work has been supported by UTN under projects PICT2010/12 and UTN1585, and by ANPCyT under project PRH PICT-2008-00242.

number of problems. Metaheuristics try to solve instances of the problem, exploring the wide space of solutions that those instances can admit [10]. Usually, these solutions are called optimum, in reference to the better or best values found for the optimization problem involved, leading to local (better solutions) or global (the best solution) optimum. Although obtaining the global optimum is desirable, sometimes the function to optimize is sufficiently complex to find the desired value within a reasonable time. For this reason, obtaining good quality "local optima" becomes a valid alternative.

The Differential Evolution (DE) algorithm is a population based metaheuristic, capable of working reliably in nonlinear and multimodal environments [9]. It starts to explore the search space by initializing multiple, randomly chosen initial points distributed in D-dimentional vectors that represent the individuals of the population. The algorithm presents two operators responsible for the creation of new solutions. First, the mutation operation creates a trial vector as a linear combination of some members of the population, then the crossover operation combines the parameter values of the trial vector with those of another member of the population, resulting in the target vector.

The classic version of DE follows a sequential processing scheme. However, the Differential Evolution algorithm (and in general metaheuristics) are naturally prone to parallelism, because most variation operations can be undertaken in parallel [1]. There are several studies that incorporate parallelism to DE, in order to improve the quality of the solutions obtained and/or diminish the execution time. In this work we follow the first objective. With the aim of improving the quality of solutions by exploring a larger sample domain, we focus on the *Island Model* [10]. Even though the execution time is similar to the sequential one, each island in the model is responsible for the evolution of the population that manages, and may use different parameter values and different strategies for any search component such as selection, replacement, variation operators (mutation, crossover), and encodings. An appropriate choice for its values may achieve quality and/or performance improvements.

In this work we present a study on the calibration for some parameters of the parallel *Island Model*, through the experimental study with various input configurations for each island of the model. The aim is to analyse the impact produced by these configurations, taking into account the quality of the solutions and the execution time of the parallel algorithm. Specifically, we focus on three of the most important input parameters of DE. They are the crossover probability and mutation factor, and the crossover type. It is known that the choice of their optimal values is an application dependent task. For two optimization benchmark problems under study, we want to get an overview about the behaviour of this model applied to solve them, considering different scenarios.

The paper is organized as follows: Section 2 describes the main characteristics of DE. Section 3 present a complete description of the *Island Model* used in this work, including its main features and its processing scheme. Section 4 shows the experiments carried out and the analysis of results. Finally, we present the conclusions and future work.

## 2 Classical Differential Evolution

The Differential Evolution algorithm has emerged as a popular choice for solving global optimization problems . Using a few parameters, it exhibits an overall excellent performance for a wide range of benchmark as well as real-world application problems [2]. Each individual belongs to a generation $g$, i.e., let $X_{i,g} = (x^1_{i,g}, ..., x^D_{i,g})$ an individual of the population, with $i = 1, ...,$N where the index $i$ denotes *i-th* population individual and N is the total number of individuals in the population. Following, we explain the three classic main operators of DE. In section 3 we also introduce the *migration* operator, which is typically used in parallel versions of metaheuristics.

***Mutation:*** After initialization, DE mutates and recombines the current population to produce another one constituted by N individuals. The mutation process begins in each generation selecting random individuals $X_{r_1,g}, X_{r_2,g}$. The *i-th* individual is perturbed using the strategy of the formula (1), where the indexes $i$, $r_1$ and $r_2$ are integers numbers different from each other, randomly generated in the range [1, N].

$$\text{“}DE/best/1\text{”} : V_{i,g+1} = X_{best,g} + (X_{r_1,g} - X_{r_2,g})F \tag{1}$$

The constant $F$ represents a scaling factor and controls the difference amplification between individuals $r_1$ and $r_2$. It is used to avoid stagnation in the search process. $X_{best,g}$ is the best individual, i.e., it has the best value of the objective function evaluation among all individuals of current generation $g$. The notation “$DE/best/1$” represents that the base vector chosen is the best individual, and “1” vector difference is added to it.

***Crossover*:** After the mutation phase, each perturbed individual $V_{i,g+1} = (v^1_{i,g+1}, ..., v^D_{i,g+1})$ and the individual $X_{i,g} = (x^1_{i,g}, ..., x^D_{i,g})$ are involved in the crossover operation, generating a new vector $U_{i,g+1} = (u^1_{i,g+1}, ..., u^D_{i,g+1})$, denominated “trial vector”, and obtained using the expression (2).

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

where $j = 1, ..., D$, and $k \in \{1, ..., D\}$. The latter is a randomly generated index chosen for each individual. This index is used to ensure that the trial vector is not exactly equal to its source vector $X_{i,g}$, then a vector component at position $k$ is taken from the mutated vector. The constant $Cr$, denominated *crossover factor*, is a parameter of the algorithm defined by the user. $Cr$ belongs to the range $[0, 1]$ and is used to control the values fraction that are copied from the mutant vector $V$. $rand_j$ is the output of an uniformly distributed random number generator, and is generated for each component $U^j_{i,g+1}$ of the trial vector.

There are two crossing operators that can be applied: binomial or exponential. Both types use the expression (2), but differ in the way it is applied. The binomial crossover operator iterates over all the components of the individual, copying the $j$th 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 $X_{i,g}$. Instead, the exponential crossover operator inherits the parameters of trial vector $U_{i,g+1}$ from the corresponding mutant vector $V_{i,g}$ starting from a randomly chosen parameter index, until the $j$th parameter value satisfying $rand_j > Cr$. The remaining parameters of the trial vector $U_{i,g+1}$ are copied from the corresponding target vector $X_{i,g}$.

***Selection*:** This phase determines which element 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 next generation. The scheme followed is presented in the expression (3).

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

The three stages mentioned above are repeated from generation to generation until the specified termination criterion is satisfied. This criterion could be finding a predefined minimal error or reaching a certain number of iterations.

Due to the potentialities provided by DE numerous variations and methods have been proposed with the aim of improving the performance of the classic technique. Among them are those trying to adapt the parameters of the algorithm, such as self-adjusting [14], [12], [3]; others using different mechanisms to optimize the individuals selection for the mutation and crossover phases [4], and some combining both methods [15]. In the next section we briefly mention some related works on parallel DE, and we present the details of our parallel proposal.

## 3 Island Parallel Model for Differential Evolution

Researchers have proposed different approaches to parallelize population-based metaheuristics, depending on the purpose to be achieved. In [16] is presented a proposal for solving the Pareto front problem. An individual in the population can be migrated with a certain probability to a random position in a random subpopulation. In [11], the model uses a ring interconnection topology and random migration rate controlled by a parameter of the algorithm. The aim of that work is to study the implications of a controlled migration constant. In [5], a parallel DE version is proposed and applied to solve biological systems. It also follows a ring interconnection topology. The analysis was done with different migration rates and they conclude by identifying the best of them. A critical issue of the last three approaches, is the consideration of the population size. The ability of the algorithm to find a solution depends on the tasks size and is related to the amount of individuals per node, making significant local and global evolution. Then, it is relevant to make experiments that encompass these scenarios. For all these reasons arises the need to perform a comparative study to test with large enough cases.

Following, we will describe the ***Island Parallel Model***. It follows a *Master-Worker* [6] scheme. Multiple instances of DE are executed in parallel on different computing nodes, each one considering a different population of individuals (Pop. 0, ..., Pop. n). We call each computing node "an island". A master process is in charge of monitoring the system as a whole, and each worker process is dedicated to compute all the generations in that island. Figure 1 represents this model. As can be seen, the master process is located in an exclusive computing node, so as to coordinate the system and to avoid delaying the response to the workers.

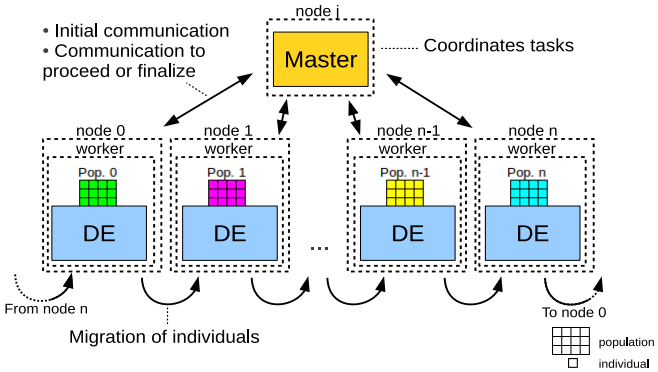


Fig. 1: **Island Model:** independent or cooperating self-contained metaheuristics.

Every certain number of generations, and considering a certain topology, begins a ***migration phase***. The amount of individuals that migrate is a certain percentage of the whole population, calculated from the number of individuals in the island. This *migration phase* represents another operator for parallel DE, and its importance lies in the need to exchange information between the islands to keep global information.

After a migration phase and replacement process, the workers inform to the master which is the best individual found. The master receives this information and temporarily stores the best individual of all those who have been sent by the workers. Then, if the termination condition is met, the master sends a message to workers indicating the end of the process. Otherwise, the master informs to continue with their evolutionary process. In our proposal, the finalization condition was defined as reaching a certain number of generations.

We recall that the Island Model significantly promote the exploration of a larger search space, because the workers explore different search spaces, since each population is initialized with a different seed. This leads to better solutions quality, although the execution time is generally higher than that of the sequential version. This parallelism technique, where multiple instances of an algorithm are launched in parallel and interrelated is useful when the aim is to deepen the search, with no particular requirements for reducing the execution time.

The following section will describe some experiments made with the aim of analysing the solutions quality and the execution time when introducing certain configurations for each island. The goal of this calibration is to adjust the effec-

tiveness of the model, considering each population with different configurations for the mutation factor and crossover probability, and the crossing type. These parameters are crucial to guide the search towards certain areas of whole search space. If these parameters are set in an inappropriate manner, it may happen that the algorithm get stagnated in a local optimum, or the solutions quality obtained may be non optimal.

## 4 Test cases and analysis of results

In the following, we describe the experiments carried out in order to test the Island Model with different configurations. In the experiments, the performance of the algorithm was tested with a set of scalable functions, obtained from [13]. For each of them, 30 executions were carried out with different seeds. The sizes of the problems considered have dimensions 100, 500 and 1000. The population was made up with 100 and 400 individuals. The function used for the test where Shifted Sphere (unimodal, search range in [-100,100], bias value of -450), and Shifted Rosenbrock (multimodal, search range in [-100,100], bias value of 390).

The average error is defined as the difference between the current value of the global optimum and the value obtained by the algorithm. If the error is zero indicates that it has been found the global optimum. For the problems considered, the best results are those that are closer to zero error.

Preliminary experiments carried out on the model conduced to a definition for the exchange rate value. In all the tests the individuals were exchanged among the islands at a migration rate of 15% every 500 iterations. It is known from literature [4], [15], [14] that the values $F$=0.5 and $Cr$=0.3 may guide the search towards good solutions. We validated and established those values in the tests.

Four experiments were carried out; some of them are associated to the crossover probability and mutation factor, and others are related to the crossover type. Although there exists a wide range of combinations and possibilities of variation on these parameters, carrying out the test and processing the results are time-consuming actions. Our test are performed with large enough dimensions, to contemplate complex optimization problems. This is an important feature, that differentiates our case of analysis regard to those cases treated in other similar studies (such as those referenced before).

Following, we provide a brief description of the test cases:

- **Case 1:** This experiment consisted in the configuration of all the islands with the same input parameters (i.e. just varying the initial seed for each island). The goal is to explore the space more thoroughly. All islands try to solve the whole optimization problem searching in a different area of the search space, having the same configuration for the rest of the parameters.

- **Case 2:** This experiment consisted in setting the half of the islands in the model with random values for the mutation and crossover probabilities, and the other half of the islands used the fixed constant values for $F$ and $Cr$.

- **Case 3:** The third experiment was performed with the aim of represent an independent behaviour of the islands, setting the input probabilities with

random values for each island in the model. This randomized configuration may reproduce a realistic scenario when the model observed is similar to a concrete natural system, where each population have its own working method. In this sense, the complete problem is solved by different entities, having their own search space and a unique search configuration.

- **Case 4:** The last experiment involved the crossover type. All the islands were setted with the constant values for $F$ and $Cr$. In this experiment we changed the crossover type to exponential crossover. With this case, we test the behaviour of the islands when the crossover type is distinct from the classic binomial one, verifying if it may conduct the search process towards other areas. This experiment can be contrasted against the Case 1.

The cases 1, 2 and 3, were performed using a binomial crossover type. Also, we can notice that the case 2 is middle point test between case 1 and case 3, trying to produce an hybrid scenario.

The islands follows a ring intercommunication topology, so that each island receives individuals from its predecessor in the topological order, and sends their own individuals to its successor in that order. The individuals to be migrated are the best member of the island plus other individuals randomly selected, and the received individuals will replace the worst members of the target population.

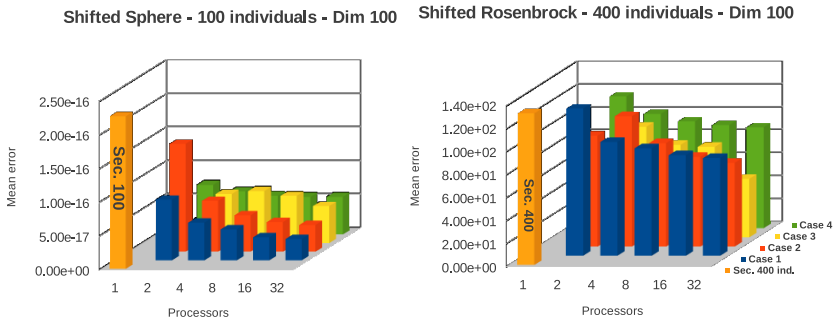


Fig. 2: Mean error of Shifted Sphere and Shifted Rosenbrock functions. Dimension 100.

In order to test scalability, all experiments included 2, 4, 8, 16 and 32 processors dedicated to the worker processes, and a separate processor for the master process. All tests were made on a cluster with 36 CPUs distributed between 9 nodes. They have 64 bits with Intel Q9550 Quad Core 2.83GHz processors and RAM memory of 4GB DDR3 1333Mz. All the nodes are connected together by Ethernet segments and switch Linksys SLM2048 of 1Gb. Base software on the cluster includes a 64 bits Debian 5 Lenny Operating System. In the codification we use the MPICH library [7] for message passing communication between participating nodes. Our algorithmic version of the Island Model is based on the sequential version of DE, obtained from [9].

Table 1 shows the average computing time, discriminating the tests according to the dimension and case analysed. The graphs of the figures 2, 3 and 4 show the mean errors obtained in the different experiments performed. In the graphs, each color represents one of the experiments mentioned above. In order

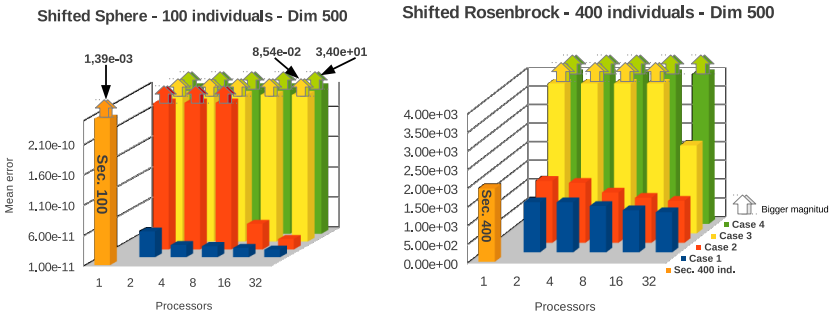


Fig. 3: Mean error of Shifted Sphere and Shifted Rosenbrock functions. Dimension 500.

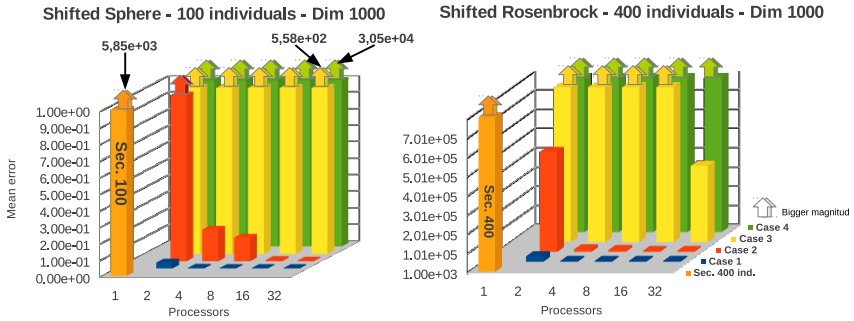


Fig. 4: Mean error of Shifted Sphere and Shifted Rosenbrock functions. Dimension 1000.

Table 1: Shifted Shpere and Shifted Rosenbrock average computing time (in seconds), obtained with the algorithm with each version.

| Shifted Sphere 100 ind. | | | | | | Shifted Rosenbrock 400 ind. | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Test case | 2 | 4 | 8 | 16 | 32 | Test case | 2 | 4 | 8 | 16 | 32 |
| **Dim 100 Sequential time:**=2.08 | | | | | | **Dim 100 Sequential time:**=12.82 | | | | | |
| Case 1 | 2,68 | 2,71 | 2,83 | 2,91 | 4,27 | Case 1 | 12,71 | 13,00 | 13,35 | 13,77 | 19,02 |
| Case 2 | 2,78 | 2,73 | 3,00 | 3,05 | 3,09 | Case 2 | 12,81 | 13,34 | 14,11 | 14,02 | 14,20 |
| Case 3 | 2,81 | 3,01 | 3,04 | 3,04 | 3,11 | Case 3 | 13,35 | 14,33 | 14,29 | 14,45 | 14,56 |
| Case 4 | 0,41 | 0,42 | 0,42 | 0,48 | 0,57 | Case 4 | 3,64 | 3,66 | 5,43 | 7,83 | 10,00 |
| **Dim 500 Sequential time:**=11.06 | | | | | | **Dim 500 Sequential time:**=63.24 | | | | | |
| Case 1 | 12,98 | 13,23 | 13,60 | 14,66 | 19,30 | Case 1 | 64,39 | 82,11 | 84,42 | 91,83 | 109,11 |
| Case 2 | 13,50 | 13,88 | 14,48 | 14,56 | 15,13 | Case 2 | 69,44 | 84,61 | 92,98 | 95,07 | 99,47 |
| Case 3 | 13,42 | 14,52 | 14,44 | 14,48 | 14,83 | Case 3 | 74,75 | 87,86 | 84,23 | 98,77 | 97,46 |
| Case 4 | 1,44 | 1,46 | 1,53 | 1,64 | 2,03 | Case 4 | 18,84 | 23,17 | 22,80 | 26,40 | 32,78 |
| **Dim 1000 Sequential time:**=22.57 | | | | | | **Dim 1000 Sequential time:**=128.64 | | | | | |
| Case 1 | 26,03 | 26,85 | 28,18 | 29,37 | 40,10 | Case 1 | 132,89 | 181,16 | 186,64 | 200,62 | 207,92 |
| Case 2 | 27,08 | 28,47 | 29,11 | 30,08 | 39,48 | Case 2 | 191,09 | 180,70 | 209,80 | 207,78 | 292,29 |
| Case 3 | 28,61 | 30,60 | 30,67 | 34,01 | 32,77 | Case 3 | 189,47 | 172,69 | 188,78 | 196,14 | 212,53 |
| Case 4 | 3,12 | 3,25 | 3,53 | 5,17 | 7,69 | Case 4 | 65,68 | 67,56 | 61,32 | 62,92 | 74,05 |

to contrast with the parallel experiments, the graphs also include two columns that represents the mean error for the sequential version. Some bar columns of the graphics have a colored arrow at top, representing that the column bar has a bigger magnitude that the maximum scale in the graphic. Moreover, we include some small labeled black arrows with the purpose of explicitly indicate the value of those big columns or to highlight some interesting value.

In first place we compare the results obtained for cases 1, 2 and 3. As can be seen, the execution time for them are similar. The test that obtains better quality results is the Case 1, i.e. the test in which all the islands are configured with the same values. When all the islands operate with the same diversification factors, the search is done in a better way. By contrast, when each island has a particular mutation and crossover probabilities, the results are not the best that can be achieved by the model. In second place, we compare cases 1 and 4. The case 4 obtains a significant reduction of the execution time. We recall that this case used the exponential crossover.

This type of crossover inherits the parameters of trial vector from the corresponding mutant vector, starting from a randomly chosen parameter index, until the $j$th parameter value satisfying $rand_j > Cr$. It is clear from this crossover type that when the condition $rand_j > Cr$ is met, the crossover iteration stops, so -in general- for each individual of the population, this action is less time consuming than the binomial crossover used in the rest of the experiments, in which all the vector components are involved. Frequently, this particularity leads to lower execution times in the overall process, but the quality of the solutions achieved is not the optimal. This can be one of the reasons because this crossover type is less used than the binomial one. But in some circumstances, it can be desirable to achieve less execution time relegating in some orders of magnitude the solutions quality. In such cases, the use of the exponential crossover can achieve that result. Then, in general terms, for these particular problems, setting both probabilities to constant values at model level and the crossover as the binomial one leads to better quality results.

## 5 Conclusions

In this paper we describe the Island Model used to obtain a parallel version for the Differential Evolution algorithm. Different experimental tests were carried out with the aim of analyse the behaviour of the model when each island is configured with different parameters. Our interest was on the crossover type and on the crossover and mutation probabilities, applied to solve the Shifted Sphere and Shifted Rosenbrock optimization problems. When using the exponential crossover type, the quality of the obtained solutions was not optimal. However this experiment achieved a significant reduction in execution time, because the crossover type characteristics. For this reason, the use of the exponential crossover may be useful when what is desired is a reduction in the execution time, relegating in some order of magnitude the solutions quality. Through the results analysis from the test cases made on the mutation and crossover prob-

abilities, it was found that the same configuration in all islands achieves better quality in the solutions. For the functions involved in the experiments, it was found that if all islands have the same diversification factors, the search leads to better quality of solutions.

This information is a preliminary experimental basis for other type of static and dynamic calibration experiments, in order to develop a self-adaptable environment for solving hard optimization problems.

## References


1. Alba, E., Tomassini M.: Parallelism and Evolutionary Algorithms. In: Proc. of the IEEE Trans. on Evol. Comp., vol. 6, num. 5, pp. 443-462 (2002)
2. Ali, M., Pant, M., Nagar, A.: Two local Search Strategies for Differential Evolution. In: Conf. on Bio-Inspired Computing: Theories and Appl., pp 1429-1435 (2010)
3. Brest, J., Zamuda, A., Bokovie, B., Maucec M., Zumer, V.: High-Dimensional Real-Parameter Optimization using Self-Adaptive Differential Evolution Algorithm with Population Size Reduction. In: IEEE Congr. on Evol. Comp., pp 2032-2039 (2008)
4. Martínez, C., Rodriguez, F., Lozano, M.: Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. In: Soft Computing, vol. 15, issue 11, pp. 2109-2126 (2011)
5. Kozlov, K., Samsonov A.: New Migration Scheme for Parallel Differential Evolution. In: Proc. Int. Conf. on Bioinf. of Genome Reg. and Structure, pp 141-144 (2006)
6. Mattson T., Sanders B., Massingill B.: Patterns for Parallel Programming. Addison-Wesley, chapter 5, pp 143-152 (2004)
7. MPICH Message Passing Interface, http://www.mpich.org/
8. Noman, N., Iba, H.: Accelerating Differential Evolution Using an Adaptive Local Search. In: Proc. of the IEEE Trans. on Evol. Comp., vol. 12, pp 107-125 (2008)
9. Price, K., Stron R., Lampinen J.: Differential Evolution: A Practical Approach to Global Optimization. Springer. New York (2005)
10. Talbi, E.: Metaheuristics: From Design to Implementation. John Wiley & Sons, Hoboken, New Jersey (2009)
11. Tasoulis, D., Pavlidis, N., Plagianakos, V., Vrahatis, M.: Parallel Differential Evolution. In: Proc. of the Congr. Evol. Comp., vol. 2, pp. 2023-2029 (2004)
12. Zhao, S., Suganthan, P., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. In: Soft Computing - A Fusion of Foundations, Methodologies and App., vol. 15, num. 11, pp. 2175-2185 (2010)
13. Tang, K., Yao, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M., Yang, Z.: Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization. Technical Report. In: Nature Inspired Computation and Applications Laboratory. USTC. China. pp. 4-31 (2007)
14. Yang, Z., Tang, K., Yao, X.: Self-adaptive Differential Evolution with Neighborhood Search. In: Proc. of the IEEE Congr. on Evol. Comp., pp. 1110-1116 (2008)
15. Yang, Z., Tang, K., Yao, X.: Scalability of generalized adaptive differential evolution for large-scale continuous optimization. In: Soft Computing, vol. 15, issue 11, pp. 2141-2155 (2011)
16. Zaharie, D., Petcu, D.: Adaptive Pareto Differential Evolution and Its Parallelization. In: Proc. of the 5th Int. Conf. Parallel Processing and Applied Mathematics, vol. 3019, pp. 261-268 (2004)