An Adaptive Hierarchical Fair Competition Genetic Algorithm for Large-Scale Numerical Optimization

Alexandre C. M. Oliveira
Federal University of Maranhão
Department of Informatic
São Luís MA, Brazil
acmo@deinf.ufma.br

Luiz A. N. Lorena
Airam J. Preto
Stephan Stephany
National Institute for Space Research
Computing and Applied Mathematics Laboratory
São José dos Campos SP, Brazil
{lorena,airam,stephan}@lac.inpe.br

Abstract

Genetic algorithms, inspired by the theory of evolution of species, are intended to be unfair. Individuals compete against each other and the best-adapted ones prevail. Unfairness is due to big differences of skills, generally evaluated by a fitness measure, in a population of individuals competing for survival. However, population diversity is important to preserve some features that are not always associated to high ranked skills. Such diversity can be achieved by imposing fairness rules to the competition. The adaptive hierarchical fair competition genetic algorithm has been proposed to comply with this feature by segregating individuals in casts or demes, according to their fitness. This work proposes a parallel implementation that enhances the capabilities and computational performance of an adaptive hierarchical fair competition genetic algorithm. The code was parallelized using the MPI (Message Passing Interface) communication library and executed in a distributed memory parallel machine, a PC cluster. Test results are shown for standard numerical optimization problems presenting hundreds of variables.

1 Introduction

Genetic Algorithms (GA) are stochastic search methods based on natural selection and reproduction of a population of individuals, that evolves along the generations. Each individual has its particular genotype and is associated to a candidate solution. A coding strategy is required to define the genotype. The evolutionary process generates new individuals using operators for selection, crossover and mutation. Particularly, real-coded GA's have been employed in numerical optimization with good results [9].

Sequential implementations of GA's have been successful for many applications in very different domains. However, very large-search spaces could possibly be better explored using Parallel GA's (PGA). In such cases, a very large population is required to sample the problem domain and processing/memory demands are heavy for a single processor, precluding the GA implementation to be efficient.

Sequential GA's may also get trapped in a suboptimal region of the search space thus being unable to find better solutions. On the other hand, PGA's can overcome such limitation by exploring in parallel different search sub-spaces. A reason for this is that multiple populations allow speciation, i.e., different populations evolve in different directions. Therefore, PGA's are not just extensions of standard sequential GA's, but constitute a new paradigm that is able to perform a selective search in the space of solutions. GA parallelization requires some decisions, such as: (a) individual evaluation or genetic operators performed or not in parallel, (b) the use of a single or multiple sub-populations (demes), (c) migration policy of individuals between processors, etc.

Some of the models described in the literature are [10]: (a) master-slave (or global) parallelization, (b) multiple demes with migration, and (c) demes with static or dynamic overlapping. In the master-slave PGA, the tasks of evaluating the individuals and applying the genetic operators are divided among slave processors. A master processor controls the evolution of the population. This is a straightforward parallel extension of standard GA's. Multiple-deme GA's are

the most popular PGA's. Depending on the number and size of the demes, they are called coarse or fine-grained models. The coarse-grained algorithms use a small number of demes with many individuals. Most of the processing time is spent in the generation of the demes and only occasionally individuals are exchanged between demes, according to a migration policy. Each deme can be associated to a GA executed in a specific processor. Migration patterns include broadcasting of individuals among processors (island model) or exchange of individuals between neighbouring processors (stepping-stones model). On the other hand, fine-grained models divide the population into a large number of smaller demes. Inter-deme communication is generally accomplished by overlapping demes. The migration policy is defined by some parameters: (a) the topology that defines the connections between demes, (b) the migration frequency and the number of individuals exchanged, and (c) the criteria for choosing the migrants. Finally, PGA's with overlapping schemes define areas obtained by the intersection of demes, following some defined topology. These areas allow the propagation of the genetic characteristics of best evaluated individuals through the demes.

A recently parallel model, the Hierarchical Fair Competition (HFC), was proposed, inspired by the stratifying nature of the society, but only a sequential version was implemented. The population is segregated in casts or demes according to individual skills [8]. These skills are evaluated by a fitness function as in standard GA's. Individuals may move from a low-fitness to high-fitness demes, according to the fitness-based admission thresholds of the demes. HFC has been successfully applied to the analog circuit synthesis problem using a genetic programming (HFC-GP) [8]. This work presents a new implementation of the hierarchical fair competition genetic algorithm applied to numerical optimization (function parameter optimization). Enhancements include parallelization, asynchronous exchange of individuals betweens casts, and different evolutionary environments for the casts. Each cast is mapped to a different processor. The code employs the Message Passing Interface (MPI) [12, 6] communication library to exchange individuals between demes. The proposed implementation was evaluated with standard numerical optimization test functions with hundreds of variables.

2 Hierarchical Fair Competition model

Convergence is a desirable feature for a GA, but must be controlled in order to avoid get trapped in local optima. It is common to have high-fitness individuals supplanting low-fitness ones or being selected more often, and thus dominating the evolutionary process. Genetic algorithms, inspired by the theory of evolution of species, are intended to be unfair. Individuals compete against each other and the best-adapted ones prevail, restricting population diversity. However, it is important to preserve sample individuals scattered in the search space in order to increase the probability of reaching the global optima in multi-modal optimization problems. Population diversity can be achieved by keeping the competition among individuals fairer.

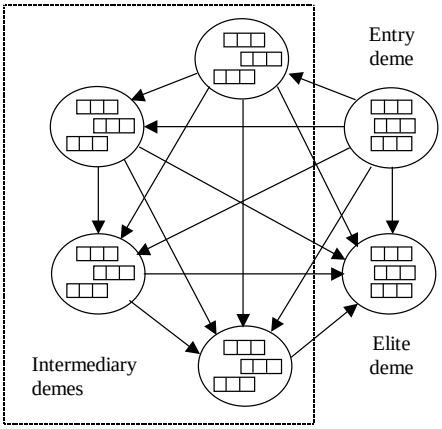

Figure 1. The Hierarchical Fair Competition model.

The Hierarchical Fair Competition (HFC) model was proposed to avoid the premature convergence in traditional evolutionary algorithms [8]. Fair competition is enforced by segregating the individuals in independent casts or classes according to their skills. Such model mimic some advanced social organizations that preclude unfair competition.

In the HFC model, multiple demes are organized in a hierarchy, in a way that each deme can only contain individuals within its specific range of fitness. Thus the fitness domain is mapped to a finite number of fitness ranges, defined by admission thresholds. Individuals are moved from low-fitness to high-fitness demes if and only if they exceed the admission threshold of upper-fitness demes. No individuals are moved to low-fitness demes. Therefore, the HFC model migration operator is unidirectional. Figure 1 illustrates the topology of the HFC model. Arrows indicate the migrations that are possible. The entry deme (primary level) may send individuals to all other demes, while the elite deme only can receive individuals, without sending any. This model can be represented by a directed graph, in which only some moves are allowed.

Concerning the migration frequency, the HFC model establish that individuals must be moved at uniform intervals, using admission buffers to collect the migrants that came from other demes. Any individuals with fitness outside (above) the fitness range of its deme must migrate to the suitable deme. The amount of individuals to be exchanged can not be predicted as it depends on the evolution of each deme. A heterogeneous evolutionary environment can be included in the HFC model. In such environment, each deme would evolve individuals employing different search strategies, subpopulation sizes, evolutionary operators, parameters, etc. Different migration patterns could also be employed [8].

The fitness ranges are determined according to the problem and to the number of levels. An adaptive scheme that employs a few generation calibration stage is employed to determine an initial range of fitness values. This stage computes the average fitness f_μ, the standard deviation σ_f, and the best fitness f_{min} of the population that are used to define the admission thresholds. Afterwards, the individuals of the initial population are classified according to their fitnesses and moved to the corresponding levels. In regular interval generations, an update stage is performed in order to recalculate the admission thresholds of the above-entry levels by evaluating again f_μ, σ_f, and f_{min}. Both the calibration and update stages are employed to deal with the lack of previous knowledge about the problem, incorporating an adaptive feature to the algorithm [8].

3 The proposed implementation

An enhanced adaptive HFC algorithm that includes parallelization is presented in this work, the Parallel Adaptive Fair Competition Algorithm (PAHFCGA). Some of the enhancements were suggested in the original proposal of the HFC model for further implementation [8] as (a) an adaptive determination of number of levels; (b) a more sophisticated admission threshold adaptation mechanism; and (c) multi-processor parallel implementation of HFC and testing on huge problems. The PAHFCGA implements (b) and (c), while presenting (d) an asynchronous exchange of individuals between levels, and (e) different sets of evolutionary parameters and operators for the demes of each level. Each deme is mapped to a different processor and individual exchanges between processors are performed through message-based communication, implemented using the Message Passing Interface (MPI) library.

It is not a trivial task to divide the fitness space into uniform ranges, as the search space may present valleys, plateaus, and peaks. In the PAHFCGA, the fitness is not equal to the value f of the objective function. Instead, a mapping is made from this value to a linear interval $[0_-, R_+]$, where R is a scaling constant, providing an even distribution of the fitness domain. The PAHFCGA starts with the calibration stage and each processor records the highest and lowest objective function values (f_U and f_L, respectively). The fitness value g_{ij} of the jth individual in ith level is:

$$g_{ij} = R\,[(f_{ij} - f_L)/(f_U - f_L)] \tag{1}$$

In minimization problems, the admission threshold AT_i of the ith level is given by:

$$\begin{aligned} AT_{[i=0]} &= +\infty \\ AT_{[i>0]} &= \overline{g} + \tfrac{(i-1)}{(D-2)}\,[\overline{g} - \overline{g}^*] \end{aligned} \tag{2}$$

where D is the number of levels (demes). $\overline{g}$ is the average of the fitness values, while $\overline{g}^*$ is the average of the above-average fitness values, defined for population size M (equal for all demes):

$$\begin{aligned} \overline{g} &= \min_i^D \left[\tfrac{1}{M}\textstyle\sum_j^M g_{ij}\right] \\ \overline{g}^* &= \min_i^D \left[\tfrac{1}{M}\textstyle\sum_j^M \max(0, \overline{g_i} - g_{ij})\right] \end{aligned} \tag{3}$$

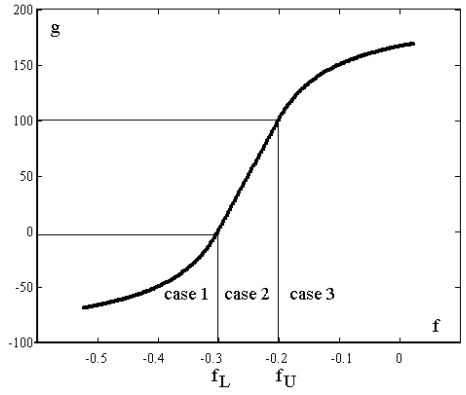

Figure 2. Mapping of $f \rightarrow g$, for $R = 100$.

During the regular evolution, individuals with objective function value out of the range $[f_U,\ f_L]$ can be generated. They must be treated as a special case of the mapping. Figure 2 depicts a mapping scheme $f \rightarrow g$ for the three possible cases that are defined by:

1) $g_{ij} = R\,[(f_L - f_{ij})/(f_U - f_{ij})]$, $\quad f_{ij} \in [-\infty, f_L]$

2) $g_{ij} = R\,[(f_{ij} - f_L)/(f_U - f_L)]$, $\quad f_{ij} \in [f_L, f_U]$

3) $g_{ij} = R\,[(1 + f_{ij} - f_U)/(f_{ij} - f_L)]$, $\quad f_{ij} \in [f_U, \infty]$

After the calibration stage, as evolution proceeds, each deme may present individuals that are classified as inferior, pertinent or superior, according to their fitness value. The entry deme has no inferior individuals nor the elite deme has superior ones. In former HFC implementations, superior individuals were kept in their demes of origin until the exchange time, participating of the deme evolutionary process. Exchange of individuals between demes was performed in a synchronous manner, after a fixed number of generations. In the PAHFCGA, individual exchange is fully asynchronous and superior individuals are put in an output buffer, being therefore excluded from the demes, not affecting any further generations.

When the output buffer of any deme is full, or a certain number of generations is reached, the evolutionary process of that deme is stopped and all superior individuals are exported, emptying the buffer. Afterwards, incoming individuals from lower levels are received in the admission buffer, if they exist. These individuals are included in the deme, taking the place of less-fitted individuals (steady-state-like updating strategy) [11]. Then, the algorithm is continued by running the next generation.

PAHFCGA employs standard genetic operators as the roulette wheel selection [5], blend crossover [4], and non-uniform mutation [9]. A heterogeneous evolutionary environment is provided by taking distinct values of three evolutionary parameters in each level: the mutation rate MR, the blend crossover parameter BLX_α, that is associated to the diversification of the offsprings, and the local search probability L_SP. Considering $0 \leq MR \leq 0.5\%$ and $0.10 \leq BLX_\alpha \leq 0.25$, discrete values of these parameters are distributed linearly among the demes, the upper bound corresponding to the elite deme and the lower bound to the entry deme. The local search operator, in this case, the Hooke and Jeeves operator [7], is used only in the elite deme. The PAHFCGA pseudo-code is:

```
1: for all demes do
2:   InitializePopulation(P);
3:   CalibrationStage calculating f_U and f_L;
4:   GetLocalParameters (MR, L_SP, BLX_α);
5:   repeat
6:     if (updatetime) then
7:       ComputeLocal (ḡ, ḡ*);
8:       ComputeGloal (ḡ, ḡ*, AT);
9:       Send (AT, AllDemes);
10:    end if
11:    EvolveDeme(P,MaxIterations);
12:    while (output buffer is not empty) do
13:      Send (individual to respective deme);
14:    end while
15:    if (entry deme) then
16:      ComplementPopulation(P);
17:    else
18:      while (there are incoming individuals) do
19:        Receive (Individual from anywhere);
20:        UpdatePopulation(P, Individual);
21:      end while
22:    end if
23:  until (stop criteria)
24: end for
```

The procedure ComplementPopulation(P) is run only in the entry level, in order to replace the exported individuals. It can be said that this level acts as a generator of individuals, feeding continuously all demes with new genetic material [8]. The stop criteria is given by a specific maximum number of objective function calls or by the quality of the best solution found.

4 Test results

Many applications related to numerical optimization are very suitable for the use of evolutionary algorithms. Such applications include neural network training, fuzzy set optimization, and inverse problems. It is a common practice to test numerical optimization algorithms with "benchmark" test functions found in literature. In this work, the well-known test functions rosenbrock (Ros), schwefel (Sch), griewank (Gri), and rastringin (Ras) are used. These functions can be set with large number n of variables and are described in [2, 3]. In particular, the PAHFCGA was tested with $200 \leq n \leq 700$, to check its effectiveness for large-scale numerical optimization. The experiments were run on a distributed memory parallel machine, a cluster with 15 nodes based on the AMD (1.67 GHz) platform and a Fast Ethernet network.

Tests intended (a) to prove the effectiveness of the proposed algorithm for large-scale problems, (b) to test the new features of the algorithm, described in the preceding section, (c) to evaluate the influence of the number of demes/processors in the performance and quality of results, and (d) to test different values of the evolutionary parameters and of the local search probability.

Some parameters were set based on [8]: deme size and buffer sizes were set to 100, the calibration stage was run with 10 generations and the update stage procedure was performed at every 2000 generations. Each trial was finished if 20×10^6 objective function calls were reached by any of processors (generally, the entry deme due to its continuous individual generation). Each experiment consists of 10 trials. SN denotes the number of successful trials. FS is the average best solu-

tion, and ET is the average maximum execution time of the successful trials. Each trial is considered successful if a 0.001 solution is reached.

Table 1 shows the results obtained by the PAHFCGA for the test functions Ros($n = 500$), Sch($n = 200$), Gri($n = 200/700$), and Ras($n = 200$). The local search probability L_SP was take in the range $0\% - 0.5\%$. For $L_SP = 0\%$ (without local search), no satisfactory solutions were obtained for Ros($n > 100$) and Gri($n > 200$). Thus, for $L_SP = 0\%$, only results are showed for Gri, Ras and Sch (both with $n = 200$). For $L_SP = 0.1\%$ and $L_SP = 0.5\%$, the expected solution was found for Gri($n = 700$) and Ros($n = 500$). The experiments were performed with $D = \{6, 9, 12, 15\}$ processors.

It is difficult to select an optimal number of processors/demes D, as performance does not scale linearly with D. This is due to the evolutionary behaviour of multiple demes. The same consideration applies to the local search probability. This operator could not improve the performance in the cases Ras($n = 200$) and Sch($n = 200$). However, for Gri($n = 200/700$), one can observe that the best results were obtained using 15 demes. Figure 3 shows the ET obtained in the test cases for the different test functions, number of variables, local search probability and number of processors.

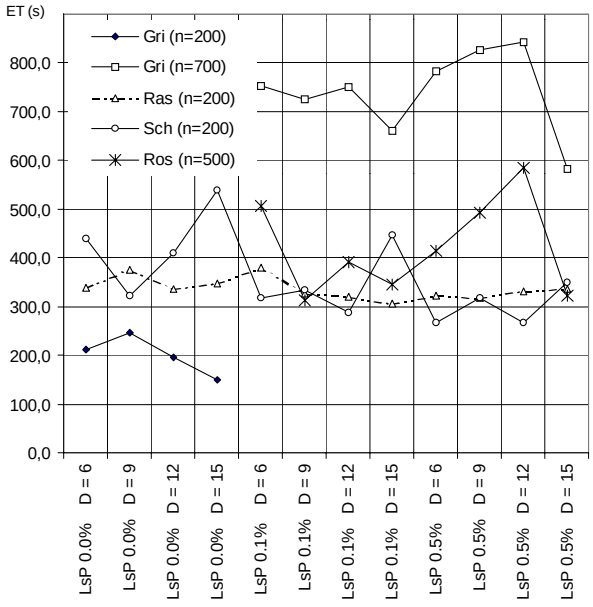

Figure 3. Execution times (ET).

A gain of performance it is always expected when parallel algorithms are used. However, it is not possible to predict the behaviour of multiple demes evolving in separate processors. The PAHFCGA was compared with a sequential real-coded genetic algorithm (SRCGA) that is very similar to the EvolveDemes procedure, i.e., using the same evolutionary and local search operators. The same stop criteria was applied and 10-trial averages were taken. Results of the SRCGA are compared to those of the PAFCGA for $n = 200$ in Table 2. In this table, PAFCGA results correspond to the case $L_SP = 0\%$ and $D = 12$. The SRCGA did not solve the Rosenbrock function for $n > 100$, neither the Gri($n = 700$) case.

5 Final Remarks

A new implementation of the Hierarchical Fair Competition GA has been presented, the Parallel Adaptive Hierarchical Fair Competition GA (PAHFCGA). The proposed implementation has enhancements such as parallelization, asynchronous exchange of individuals betweens casts, and fully stratified casts, i.e., each cast has its own set of evolutionary parameters. These new features provided a higher fidelity to the original model. A local search operator was also included in the elite cast. Each cast is mapped to a different processor according to a distributed memory model. Individual exchanges between casts are performed through message-based communication between processors, implemented using the Message Passing Interface (MPI) communication library.

The PAHFCGA evaluation was performed using standard numerical optimization test functions with hundreds of variables. Results were compared with a sequential GA that has the same evolutionary and local search operators. The PAHFCGA presented significant improvements in the quality of solutions and computation performance. It is difficult to define an optimal number of processors for all the tested benchmark functions. The same applies concerning the suitability of the use of the local search operator.

The HFC model does not divide into processors the search space, but the fitness space. Fitness does not always reflect the entire complexity of the search space. It is possible that certain fitness ranges become overcrowded of search space points, causing an unbalanced migration. Even though, the use of such model in the PAHFCGA reduced the execution time in the large-scale numerical optimization problems that were tested. Results obtained so far are very promising. Large-scale optimization is commonly found in areas such as inverse design or neural network training, thus presenting a good potential for the use of the PAHFCGA. In a further step, it is intended to use this algorithm in an inverse problem related to aircraft structural damage detection [1].

Table 1. PAFCGA **results for different evolutionary parameters.**

f	n	FS	SN	ET(s)	FS	SN	ET(s)	FS	SN	ET(s)	FS	SN	ET(s)
		L_SP 0%		D = 6	L_SP 0%		D = 9	L_SP 0%		D = 12	L_SP 0%		D = 15
Gri	200	0.0010	8	212.1	0.0034	6	246.5	0.0032	9	196.3	0.0010	10	149.8
Ras	200	0.0009	8	338.8	0.0016	8	375.8	0.0015	8	335.3	0.0021	4	348.7
Sch	200	0.0010	10	439.4	0.0009	10	323.0	0.0010	10	409.0	0.0010	9	537.6
		L_SP 0.1%		D = 6	L_SP 0.1%		D = 9	L_SP 0.1%		D = 12	L_SP 0.1%		D = 15
Gri	700	0.0003	10	753.1	0.0005	10	725.8	0.0002	8	751.1	0.0001	10	660.4
Ras	200	0.0026	8	379.8	0.0034	6	327.2	0.0023	6	319.1	0.0013	5	305.5
Ros	500	0.0007	10	506.7	0.0005	8	312.9	0.0007	10	392.1	0.0004	10	344.9
Sch	200	0.0009	10	318.5	0.0010	8	334.2	0.0009	8	288.2	0.0009	10	445.7
		L_SP 0.5%		D = 6	L_SP 0.5%		D = 9	L_SP 0.5%		D = 12	L_SP 0.5%		D = 15
Gri	700	0.0003	10	781.8	0.0002	10	825.4	0.0002	10	842.1	0.0001	10	583.3
Ras	200	0.0022	8	322.4	0.0023	6	317.3	0.0029	8	330.6	0.0026	8	336.8
Ros	500	0.0006	10	415.2	0.0009	10	491.6	0.0008	10	585.0	0.0003	8	322.3
Sch	200	0.0010	10	266.7	0.0011	8	318.5	0.0010	8	267.9	0.0018	8	348.8

Table 2. SRCGA × PAFCGA **for** $n = 200$**.**

f	FS	SN	ET(s)	FS	SN	ET(s)
Gri	0.0183	4	566.4	0.0032	9	196.3
Ras	0.0130	2	480.6	0.0015	8	335.3
Sch	0.0520	3	529.5	0.0010	10	409.0

6. Acknowledgements

The authors acknowledge FAPESP for the support received in the research project "Física dos Materiais num Ambiente de Memória Distribuída" (proc. 01/03100-9) and CNPq for the financial support (proc. 300837/89-5).

References

[1] L. Chiwiacowsky, H. Campos Velho, and P. Gasbarri. The damage identification problem: a hybrid approach. In Proceedings, pages 1393–1402, São José dos Campos (SP), 18-22 August 2003. Thematic Congress on Dynamics and Control (DINCON 2003), 2, São José dos Campos (SP), Brazil, DINCON.

[2] K. A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

[3] J. Digalakis and K. Margaritis. An experimental study of benchmarking functions for genetic algorithms. International Journal of Computer Mathematics, 79(4):403–416, 2002.

[4] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval schemata. In L. D. Whitley, editor, Proceedings, volume 2, pages 187–202, San Mateo, CA, 1993. Foundations of Genetic Algorithms, Morgan Kaufmann Publishers.

[5] D. E. Goldberg. Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[6] W. Gropp, E. Lusk, and S. A. Using MPI: portable parallel programming with the message passing interface. The MIT Press, Cambridge, USA, 2 edition, 1999.

[7] R. Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. Journal of the ACM, 8(2):212–229, 1961.

[8] J. Hu, E. D. Goodman, K. Seo, and M. Pei. Adaptive hierarchical fair competition (AHFC) model for parallel evolutionary algorithms. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, Proceedings, pages 772–779, New York, 9-13 July 2002. Genetic and Evolutionary Computation Conference (GECCO 2002), San Francisco (USA), Morgan Kaufmann Publishers.

[9] Z. Michalewicz. Genetic algorithms + data structures = evolution programs. Springer-Verlag, Berlin, 1996.

[10] M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In L. Jain, editor, Proceedings, pages 88–92, Adelaide, August 1999. International conference on knowledge-based intelligent information engineering systems (KES'99), 3, Adelaide (AU), IEEE.

[11] A. C. M. Oliveira and L. A. N. Lorena. Real-coded evolutionary approaches to unconstrained numerical optimization. In J. M. Abe and J. I. Silva Filho, editors, Advances in Logic, Artificial Intelligence and Robotics, volume 2. Congress of Logic Applied to Technology (LAPTEC2002), 3, São Paulo, Brazil, Plêiade, 2002.

[12] P. Pacheco. Parallel Programming with MPI. Morgan Kaufmann Publishers, San Francisco, USA, 1996.