Parameter Tuning of a Parallel Hierarchical Island Based Model*

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. One of the major drawbacks of using Evolutionary Algorithms is the determination of the input parameters, since they have an important influence in the effectiveness of the search. When using Parallel Evolutionary Algorithms, such drawbacks are magnified, since they incorporates new parameters needed to configure the inherent characteristics of the parallel model. Achieving an adequate configuration can mean an optimization problem itself. This research group has developed a parallel distributed model characterized by a hierarchy of processes communication organized in islands that cooperate in the search process. In this work we present a study of calibration for some input parameters that determines good results quality, with the aim of tuning them taking into account different configurations applied globally to the model, or locally to each island.

1 Introduction

During the 80' years computer scientists were inspired by the theory of evolution proposed by Charles Darwin, developing algorithms that attempt to simulate the species creation and evolution. These methods, called Evolutionary Algorithms (EAs), are based on the evolution of sets individuals, also known as populations, and on the notion of *competition*, since only those individuals with the most favourable variations will survive [6]. Considering a sample of the search space (the population of individuals) the main characteristic of this methods is to guide the search towards an acceptable individual (or the best one) allowing for a considerable reduction of the search time. At present, there is a broad spectrum of EAs including the well known *Genetic Algorithms* (GA) proposed in 1975, *Ant*

* This work has been supported by UTN under project EIUTNME0002170, and by ANPCyT under project PRH PICT-2008-00242.

Colonies Optimization (ACO) proposed in 1992 and *Differential Evolution* (DE) developed in 1995, between others, and hybrid methods formed by a combination of two or more algorithms, including other optimization methods.

In their conception, the EAs where formulated using a sequential processing scheme. However, these methods are naturally prone to parallelism, since most variation operations can be undertaken in parallel [1]. Furthermore, the optimization problems become even more complex, and their resource requirements are also increased. The parallel nature of EAs and the need for more powerful calculation capacity led researchers to the development of techniques that involve high performance computing tools with the aim of reducing the execution time or exploiting the search capabilities of the algorithms, giving rise to the Parallel Evolutionary Algorithms (PEAs).

The effectiveness and efficiency of the search process has a direct relation with the values for the parameters needed by the algorithm [6]. These values are provided by the user, and their establishment is not a trivial task. In some cases, they can be taken from literature, in other cases from some previous experimentation under similar scenarios may allow for adopting those known values. When the results quality is not the optimal, a calibration for the input parameters may lead to much better results. But when a PEA is used, in addition to provide the input values for the classical EA, it is necessary to consider those parameters inherent to the parallel model. In this sense, the PEAs may require greater effort for the establishment of the initial configuration.

In this work we present an off-line parameter tuning of some crucial input parameters for a parallel distributed model, called Hierarchical Island Based Model for Differential Evolution (HIBM-DE). It is based on islands, it is characterized by a double level of information exchange and for the usage of the Differential Evolution algorithm as EA. The processes involved are associated with a certain level of cooperation, allowing them to explore in a more comprehensive way the search space of the problem and favouring the maintenance of the population diversity. With this work, we attempt to determine how is the influence of the input parameters when de configuration is applied globally to the islands or locally inside each island. This work also constitutes a first step from the study of different strategies for automatic tuning techniques.

This paper is organized as follows: section 2 describes the main characteristics of the EAs, and give some details of the processing scheme of the Differential Evolution algorithm. Section 3 presents the parallel model used in this work. Section 4 shows the experiments carried out and the analysis of results. Finally, we present the conclusions and future work.

2 Evolutionary Algorithms

The Evolutionary Algorithms are stochastic population based metaheuristics developed on the basis of the evolutionary theories that describes the creation and evolution of species [6]. They are iterative algorithms that start with an initial population of individuals, which is a sample of the search space. Every individ-

ual represent a possible solution of the problem to be addressed, and belongs to a generation or epoch. An objective function associates a fitness value with each individual, indicating its worth. In each generation, a new population will be obtained based on the current population: the individuals are selected from the current population following a selection criterion, and are reproduced using variation operators (eg. mutation, crossover). Finally, some individuals from the current population are replaced by the newly ones generated, following some replacement criteria. This new individuals will be part of the new population, that will be part of the next generation.

Algorithm 1: Template of an Evolutionary Algorithm

```
Generate(P(0)); /* Initial population */
t=0;
while not Termination_Criterion(P(t)) do
    Evaluate(P(t));
    P'(t)= Selection(P(t));
    P'(t)= Reproduction(P'(t));
    Evaluate(P'(t));
    P(t+1)= Replace(P(t), P'(t));
    t=t+1;
endwhile
Output: Best individual or best population found.
```

Algorithm 1 illustrates the template for an Evolutionary Algorithm. Usually the initial population is generated with random values. The selection operator embodies the competition process, given that it responds to the principle "the better is an individual, the higher is its chance of being parent" [6]. This drive the population to better solutions. An individual is better than other if its evaluation under the objective function has less or equal value than the other individual (for minimization problems). However, worst individuals may contribute with their genetic material and have some chance to be selected. The reproduction phase is characterized by generation of new individuals through the application of two operators, such as the mutation operator and the crossover operator. The mutation consist in the generation of small changes of the selected individuals, this represents a movement in the landscape of the problem under consideration. The role of recombination or crossover operator is to generate new descendant individuals (offspring) which inherit some characteristics from the two parents. A crossover rate (Cr) controls the portion of the inherited characteristics from each parent. The termination criterion of the algorithm can be reaching a certain value (mean error, mean standard deviation, etc.), reaching a certain number of function evaluations (NFE) or reaching a predefined number of generations.

There are different approaches to parallelize the EAs [6, 5]. In general, they constitute different instances of the *master/worker* model [3] with an operator of cooperation between the workers called *migration*. The master process initializes the population and can distribute portions of it between the workers, or can retain the whole population and only send to the workers an individual

to be evaluated with the fitness function. The idea of *island* arises with these concepts, an island is considered to be a logical entity composed by individuals that are evolved by a process or a group of processes. In the *migration phase*, some individuals may move from one island to another target island in order to integrate the destination population. An algorithm that involves islands and migration, is said to be an algorithm that follows the *Island Model*.

2.1 Differential Evolution

The EA considered in this work is Differential Evolution [5], a population based metaheuristic that starts with an initial solution, usually randomly generated within the whole search space, which is evolved using three main operator: *mutation*, *crossover* and *selection*. Each individual is represented as a D-dimensional vector and belongs to a generation g.

The *mutation* operator generates new individuals for each target vector using the formula $V_{i,g+1} = X_{best,g} + (X_{r_1,g} - X_{r_2,g})F$. The constant F 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. There are other mutation strategies, but they are out of scope of this work. After the mutation phase, each perturbed individual $V_{i,g+1}$ and the individual $X_{i,g}$ are involved in the *crossover* operation, generating a new vector $U_{i,g+1}$, denominated "trial vector". A *crossover factor* $Cr \in (0, 1)$ and is used to control the values fraction that are copied from the mutant vector V. The *selection* 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.

Researchers have proposed different approaches to parallelize DE, depending on the purpose to be achieved. In [11] 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 [8], 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 [2], a parallel DE version is proposed and applied to solve biological problems. It also follows a ring interconnection topology. The analysis was carried out with different migration rates and they conclude by identifying the best of them.

3 Island Based Model for Differential Evolution

Following, we will describe the *Hierarchical Island Based Parallel Model for Differential Evolution*. It is based in two different parallel alternatives for Differential Evolution, called *Classic Island Based Differential Evolution* (CIBM-DE) and *Subpopulation Island Based Differential Evolution* (SIBM-DE) [7].

In CIBM-DE multiple islands cooperate in the search process. Each island is composed by a worker who initializes and evolves its own population. This model promotes a broader coverage of the problem under consideration since each island is initialized with a different seed, but the execution time is similar or even higher to the sequential version, considering each single island having a population configuration with the same size as the sequential version. The migration phase in this model is characterized by sending some individuals from an island to another neighbour island.

In SIBM-DE a master process initializes the population and distributes some individuals between the islands. In each island a worker process evolve its subpopulation, and communicates with other islands in the migration phase to send some individuals. Is clear that this model promotes a reduction of the execution time, since the initial population is partitioned into 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 reduction of the execution time, it has been found that it does not achieve a good quality of results compared with the sequential version or with the CIBM-DE model, and the results quality get worse as the subpopulation size gets smaller.

As can be seen, each model has a particular characteristic making it appropriate to an specific scenario. In order to achieve a balance between those characteristics, we have proposed HIBM-DE, trying to combine the benefits of each model mentioned above. The Hierarchical Island Based Model (HIBM-DE) provides two levels of parallelism. One level can enhance the computational speed (SIBM-DE), whilst the other can lead to a broader coverage of the problem domain to be addressed (CIBM-DE).

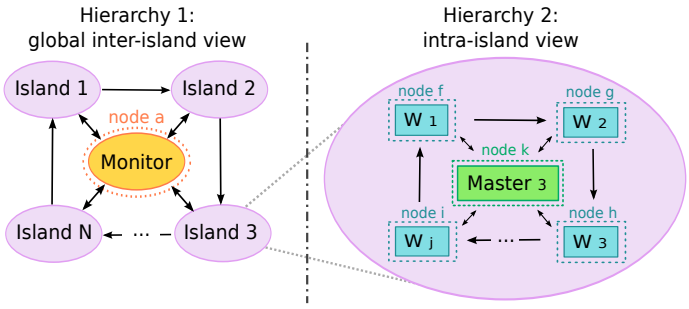

Fig. 1: **HIBM-DE:** Two hierarchy levels.

Figure 1 describes HIMB-DE. As can be seen, it comprises two levels of hierarchy. The hierarchy 1 is composed by a global inter-island view. The processes are organized in islands, they are interconnected following a ring topology. A monitor process, located into a separate computing node, is in charge of coordinating the islands actions. Each island evolve a unique population, each one initialized with a different seed. Between successive pair of islands, and after certain number of generations, a inter-island migration phase is launched. Each island sends some individuals to its corresponding neighbour island, in the

topological order. This action promotes diversity of the population, because the individuals migrated are integrated in the destination populations.

Inside each island the model follows a local intra-island view, or hierarchy 2. A master process is the responsible of coordinating the workers actions, which are connected to each other also following a ring topology and each one is located into a different computing node. The evolutionary process begins in the masters, which initializes the population and distributes it between the workers. They evolve their subpopulations and periodically communicate each other to share some individuals. This intra-island migration phase promotes a local diversity inside an island. The amount of individuals sent by the workers is a certain percentage of the population, calculated proportionally to the number of workers in the island. The replacement policy is semi-elitist, since the individuals to be sent are the best of the population an the rest is completed by means of a random selection. When the worker receives the individuals, it integrates them into its subpopulation replacing the worst ones (but only if the fitness value of the received individuals is better than the worst individuals).

It is important to remark that the migration phase of the hierarchy 1 and 2 are overlapped one from another. Once the workers of an island send to the master the individuals to be migrated, they continue evolving their subpopulations until some posterior generation in which they expect to receive individuals that provides from another islands. While the evolutionary process in the workers continues, the masters take some time to collect all the individuals from the workers, to send them to the next island, and to receive the new individuals to be integrated into each worker subpopulation. In this way, the inter-island migration is a non blocking operation for the workers and they remain active evolving their subpopulations, and minimizing idle periods.

The model comprises the characteristics of CIBM-DE, since each island has a whole population initialized with a different seed. But instead of having a unique worker responsible for evolving the population (such as in CIBM-DE), 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, HIBM-DE integrates both parallel models, and provides a flexible configuration in which the islands can be composed by the amount of workers needed to achieve the time reduction desired, and the user may instantiate the number of islands needed to also achieve the results quality desired. The models CIBM, SIBM and HIBM can be applied to other EAs, in this work we focus on the Differential Evolution algorithm as the main EA for them.

4 Test cases and analysis of results

In the following, we describe the experiments carried out in order to test HIBM-DE with different configurations. The performance of the model was tested with a set of scalable functions, obtained from [9]. For each of them, 30 executions were carried out with different seeds. The size of the problems considered have dimension 1000, the population was made up with 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. In the codification we use the MPICH library [4] for message passing communication between participating nodes. All tests were made on a cluster with 32 CPUs distributed between 8 nodes Intel Q9550 Quad Core (Debian 5 Lenny, Ethernet and switch Linksys SLM2048 of 1Gb).

The experiments carried out in this work are within the *off-line* parameter tuning. In this type of calibration the initialization of the values of different parameters are fixed before the execution of the algorithm [6]. We propose the calibration of the Crossover probability (Cr) and the Mutation factor (F), since they are crucial in the evolutionary process, because they determine the characteristics of the individuals in each generation. Our intention is to determine the influence of these parameters for HIBM-DE when the calibration is applied globally to each island or locally inside the islands. The first case, is related with the first hierarchy level, since each island of the model will behave in a unique way exploring the search space. In the second case, the hierarchy two is affected because each worker of an island will operate in a different way, with their own parameter values. Following, we provide a brief description of the test cases. For each of them we considered both types of configuration: the global configuration that affects the hierarchy 1 (we consider a coarse grain, and each entity is composed by an island), and the local or intra-island configuration that affects the hierarchy 2 (we consider a fine grain, and each entity is composed by a worker):

- **Case 1:** All the islands have the same configuration, Cr was setted to 0.3 and F to 0.5. These initial values were taken from literature. They are well known for their effectiveness with functions similar to those used in this work [10]. This case will obtain the same results both with a global or local configuration.

- **Case 2:** We carried out a previous off-line calibration of Cr and F. We conclude on executing this experiment using Cr=0.45 and F=0.5 as the best values found for these parameters and we apply them to all the islands. As in case 1 the results obtained with a global or local configuration will be the same.

- **Case 3:** This experiment consisted in setting the half of the entities considered with the best values for the mutation and crossover probabilities (Cr=0.45, F=0.5), and the other half of the entities used random values for Cr and F, but considering a range within ± 0.2 points from each one. Then, Cr uses random values from the interval $[0.25, 0.65]$ and F was in the range $[0.3, 0.7]$.

- **Case 4:** In this experiment all the entities used random values for Cr and F in the range within ± 0.2 points from each one ($Cr \in [0.25, 0.65]$ and $F \in [0.3, 0.7]$).

- **Case 5:** In this experiment only the value of Cr was altered. All the entities used F=0.5 but half of them used random values for Cr within ± 0.2 points from each one ($Cr \in [0.25, 0.65]$).

- **Case 6:** In this experiment, as in the previous case, only the value of Cr was altered, the entities used F=0.5 and random values only for $Cr \in [0.25, 0.65]$.

Summarizing, in the first two cases all the islands and workers have the same values for Cr and F. Case 3 and 4 alters the values for Cr and F, affecting the hierarchy 1 and 2 in a half or in the total of islands, respectively. Cases 5 and 6 only alters the value of Cr, affecting the hierarchy 1 and 2 in a half or in the total of islands, also respectively.

Figures 2 and 3 shows the obtained results. In each graph the mean error obtained with different configurations of islands and workers per island is shown. The notation HIBM-DE-X_Y means that the experiment was carried out using X number of islands, each one composed by Y workers. In the test we use a configuration of two and four islands, and two or four workers per island.

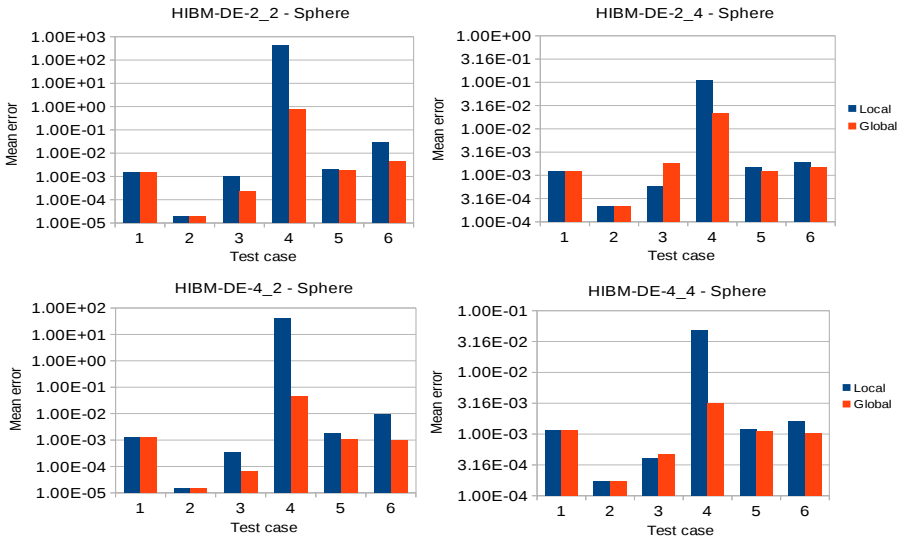

Fig. 2: HIBM-DE 2-2, HIBM-DE 2-4, HIBM-DE 4-2 and HIBM-DE 4-4 for Sphere.

As can be seen from figures, the cases 1 and 2 obtains the same mean error for a local or global configuration, since all the islands and workers use the fixed values. Also, better values are found in case 2, this is consequent with the preliminary calibration made on the parameters. Comparing the results of cases 3 and 4, especially in the test cases of Sphere function, it can be noticed that using random values for the parameters is not the best option. The worst of the six cases was the case 4 using a local configuration. In the rest of the cases the best results are generally found using a global configuration, i.e. when the hierarchy 1 is affected. This shows that the model explores in a more comprehensive way the search space of the problem when all the islands operate applying the mutation and crossover operator working as a uniform entity.

Comparing the results respect to the number of islands, it can be seen that when the number of islands is increased the results are better for each test case. For example, the results obtained with Sphere using HIBM-DE-4_2 are better

than using HIBM-DE-2_2. This is a characteristic of the model, when the number of islands is duplicated, the search space of the problem is augmented, since each island is initialized by a different seed.

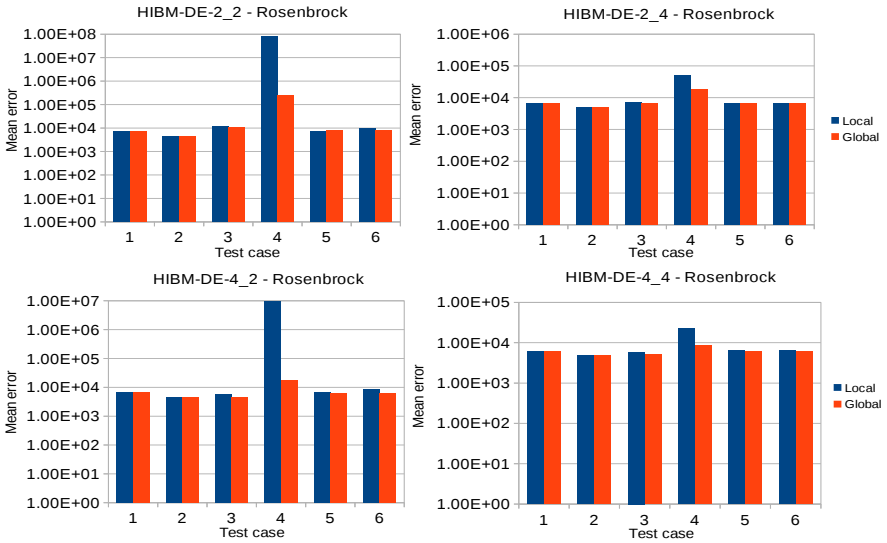

Fig. 3: HIBM-DE 2-2, HIBM-DE 2-4, HIBM-DE 4-2 and HIBM-DE 4-4 for Rosenbrock.

Table 1 shows the mean time for each test case. As it is expected, independently from a global or local configuration, the execution time for the same amount of islands and workers per islands obtains similar execution times.

Table 1: Average runtime for all the test cases (in seconds).

		HIBM-DE 2-2		HIBM-DE 2-4		HIBM-DE 4-2		HIBM-DE 4-4	
		intra	global	intra	global	intra	global	intra	global
Sphere	case 1	63.34s	63.16s	28.86s	32.86s	61.64s	63.11s	33.44s	32.73s
	case 2	64.59s	64.74s	33.85s	33.79s	64.22s	63.87s	35.56s	35.13s
	case 3	65.65s	68.84s	32.63s	33.02s	62.57s	63.38s	34.53s	34.36s
	case 4	63.01s	62.64s	32.83s	32.72s	63.96s	61.72s	34.47s	34.13s
	case 5	61.05s	60.27s	30.72s	30.26s	62.06s	62.34s	32.19s	31.90s
	case 6	62.91s	62.06s	31.02s	30.87s	63.23s	63.36s	32.79s	31.71s
Rosenbrock	case 1	73.34s	72.40s	38.41s	37.78s	71.02s	73.63s	39.03s	38.67s
	case 2	74.30s	74.04s	38.97s	38.90s	72.98s	73.02s	40.44s	39.93s
	case 3	72.98s	72.84s	38.10s	37.52s	73.21s	73.24s	38.92s	39.36s
	case 4	72.06s	73.22s	38.12s	37.71s	72.93s	70.98s	38.71s	39.03s
	case 5	69.60s	69.98s	35.12s	35.07s	71.79s	71.90s	37.41s	37.19s
	case 6	77.72s	73.06s	36.03s	36.03s	72.89s	72.60s	37.98s	36.88s

It can be observed that HIBM-DE-2_2 obtains for both local or global configuration, execution times about 60 seconds for Sphere and 70 seconds for Rosenbrock. As a characteristic of the model, when the number of workers per island is duplicated, the execution time is reduced in approximately the half. HIBM-DE-2_4 obtains means execution times of about 30 s. for Sphere and 37 s. for Rosenbrock.

5 Conclusions

In this paper we have described an off-line parameter tuning of two crucial parameters for the Differential Evolution algorithm. The parameters under consideration for the calibration were the crossover probability (Cr) and the mutation factor (F), since they are crucial for the population determination in each generation. This work is focussed in HIBM-DE, a parallel model characterized by two levels of hierarchy. Our main goal is to determine the behaviour of the model when the parameters are configured with different values, applied locally inside each island or globally to each island. We have proposed six test cases, and on its hand, each one is unfolded in two different configurations related with the two hierarchies. From the results analysis it was found that the model explores in a more comprehensive way the search space of the problem when a global configuration is used, that is, setting all the islands with the specific input value. Also, the results showed that the execution time of the test cases are similar, independently of a global or local configuration, but when the number of workers per island is duplicated, the execution time is reduced in approximately a half. This experimentation is a preliminary study for other type of static and dynamic calibration experiments, in order to develop a self-adaptable parallel environment for solving hard optimization problems.

References

1. Alba, E., Tomassini M.: Parallelism and Evolutionary Algorithms. Proc. of the IEEE Trans. on Evol. Comp. **6 (5)** (2002) 443–462
2. Kozlov, K., Samsonov A.: New Migration Scheme for Parallel Differential Evolution. Proc. Int. Conf. on Bioinf. of Genome Reg. and Structure. **2** (2006) 141-144
3. Mattson T., Sanders B., Massingill B.: Patterns for Parallel Programming. Addison-Wesley. **5** (2004) 143-152
4. MPICH Message Passing Interface, http://www.mpich.org/
5. Price, K., Stron R., Lampinen J.: Differential Evolution: A Practical Approach to Global Optimization. Springer. New York (2005)
6. Talbi, E.: Metaheuristics: From Design to Implementation. John Wiley & Sons, Hoboken, New Jersey (2009)
7. M.L. Tardivo, P. Caymes-Scutari, M. Méndez-Garabetti, G. Bianchini, Two Models for Parallel Differential Evolution. Proc. of the High Performance Computing Latin American Symposium, Mendoza, Argentina. (2013) 26-36
8. Tasoulis, D., Pavlidis, N., Plagianakos, V., Vrahatis, M.: Parallel Differential Evolution. Proc. of the Congr. Evol. Comp. **2** (2004) 2023-2029
9. 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. Nature Inspired Comp. and Applications Lab. USTC. China. (2007) 4-31
10. Yang, Z., Tang, K., Yao, X.: Self-adaptive Differential Evolution with Neighborhood Search. Proc. of the IEEE Congr. on Evol. Comp. (2008) 1110-1116
11. Zaharie, D., Petcu, D.: Adaptive Pareto Differential Evolution and Its Parallelization. Proc. of the 5th Int. Conf. Parallel Processing and Applied Mathematics. **3019** (2004) 261-268