A Dynamic Artificial Immune Algorithm Applied to Challenging Benchmarking Problems

Fabrício Olivetti de França,
Fernando J. Von Zuben, *Member, IEEE*

***Abstract*— In many real-world scenarios, in contrast to standard benchmark optimization problems, we may face some uncertainties regarding the objective function. One source of these uncertainties is a constantly changing environment in which the optima change their location over time. New heuristics or adaptations to already available algorithms must be conceived in order to deal with such problems. Among the desirable features that a search strategy should exhibit to deal with dynamic optimization are diversity maintenance, a memory of past solutions, and a multipopulation structure of candidate solutions. In this paper, an immune-inspired algorithm that presents these features, called dopt-aiNet, is properly adapted to deal with six newly proposed benchmark instances, and the obtained results are outlined according to the available specifications for the competition at the Congress on Evolutionary Computation 2009.**

I. Introduction

For many years optimization heuristics have been studied in order to solve general problems that can be applied in real-world [1], [2]. Most of those problems usually present several constraints that may represent a limited resource, a minimal requirement, or a range of feasible values. Additionally, some of them are associated with challenging scenarios involving noisy objective function evaluation, costly objective functions, or functions that change over time (i.e., dynamic functions).

Although most optimization techniques focus on the optimization of static, non-changing problems, many real-world optimization problems are actually dynamic, like the network routing problem, blind FIR equalization of time-varying channels [3], [4], and dynamic traveling salesman problems [5]. In dynamic optimization, not only the attributes of the optimization function may change over time, but there may also be certain variants of the actual goal of the optimization problem, including a redefinition of the domain of variables [6]. To efficiently solve a dynamic optimization problem, the search strategy needs to efficiently find high-quality solutions, but it also has to cope with the constant variation of the optimization surface, being capable of tracking those solutions in response to the displacement promoted by the varying conditions.

Fabrício Olivetti de França and Fernando J. Von Zuben are with the Laboratory of Bioinformatics and Bioinspired Computing (LBiC), Department of Computer Engineering and Industrial Automation (DCA), School of Electrical and Computer Engineering (FEEC), University of Campinas (Unicamp). Av. Albert Einstein – 400 – Building G2 – Room LE 14G, Campinas – SP – Brazil, Zip Code 13083–852, P. O. Box 6101 (e-mail: {olivetti, vonzuben}@dca.fee.unicamp.br).

As pointed out by Branke [6], the desirable features of an optimization algorithm in order to deal with such class of problems are the following:

- Diversity Creation: the algorithm must be capable of generating diverse solutions;
- Diversity Maintenance: the algorithm should avoid the convergence of every solution into one single optimum;
- Memory of solutions: some past good solutions may become good start points to find a new global optimum;
- Multipopulation: with several populations searching in parallel and obeying the above features, the algorithm becomes more robust and so it may explore the search space more efficiently.

One class of immune-inspired algorithms, based on artificial immune network principles and named aiNet family [7], already presents each one of the above features naturally incorporated within its meta-heuristic framework, thus being great candidates to deal with dynamic problems.

One of the aiNet-based algorithms created specifically for dynamic optimization is the one called dopt-aiNet [8] (Artificial Immune Network for Dynamic Optimization), that improves the original opt-aiNet [9] (created for static optimization problems) by introducing a set of complementary mutation operators and a better mechanism to maintain the diversity of solutions. In this paper, the dopt-aiNet algorithm will be adopted to deal with the benchmark problems of the competition on Dynamic Optimization to be held at the Congress of Evolutionary Computation 2009 [10]. The adaptations to be introduced to the original version of dopt-aiNet are motivated by the limited resource available to the algorithm at each time frame of the changing environment.

This paper is organized as follows. Section II presents some general conceptual aspects of Artificial Immune Systems and its dynamic optimization extension, named dopt-aiNet. The modifications made to the original version of the dopt-aiNet algorithm is presented in Section III. The experimental methodology employed and the obtained results will be outlined in Section IV. Finally, the concluding remarks of the paper and further steps of the research will be presented in Section V.

II. Artificial Immune Network for Dynamic Environments

In this section, the main conceptual aspects associated with this work will be presented. First, a brief description of Artificial Immune Systems and its characteristics will be

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

given, followed by an explanation of the original dopt-aiNet algorithm.

A. Artificial Immune Systems

The Artificial Immune System (AIS) paradigm was created from attempts to model and apply immunological principles to problem solving in a wide range of areas such as optimization, data mining, computer security and robotics [11]. When devoted to the solution of optimization problems, the algorithms developed based under this paradigm present three advantages over other population-based strategies: (*i*) they are inherently able to maintain population diversity (as modules with some resemblance with niching and fitness sharing are intrinsic parts of them); (*ii*) the size of the population at each generation is automatically defined according to the demand of the application; and (*iii*) local optimal solutions are simultaneously preserved once located.

One branch of AIS algorithms follows the framework developed by de Castro & Von Zuben [12], named aiNet, which is based on the clonal selection theory [13] and the Jerne's immune network theory [14], which presents mechanisms for stimulation inhibition of immune cells. In the particular case of this work, all the immune-inspired algorithms consider only suppression, which corresponds to the elimination of some of the self-recognizing cells [7]. One algorithm of the aiNet family was conceived to optimize continuous functions and was named opt-aiNet [9], as an extension of the aiNet framework to deal with multimodal optimization problems. The pseudo-code of the opt-aiNet algorithm is given in Alg. 1.

In contrast with Genetic and Evolutionary Algorithms, in AIS the population of the algorithms is composed of cells, the main mechanism of reproduction is cloning (which generates identical copies of cells), and the only operator adopted for genetic variability is mutation, which is performed at high rates. Also, the network suppression mechanism is responsible for the elimination of redundant solutions.

Algorithm 1 The opt-aiNet algorithm.

```
cell = generate_initial_solutions();
while stop condition not met do
    for each cell i do
        clones = clone(cell_i);
        mutated_clones = mutate(clones);
        cell_i = select(mutated_clones ∪ cell_i);
    end for
    network_suppression();
    insert_new_cells();
end while
```

From Algorithm 1 we can see that the opt-aiNet metaheuristic consists of simple procedures. First, the cell population is initialized by generating uniformly distributed random samples in the search space. After that, the algorithm enters its main loop where, for each cell, Nc clones are generated and mutated by means of a Gaussian operator (see Eq. 1) and then the original cell is replaced with the best clone if it has a better objective-function value. After every cell is mutated, the algorithm starts the process of network suppression, where it tries to detect redundancy: if two solutions are within a distance threshold (using Euclidean measure), the algorithm removes the worst one, thus maintaining the diversity. Subsequently, new solutions are randomly generated and inserted into the population in order to increase diversity. Finally, the process enters a loop until a stop condition is met.

The mutation operator can be described as:

$$c' = c + \alpha \times N(0,1), \tag{1}$$

where c is the clone, c' is the mutated clone, α is the step size (user defined) and $N(0,1)$ is a Gaussian random vector of 0 mean and unitary variance.

It is also easy to notice that the suppression and insertion procedures are responsible for a dynamic variation on the population size. It works just like an implicit objective of delegating each cell to a single optima, by discouraging that two or more cells explore the same region of the search space. Thus, the algorithm not only aims at finding the global optimum, but also a number of local optima as well.

B. The dopt-aiNet algorithm

The dopt-aiNet algorithm was proposed by de França et al. [8] aiming at improving the original opt-aiNet algorithm to deal with dynamic environments. New supplementary operators are created, devising a better distance measure to use within the suppression phase and dividing the multiple cells into Active Population and Memory Population (each cell can be seen as a single population, since it independently evolves to the nearest local optimum by using a radius of influence).

Regarding the mutation operators, in order to quickly locate the nearest local optimum, the Gaussian mutation was modified and two other mutation operators were created. The change in the Gaussian mutation was made in the step size, which is now automatically calculated with a Golden Section procedure [15]. One of the new mutation operations included in dopt-aiNet is the one-dimensional mutation: given a problem in $\Re^n$, n clones are generated and each one suffers a Gaussian mutation in only one of the problem's dimension. The other mutation operator included in this algorithm is the so called Cell Duplication, in which a given variable is randomly selected and its value is copied to the other variables as long as the objective-function value is improved.

Concerning the distance measure used in the suppression phase, it was created a new measure called Line Distance. Given two points, A and B, this distance is calculated by taking the middle point, C, between them, and using these three points to build vectors on the $n+1$-dimensional space by using the objective-function value as the last variable ($A' = [A, f(A)], B' = [B, f(B)], C' = [C, f(C)]$). Therefore, the distance measure is given by the distance between the point C' and the line formed by the points A' and B'.

If both A and B are on the same local optimum, the line formed between A' and B' will be close to the local optimum contour and, thus, the point C' will have a small distance to this line. But if they are on different optima, then the line will just intersect the contours, and the middle point will most likely have a larger distance to this line.

Finally, the population of cells is divided into Active Population and Memory Population: whenever a cell has supposedly found a local optimum (inferred by a number of iterations without any improvement) it is transferred to the Memory Population, where it will no more participate on the mutation procedures, but it will just be a reference to the suppression operations. From time to time, a given cell from this population is selected and its objective-function value is recalculated in order to detect a change in the environment, if this is the case, every Memory cell is copied back to the Active Population.

In [8], some experimental points to a high potential of dopt-aiNet to deal with static and dynamic environments under different conditions.

III. Reducing the computational burden of dopt-aiNet

Although the new procedures created for dopt-aiNet makes the algorithm more robust regarding the optimization of the objective-function and the tracking of moving optima, they are also computational costly. Although each procedure can be natively parallelized, making it able to be applied even on a fast-paced changing environment, in the CEC'2009 competition the frequency of change of the objective function and the performance measures are based on the number of function evaluations. So there would be no gain in parallelization in this situation. In order to make dopt-aiNet more competitive, some adaptations in relevant mechanisms of the algorithm were made.

The most drastic modification was the removal of the two new mutations operators: one-dimensional mutation and cell duplication. Although they tend to improve the performance of the algorithm by reducing the amount of iterations necessary to reach a local optima, they tend to consume a lot of function evaluations (they both perform n function evaluations each time) making the algorithm not suitable to the competition.

Also, after every cell has converged (or the average objective-function value has no significant change over iterations) the suppression and insertion operations are performed together with a replacement of the worst cells. This stimulates that the memory repertoire are constantly changing. The threshold value for this significance was adopted as 0.1 following the imposed condition:

$$|avg_fit() - old\_avg| < 0.1 \tag{2}$$

In order to detect when the environment has changed, each cell is evaluated before the cloning process, so the mutation is always based on the updated value of the objective-function. In the Gaussian mutation process, the Golden Section is limited to search within the range of values of $[-0.01; 0.3]$ (empirically defined) and also the search was limited to 20 function evaluations. With these limitations the Golden Section cannot find the optimal value for the step size but it is still better then using a constant user-defined value.

Another limitation imposed was the number of clones per cells, that was reduced to just 1. The rationale to this is, given that we use the Golden Section to find the best step size, with just one clone the cell is already likely to improve and, thus, reducing the number of function evaluations required per cell in each iteration.

Finally, the population size is started with 10 cells and limited to a maximum of 50 cells. The use of a Memory Population becomes obsolete, since we work with a small and constantly changing population (also excluding the extra cost of maintaining and checking another population). The final parameter left is the cell suppression threshold, that was set empirically to 5. This value is good enough to reveal cells belonging to the same optimum without discarding much of those that were apart.

IV. Experimental Results

In this section the methodology proposed in [10] will be briefly described and the results will be presented in the required format, together with a table of scores. The total mark score will be compared to a simple PSO in two versions, one without any modification (named here as PSO) and another one that restarts whenever the environment changes (called here as $rPSO$). In other words, rPSO is informed when the environment changes, so that it is solving several different static optimization problems. It is important to notice that the dopt-aiNet algorithm will not be informed of any change except when the dimension of the problem changes (as it will be described bellow). The objective of these comparisons are twofold: to find out how a well-known algorithm performs in this situation without any modification (PSO), and to show how an algorithm that is already meant to deal with such scenario (dopt-aiNet) is better than just restarting the algorithm at any detection of change (rPSO).

A. Methodology and Functions

A dynamic optimization problem on this benchmark framework is defined as:

$$F = f(x, \phi, t), \tag{3}$$

where F is the optimization problem, f is the objective function, x is a feasible solution in the search space, t is the real-world time and ϕ is the system control parameter, which determines the solution distribution in the objective-function landscape. The dynamism is associated with a deviation of the solution distribution from the current configuration by tuning the system control parameters as:

$$\phi_{next} = \phi_{current} \oplus \Delta\phi, \tag{4}$$

where $\Delta\phi$ is the deviation.

Additionally, to simulate real world problems, some scenarios promote changes in the dimension of the problems according to Eq. 5, thus leading to a more challenging environment.

$$D_{next} = D_{current} + sign \times \Delta D, \tag{5}$$

with ΔD being the quantity of change and $sign$ is the indication of increase or decrease in the number of variables.

The functions used to create this benchmark framework are depicted in Table I. These functions were then rotated, composed and combined to form 6 different problems with different degrees of difficulty.

With that, the following test problems were created:

F_1: Rotation peak function (with 10 and 50 peaks)
F_2: Composition of Sphere's function
F_3: Composition of Rastrigin's function
F_4: Composition of Griewank's function
F_5: Composition of Ackley's function
F_6: Hybrid Composition function

The basic parameters of the test problems are described in what follows:

Dimension: n(fixed)$= 10$; n(changed)$\in [5, 15]$
Search range: $x \in [-5, 5]^n$
Change frequency: $frequency = 10,000 * n$
The number of changes: $num\_change = 60$
Period: $p = 12$

A total of 7 dynamic scenarios with different degrees of difficulty were proposed:

small step change: a small displacement;
large step change: a large displacement;
random change: Gaussian displacement;
chaotic change: logistic function;
recurrent change: a periodic displacement;
recurrent with noise: the same as above but the optimum never returns exactly to the same point;
changing the number of variables: the dimension of the problem varies.

All the parameters regarding the dynamic displacements and a more detailed explanation of the inner process of this framework can be found in [10] and is omitted here.

B. Results and Discussion

The results were generated considering 20 runs on an Athlon64 3500+ machine with 1GB of RAM, compiled with gcc and under Windows XP Operational System. For each test function in each scenario, it was calculated the average absolute distance from global optima (Eq. 6) at the end of each test case scenario, the average relative distance to the global optima from a sampled set of evaluations (Eq. 7) and a weighted score calculated as described in [10]. Also it was evaluated the average best, average worst, average mean and standard deviation of the distance between the best solution and the global optimum at that given time during a sampling period of time for each problem.

$$E^{last}(t) = |f(x_{best}(t)) - f(x^*(t))| \tag{6}$$

$$E(t) = \frac{\frac{f(x^*(t))}{x_{best}}}{1 + \frac{\sum 1 - \frac{f(x^*(t))}{x_{best}}_{sampledset}}{\frac{change\_frequency \times dimension}{sample\_frequency}}}, \tag{7}$$

Eqs. 6 and 7 are devoted to minimization problems. For maximization problems, the inverse of the fraction $\frac{f(x^*(t))}{x_{best}}$ should be used instead.

In Table II we can see the general performance of dopt-aiNet on these benchmark experiments. The best way to analyze these results is by looking at the average relative distance and score values, as they depict how close, in average, the algorithm was from the optimum during the optimization process (not just by the end of the run but during the whole optimization). At the first problem, that presents just a few peaks (alternating from 10 to 50), the algorithm had the best performance in each of the proposed scenarios. At the third and the sixth functions (composition of Rastrigin and hybrid composed functions, respectively) the algorithm presents a worse performance due to the difficulty in finding the global optima even when the environment is static (the longer it takes for the algorithm to find the optimum the lower the score). At the other functions, the dopt-aiNet algorithm performed fairly well and can be competitive with other approaches.

When comparing total mark score with PSO and rPSO, in Table III, we can see that the performance of the PSO without restarting is much worse, as expected, than the other two algorithms, and that the score of dopt-aiNet when compared to the informed PSO is about 20% better, pointing out that the proposed meta-heuristic is capable of using the past experiences and multipopulational approach to deal with dynamic environments, even though with some features of the original dopt-aiNet being stripped out in these experiments.

Tables IV to IX provide more detailed information about the results for each function at each scenario as it depicts the average mean, best and worst error values (difference between the global optimum and the best result found so far). It is specially useful to look at the best and mean average, the first one because it shows how close the algorithm got from the optimum during the entire scenario and the second because it shows the speed of convergence (from Eq. 7, if we have a lower average it means the algorithm has found a good result sooner, since the average is taken from several different instants of the optimization process). Regarding F_1, Table IV confirms what was pointed out earlier that dopt-aiNet could find good results quickly for this function, no matter the scenarios. For function F_2, on Table V, the results indicate that, although the algorithm was able to get close to the optima, it still struggled along the search in some of the scenarios, specially T_5 (recurrent change) where its average mean was very high compared to the other cases.

About F_3, Table VI shows that the dopt-aiNet had some difficulties in finding the global optimum as its best values

TABLE I

DETAILS OF THE BASIC BENCHMARK FUNCTIONS

name	function	range
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]
Rastrigin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5,5]
Weierstrass	$f(x) = \sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$	
	$a = 0.5, b = 3, k_{max} = 20$	[-0.5,0.5]
Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[-100,100]
Ackley	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	[-32,32]

TABLE II

AVERAGE ABSOLUTE DISTANCE FROM GLOBAL OPTIMA, AVERAGE RELATIVE DISTANCE FROM GLOBAL OPTIMA AND SCORE OBTAINED BY DOPT-AINET IN THE 6 PROPOSED PROBLEMS UNDER THE 7 DIFFERENT SCENARIOS.

problem	change type	avg. abs.	avg. rel.	score
1	1 (10 peaks)	0.13528	0.902411	0.0135362
1	1 (50 peaks)	0.364426	0.894062	0.0134109
1	2 (10 peaks)	5.86683	0.75644	0.0113466
1	2 (50 peaks)	4.74855	0.79796	0.0119694
1	3 (10 peaks)	4.25452	0.763523	0.0114528
1	3 (50 peaks)	5.25312	0.791155	0.0118673
1	4 (10 peaks)	5.35633	0.775512	0.0116327
1	4 (50 peaks)	2.65653	0.806397	0.012096
1	5 (10 peaks)	4.43562	0.706345	0.0105952
1	5 (50 peaks)	2.86407	0.742108	0.0111316
1	6 (10 peaks)	9.94074	0.674239	0.0101136
1	6 (50 peaks)	6.83305	0.706515	0.0105977
1	7 (10 peaks)	4.21102	0.769844	0.00769844
1	7 (50 peaks)	4.41727	0.785618	0.00785618
2	1	0.0984018	0.732236	0.0175737
2	2	8.12093	0.504531	0.0121088
2	3	17.9979	0.478578	0.0114859
2	4	1.06527	0.658821	0.0158117
2	5	101.384	0.305631	0.00733513
2	6	6.51929	0.503996	0.0120959
2	7	3.73853	0.570433	0.00912693
3	1	810.83	0.0151338	0.000363212
3	2	1078.75	0.00836012	0.000200643
3	3	1073.43	0.00852786	0.000204669
3	4	1031.53	0.00586033	0.000140648
3	5	1023.9	0.018781	0.000450745
3	6	1186.9	0.00516513	0.000123963
3	7	1061.83	0.00880818	0.000140931
4	1	1.42272	0.66186	0.0158847
4	2	122.441	0.289467	0.00694722
4	3	98.6688	0.387061	0.00928945
4	4	4.26323	0.570938	0.0137025
4	5	304.566	0.153231	0.00367754
4	6	12.6334	0.454639	0.0109113
4	7	52.901	0.424358	0.00678973
5	1	40.8943	0.387405	0.00929773
5	2	34.4531	0.361252	0.00867004
5	3	34.942	0.357802	0.00858724
5	4	120.637	0.283915	0.00681397
5	5	943.223	0.0599851	0.00143964
5	6	480.337	0.13306	0.00319345
5	7	219.466	0.233935	0.00374296
6	1	20.4434	0.462213	0.0110931
6	2	391.196	0.137545	0.00330109
6	3	456.441	0.118135	0.00283523
6	4	83.9698	0.362031	0.00868876
6	5	845.862	0.0341991	0.000820778
6	6	482.207	0.106351	0.00255242
6	7	372.474	0.13733	0.00219729
Total mark (100*sum(score)):				38.2904

TABLE III

COMPARISON OF THE TOTAL SCORE AMONG DOPT-AINET, PSO AND RPSO

	dopt-aiNet	**PSO**	**rPSO**
Total mark score:	38.2904	0.0014	31.5992

were very high in each one of the scenarios, perhaps caused by the lack of the two supporting mutations and limited resources that was available. Concerning F_4, depicted in Table VII, the dopt-aiNet got very close to the global optimum although it presents a slow convergence rate in some of the scenarios (T_2, T_3, T_5 and T_7). On average, however, it can be considered a good performance since we are dealing with a multimodal function with many local optima. F_5, as shown on Table VIII, produces a similar result when compared to F_4, as it stays close to the global optimum, also presenting the same slow convergence rate in some of the scenarios.

Finally, Table IX presents again a behavior similar to what has been observed on highly multimodal functions, as it is the case here, of slow convergence to the optimum, exceptionally for T_5 where again it was not much close, but it still had presented much improvement over the average worst, implying that the algorithm started with a bad solution and has tried to optimized as far as it could along the available function evaluations. Figure 1 shows the convergence plot for each function on the experiments with a fixed dimension (T_1 to T_6).

The source code of the dopt-aiNet algorithm and the output obtained for the benchmark problems can be found at `http://sites.google.com/site/fabricioolivetti/toolboxes/DBG_dopt_src.zip` and `http://sites.google.com/site/fabricioolivetti/toolboxes/DBG_dopt_relat.zip`, respectivelly.

V. FINAL COMMENTS

In this paper the dopt-aiNet algorithm, proposed as an extension of the opt-aiNet for dynamic environments, was adapted and tested against a set of benchmark instances with different challenging scenarios on dynamic optimization. The original algorithm was submitted to some modifications, having some of its mutation operators removed and adjusted

to use limited resources in order to act more promptly toward the production of competitive results.

The results have shown that the adjusted dopt-aiNet could track the global optima very fast most of the time, except for one of the functions as it could not reach the global solution. For now, a simple comparison was made with a PSO algorithm that is restarted at each time the environment changes. One of its remarkable feature, the dynamic population size for maintaining diversity, could not be properly utilized on this experimental setup due to the restrictive total number of evaluations and, thus, the diversity of solutions may have produced a negative effect on the results.

But, since the dopt-aiNet is a parallel algorithm (the evolution of each cell does not depend on the entire network most of the time) and nowadays the cost of multiprocessing has become affordable by means of multicore programming or graphic cards' multiple gpu programming (like NVidia's CUDA), the algorithm could be used without any modifications to improve its overall performance.

As future work, we intend to unify the mutation operators into a single one, that is capable of making the current solution converge to the nearest local optima as quick as possible, create a self-adjustable network of cells and a more efficient separation of active and memory population in order to spare some function evaluations by detecting when a given cell cannot be improved anymore.

ACKNOWLEDGMENT

The authors would like to thank CNPq and CAPES for their financial support.

REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. Wiley-Interscience, 2006.

[2] Z. Michalewicz and D. B. Fogel, *How to solve It: Modern Heuristics*. Springer, 2000.

[3] C. Junqueira, F. de Franca, R. Attux, R. Suyama, L. de Castro, F. V. Zuben, and J. Romano, "A proposal for blind FIR equalization of time-varying channels," *Machine Learning for Signal Processing, 2005 IEEE Workshop on*, pp. 9–14, Sept. 2005.

[4] C. Junqueira, F. O. de Franca, R. Attux, C. Panazio, L. de Castro, F. V. Zuben, and J. Romano, "Immune-inspired dynamic optimization for blind spatial equalization in undermodeled channels," *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 2896–2903, 2006.

[5] F. de Franca, L. Gomes, L. de Castro, and F. V. Zuben, "Handling time-varying TSP instances," *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 2830–2837, 2006.

[6] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[7] L. N. de Castro and F. J. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[8] F. O. de França, F. J. Von Zuben, and L. N. de Castro, "An Artificial Immune Network for Multimodal Function Optimization on Dynamic Environments," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Washington D.C., USA, 2005.

[9] L. N. de Castro and J. Timmis, "An artificial immune network for multimodal optimisation," in *2002 Congress on Evolutionary Computation. Part of the 2002 IEEE World Congress on Computational Intelligence.* Honolulu, Hawaii, USA: IEEE, May 2002, pp. 699–704. [Online]. Available: http://www.cs.ukc.ac.uk/pubs/2002/1374

[10] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for CEC'2009 competition on dynamic optimization," Available at http://www3.ntu.edu.sg/home/EPNSugan/index_files/CEC-09-Dynamic-Opt/CEC09-Dyn-Opt.htm., University of Leicester, University of Birmingham, Nanyang Technological University, Tech. Rep., September 2008.

[11] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[12] L. N. de Castro and F. J. Von Zuben, "aiNet: An Artificial Immune Network for Data Analysis," in *Data Mining: A Heuristic Approach*, H. A. Abbass, R. A. Sarker, and C. S. Newton, Eds. Idea Group Publishing, 2001, pp. 231–259.

[13] F. M. Burnet, "Clonal selection and after," in *Theoretical Immunology*, G. I. Bell, A. S. Perelson, and G. H. Pimgley Jr, Eds. Marcel Dekker Inc., 1978, pp. 63–85.

[14] N. K. Jerne, "Towards a network theory of the immune system," *Ann. Immunol., Inst. Pasteur*, vol. 125C, pp. 373–389, 1974.

[15] J. Kiefer, "Sequential minimax search for a maximum," in *Proceedings of the American Mathematical Society*, vol. 4, 1995, pp. 502–505.

TABLE IV

ERROR VALUES ACHIEVED FOR PROBLEMS F_1

Dimension(n)	Peaks(m)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	10	Avg_best	0.0048	0.0027	0.0052	0.0076	0.0052	0.0087
		Avg_worst	5.1786	46.1036	41.4286	37.0052	19.5234	71.4790
		Avg_mean	0.1353	5.8667	4.2545	5.3563	4.4356	9.9407
		STD	1.0061	10.2772	8.1828	8.9414	5.5545	15.8214
	50	Avg_best	0.0072	0.0040	0.0057	0.0118	0.0078	0.0104
		Avg_worst	4.5776	29.9379	33.7780	37.9725	24.1907	62.4719
		Avg_mean	0.3644	4.7485	5.2531	2.6565	2.8641	6.8330
		STD	0.9275	6.7580	6.6830	5.9773	4.1579	11.8790
T_7(5-15)	10	Avg_best	—	—	0.0022	—	—	—
		Avg_worst	—	—	39.5959	—	—	—
		Avg_mean	—	—	4.2110	—	—	—
		STD	—	—	8.6873	—	—	—
	50	Avg_best	—	—	0.0039	—	—	—
		Avg_worst	—	—	25.0307	—	—	—
		Avg_mean	—	—	4.4172	—	—	—
		STD	—	—	6.4528	—	—	—

TABLE V

ERROR VALUES ACHIEVED FOR PROBLEMS F_2

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	0.0534	0.0678	0.0813	0.0596	0.1032	0.0582
	Avg_worst	0.2102	68.0774	473.8170	14.0593	441.2040	51.9411
	Avg_mean	0.0984	8.1209	17.9979	1.0652	101.3840	6.5192
	STD	0.0291	14.3832	62.2259	2.8269	134.5180	13.8172
T_7(5-15)	Avg_best	—	—	0.0663	—	—	—
	Avg_worst	—	—	36.9357	—	—	—
	Avg_mean	—	—	3.7385	—	—	—
	STD	—	—	7.9542	—	—	—

TABLE VI

ERROR VALUES ACHIEVED FOR PROBLEMS F_3

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	674.0810	943.8250	943.781	727.1850	907.9080	691.9480
	Avg_worst	1103.6600	1270.5000	1240.51	1644.5500	1202.0900	1834.1700
	Avg_mean	810.8300	1078.7500	1073.4300	1031.5300	1023.90	1186.9000
	STD	66.1085	64.1245	64.9950	274.7490	57.8713	292.2960
T_7(5-15)	Avg_best	—	—	786.6300	—	—	—
	Avg_worst	—	—	1259.2700	—	—	—
	Avg_mean	—	—	1061.8300	—	—	—
	STD	—	—	110.0980	—	—	—

TABLE VII

ERROR VALUES ACHIEVED FOR PROBLEMS F_4

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	0.0679	0.1222	0.0864	0.0543	0.1497	0.0618
	Avg_worst	26.0705	586.2790	580.6420	51.9689	562.5500	336.7740
	Avg_mean	1.4227	122.4410	98.6688	4.2632	304.5660	12.6334
	STD	4.5459	201.6270	196.6950	9.7255	203.2430	55.8386
T_7(5-15)	Avg_best	—	—	0.0864	—	—	—
	Avg_worst	—	—	547.4190	—	—	—
	Avg_mean	—	—	52.9010	—	—	—
	STD	—	—	130.5930	—	—	—

TABLE VIII

ERROR VALUES ACHIEVED FOR PROBLEMS F_5

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	0.2511	0.4368	0.3469	0.2172	11.5370	0.3173
	Avg_worst	1728.1000	705.1520	786.2750	1375.1600	1927.6400	1910.6400
	Avg_mean	40.8943	34.4531	34.9420	120.6370	943.2230	480.3370
	STD	221.2120	119.8960	115.0250	293.5420	633.3180	610.8020
T_7(5-15)	Avg_best	—	—	0.2742	—	—	—
	Avg_worst	—	—	1651.6300	—	—	—
	Avg_mean	—	—	219.4660	—	—	—
	STD	—	—	427.8170	—	—	—

TABLE IX

ERROR VALUES ACHIEVED FOR PROBLEMS F_6

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	0.1003	0.2871	0.2873	0.1074	38.4149	0.1442
	Avg_worst	418.3680	937.3390	1018.5300	906.2330	1101.5000	1324.4200
	Avg_mean	20.4434	391.1960	456.4410	83.9698	845.8620	482.2070
	STD	79.3230	395.4350	405.0380	220.1770	251.2080	434.4210
T_7(5-15)	Avg_best	—	—	3.8001	—	—	—
	Avg_worst	—	—	1077.8100	—	—	—
	Avg_mean	—	—	372.4740	—	—	—
	STD	—	—	394.6680	—	—	—

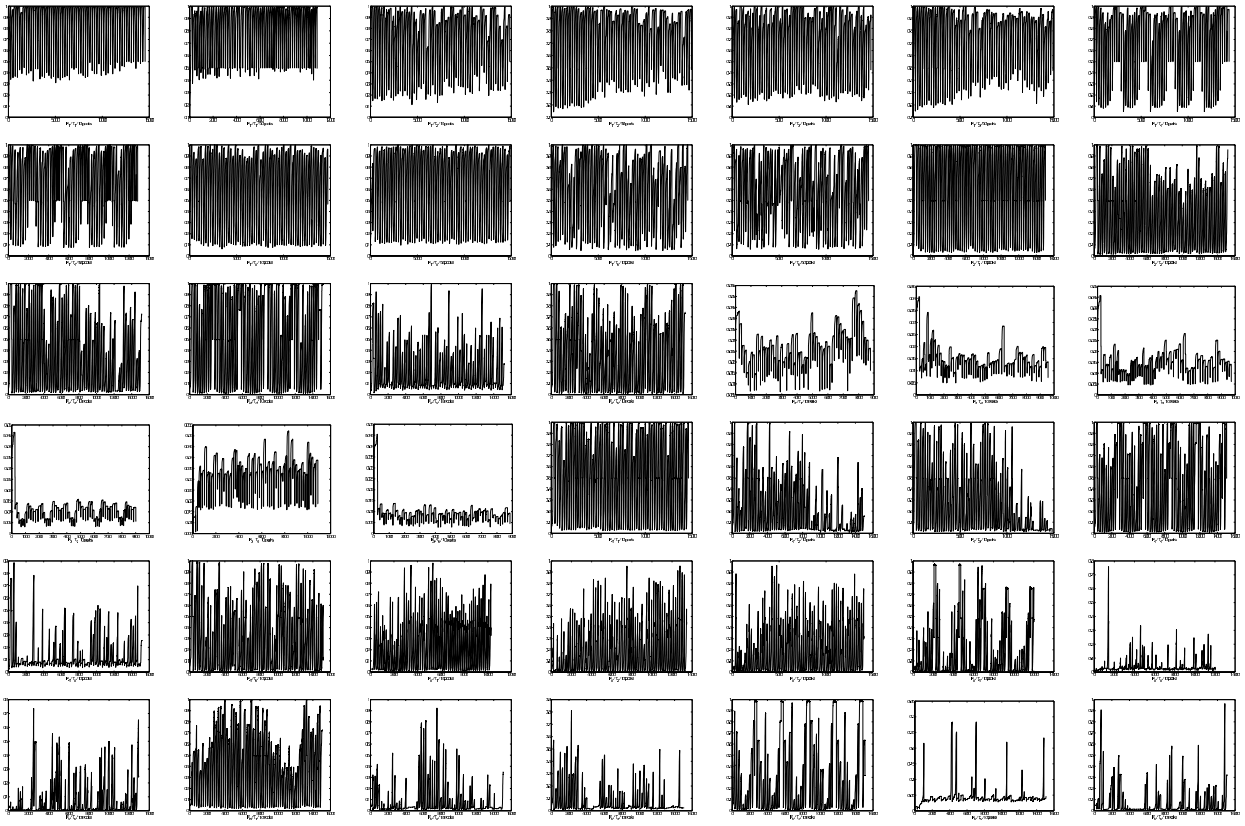

Fig. 1. Convergence behavior for each function and enviromental condition with dimension = 10, following the same sequence as the one in Table II.