

Hybrid Immune Algorithm for Multimodal Function Optimization

Małgorzata Lucińska¹ and Sławomir T. Wierzchoń^{2,3}

¹ Kielce University of Technology, Kielce, Poland

² Institute of Computer Science Polish Academy of Sciences, Warsaw, Poland

³ University of Gdańsk, Gdańsk, Poland

Abstract

This paper presents a new tool for multimodal function optimization based on the natural immune system metaphor. The proposed algorithm combines elements of immune network and clonal selection together with a heuristic search method. It is used for optimization of multimodal functions. The hybrid approach allows for fast localization of the optima and outperforms other presented immune approaches to the function optimization problem.

Keywords: Artificial immune systems, clonal selection, function optimization

1 Introduction

Artificial immune systems (AIS for brevity) are computational systems inspired by theoretical immunology, observed immune functions, principles and mechanisms in order to solve problems — de Castro and Timmis (2002). The fundamental features of the natural immune system, like distribution, adaptability, learning from experience, complexity, and coordination have decided that immune algorithms have been applied to a wide variety of tasks, including optimization.

The task of the immune system is to identify and destroy foreign invaders or antigens. The natural immune system is composed of lymphocytes — B lymphocytes (B-cells) and T lymphocytes (T-cells). B-cells produce antibodies, which bind to the invading antigens and help destroy them. Each B-cell produces only one kind of antigenic receptors. When an antigen enters the body, it activates only the lymphocytes whose receptors can bind to it. Activated by an antigen and with a second signal from accessory cells, such as the T-cells, the B-cells proliferate (divide) producing large number of clones. In the final stage these clones can mutate in order to produce antibodies with very high affinity to a specific antigen. The process is explained by the clonal selection principle, according to which only those cells that recognize the antigens are selected to proliferate. The selected cells are subject to affinity maturation, which improves their affinity to the antigens.

There have been a number of algorithms developed for optimization, inspired by the clonal selection principle and affinity maturation process, for example de Castro and Timmis (2002), de Castro and Von Zuben (2002), and Walker and Garrett (2003), to name a few.

Recently many hybrid approaches, joining immune metaphor with other biological or heuristic inspirations, have been proposed to solve optimization problems e.g. Liu and Xu (2008). This paper introduces a new optimization algorithm — Hybrid Immune Algorithm (HIA), which in spite of immune inspirations uses a heuristics search method. It gradually limits search area of an optimized function domain and leads to faster convergence to the optimum.

This paper begins with a brief introduction of the most representative immune optimization algorithms. A detailed description of the HIA steps is then presented together with theoretical comparison to opt-aiNet algorithm. Next section describes dependencies between different algorithm parameters and its performance. The article ends with comparison of HIA to other immune algorithms results and conclusions.

2 Artificial Immune Systems for Function Optimization

One of widely established immune optimization algorithms is opt-aiNet, consult de Castro and Von Zuben (2001) and de Castro and Timmis (2002) for details. Immune network algorithm AiNet was initially proposed to perform data compression and clustering tasks and then adapted to optimization problems as opt-aiNet. The main goal of the algorithm is to find all local and global function optima and to maintain them. It creates a population of antibodies, which are encoded as real-valued vectors in an Euclidean space and represent candidate solutions to the function being optimized. They are evaluated against the objective function (affinity determination) and cloned proportionally to their affinity. Then cells undergo a process of mutation, which is inversely proportional to their affinity. According to clonal selection principle the clones with the highest affinity are placed into a clonal memory set. The chosen clones interact with each other, that results in suppression of the individuals, whose affinity with others is less than a specified threshold. The remaining clones are incorporated with antibody network. In the last step new randomly generated antibodies join the network, that is akin to the metadynamics of the system. Network suppression eliminates any similar or non-stimulated antibodies and antibodies that fall below the determined suppression threshold. One of the most important feature of opt-aiNet algorithm is a dynamic population size, controlled by the suppression mechanism.

Another benchmarking optimization algorithm, called B-cell algorithm (BCA), is also inspired by the clonal selection process — Kelsey and Timmis (2003). However, unlike opt-aiNet, it focuses on finding the global function optimum in complicated landscape. It employs N-dimensional vectors of 64-bit strings, which represents bit-encoded double precision numbers. These vectors are considered to be B-cell within the system. After evaluation with the objective function each B-cell is cloned to produce a clonal pool. In order to ensure metadynamics of the system one clone from a pool is selected at random and undergoes a random changes, according to a certain probability. Each B-cell is then subject to a special contiguous somatic hypermutation mechanism. This is a special feature of the BCA and operates by subjecting contiguous regions of the vector to mutation.

Both opt-aiNet and BCA have received a lot of attention in the AIS literature,

whereas the Cooperative Immune Network with Particle Swarm Behavior (CoAIN) algorithm described by Liu and Xu (2008) represents the current achievements of immune metaphor strategies in the field of function optimization. It employs a model of cooperative artificial immune network, in which interactions are not only suppression but also cooperation. The cooperative strategy is inspired by particle swarm behavior, each network cell cooperates as particle flying around in a multidimensional searching space to adjust position according to its own experience and the experience of its neighbor, making use of the best position encountered by it and other particles. Another feature of the algorithm is dynamical change of number of clones, that depends on their fitness values and size of mutation.

The above introduced algorithms are just representative examples of many others immune optimization algorithms and are used as benchmark solutions in a later section of the article. When compared with other evolutionary approaches to the problem of function optimization, immune metaphor generates better results for multi-modal functions, eg. Cutello *et al.* (2005). Interactions between cells of the immune system and with an environment allow to escape from local optima and to converge fast to global solution.

3 Hybrid Immune Algorithm

Hybrid Immune Algorithm (HIA) can be thought of as an extension of the opt-aiNet algorithm. It employs a very similar system model and uses the same terminology as de Castro and Timmis (2002):

- Cell (antibody): individual, which is described as a real-valued vector in an Euclidean shape-space;
- Fitness: fitness of a cell equals a value of the objective function in a point with coordinates representing the given cell;
- Affinity: Euclidean distance between two cells;
- Clone: offspring cells, that are identical copies of their parent cell. They undergo somatic mutation in a further step.

For the sake of clarity we assume that the algorithm searches for function minimum. Main steps of the HIA are the following:

- Initialization. Generate a set P of candidate solution.
- Cloning and mutation. Each element $p \in P$ produces a given number of clones, $|c|$, which are subject to mutation.
- Selection. For each element $p \in P$ find the best clone c^p with the smallest value of the objective function among its clones. If $f(c^p) < f(p)$ replace p by c^p , else increase age of the element p .
- Affinity maturation. If an individual p reaches a given age move it to the memory set M and generate a new individual in the population set.
- Suppression. For each element $m \in M$ determine the similarity with other antibodies in the memory set. From a pair (m, m^*) of antibodies, whose affinity is less than a specified threshold, eliminate the worse element. Limit a size of the memory to $|m|$ cells.

- Repeat steps 2-5 until a termination condition is met.

A population is initialized randomly. A number of generated cells is a parameter of the algorithm, chosen experimentally. Each cell produces a constant number of clones (a parameter). Mutation of the clones is realized in two ways. The first one uses Gaussian distribution independently for each dimension. A mean value in the distribution equals the parent's coordinate and a standard deviation depends on a change of the coordinate during the previous iteration. The following formula describes the process:

$$c_k = N(p_k, \alpha_k) \quad (1)$$

$$\alpha_k = 2(p_{k-1} - c_{k-1}^p) \quad \text{if } (c_{k-1}^p) < f(p_{k-1}) \quad (2)$$

$$\alpha_k = \alpha_{k-1} \quad \text{if } (c_{k-1}^p) \geq f(p_{k-1}) \quad (3)$$

where c_k and p_k are vectors of coordinates describing appropriately a mutated cell and a parent cell in an iteration k , $N(p_k, \alpha_k)$ is a Gaussian random variable of parent coordinates mean and standard deviation α_k . The standard deviation is a vector with coordinates proportional to a difference between the parent cell and its best clone coordinates in the previous iteration, if the best clone fitness is better than that of the parent. Otherwise the standard deviation remains the same as in the previous iteration.

The second way of mutation concerns only one dimension, the other coordinates of the mutated clone remain the same as of the parent cell. The dimension is chosen randomly and a new value of the coordinate is generated with a help of uniform distribution within appropriated ranges.

$$c_k^i = p_k^i \quad \text{for } i \in (1, D) \quad \text{and } i \notin s \quad (4)$$

$$c_k^s = r \quad r \in (x_{min}^s, x_{max}^s) \quad (5)$$

where c_k^i and p_k^i are i -th coordinates of vectors describing appropriately a mutated cell and a parent cell in an iteration k ; s indicates a randomly chosen dimension, for which a coordinate is changed; r is a new value of the coordinate (in the dimension s), generated at random within a range limited by the minimum x_{min}^s and maximum x_{max}^s values allowed for this dimension; D is a number of dimensions.

The second kind of mutation is carried out in 20% of cases (discussed later in the article), which are also chosen at random. The first type of mutation is focused mainly on exploitation of a search space, whereas the second one enables better exploration avoids getting stuck in a local extreme.

In the next step of the algorithm the mutants are evaluated against the objective function to be optimized. Best individual is selected from all the mutants produced by one cell. Fitter descendants replace their parents. For each dimension a change of the coordinate value is calculated, which equals a difference between two coordinate values — one of the newly chosen best cell in the generation and the second one of the old parent. The change value will be used in the next iteration as a standard deviation in the Gaussian distribution for mutation purpose (2). In case a cell is not replaced by its clones, its age is increased. Antibodies older than an established threshold (a parameter) are removed from the population and placed in a memory set. Similar policy is used in the opt-IMMALG algorithm by

Cutello *et al.* (2005). The authors have introduced so called aging operator, which eliminates old cells in a population. The mechanism allows for maintaining high diversity in order to avoid premature convergence. The difference between the two algorithms lies in a possibility of further use of the old antibodies as memory cells in HIA. Cells creating the memory interact with each other. If affinity between two individuals is smaller than a specified value (suppression threshold) the worse one is removed from the memory. In addition the memory has a size $|m|$ limited to a few fittest cells. The antibody, which was moved to the memory, is replaced in the population by a new individual. At the first stage of the algorithm the new cells are generated randomly within the whole search space. When memory size reaches its specified value $|m|$ newcomers are created with a use of Gaussian distribution. In each dimension a mean value equals a coordinate of the best cell in the memory and standard deviation depends on the maximum difference between any two memory cells coordinates in the appropriate dimension. The creation of new cells is described by the formulas:

$$a_k^i = N(b_k^i, \sigma_k^i) \quad \text{for } i \in (1, D) \quad (6)$$

$$\sigma_k^i = \beta_k \sigma_{k-1}^i + (1 - \beta_k) \Delta_k^i \quad (7)$$

$$\Delta_k^i = \max\{m_{kc}^i - m_{kd}^i; \quad c, d \in (1, |m|)\} \quad (8)$$

$$\beta_k = \beta_0 (1 - (1 - 1/k)^q) \quad (9)$$

where a_k^i is i -th coordinate of the new cell, generated with a use of Gaussian distribution in the iteration k ; b_k^i stands for the i -th coordinate of the best cell in the current memory; β_k is a learning factor in the iteration k ; $\beta_0 = 0.8$ and $q = 5$; Δ_k^i describes the maximum difference between any two memory cells coordinates; m_{kc} , m_{kd} represent cells of the current memory.

In the initial phase of the HIA execution new cells are generated within the whole search area, this results in quite intensive exploration. Later, when some information about the landscape is discovered, the search area becomes more and more narrowed. Exploration gives gradually way to exploitation. Standard deviation of the Gaussian distribution, responsible for reduction of the search area, is controlled by a learning factor β_k . It prevents too fast convergence to a local extreme. In case a use of the current distribution parameters leads to generation of cells and clones, which are not able to achieve a fitness level of the best memory cell during three subsequent iterations, they are replaced by old parameter values. Such a policy avoids exploitation of misleading areas.

The stopping criterion can base upon an achievement of the optimum value or a number of iterations.

There is a number of similarities between the HIA and opt-aiNet algorithms. Both of them use real-valued vectors in order to represent individuals of the population. Another thing in common is a Gaussian mutation, however they differ in ways of establishing the distribution parameters. Opt-aiNet has a standard deviation, that is inversely proportional to the parent fitness, normalized for all the cells in the population. The standard deviation in HIA depends only on the parent coordinate changes. The algorithms also share the same selection mechanism, where each individual generates $|c|$ offsprings and then the fittest cell is chosen

from the joined set of the parent and its offspring. Suppression of cells with an affinity smaller than a given threshold is another similarity of the two algorithms. Both in opt-aiNet and in HIA newcomers are allowed to enter the population in case that the current cells cannot significantly improve fitness.

On the other hand there are some crucial differences between the two algorithms. Opt-aiNet has a dynamic adjustment of the population size, whereas in HIA the population is fixed and can consist of only one individual. The same concerns size of the memory. According to opt-aiNet strategy it is regulated by suppression with other cells, in HIA the number of cells preserved in the memory is fixed. The main difference lies however in a way of production of new cells. HIA adopts quite novel policy, according to which newcomers are created not with a help of uniform distribution within the whole search area (as in the original version of opt-aiNet) but using Gaussian distribution, with dynamically adjusted parameters. Such a method allows for gradual limitation of a search space without a threat of too fast decrease of exploration, which is accomplished by the second type of mutation.

Andrews and Timmis (2006) propose a modification of opt-aiNet, which also concerns creation of new cells. They introduce a diversity operator and demonstrate that it increases the chance of locating optima on the boundaries of the function. However, the operator application to clustering problem does not change performance of aiNet. The way the new cells are created with a help of the diversity operator is quite different from the idea presented in this paper. In HIA the parameters governing creation of new cells depend on best cells in the system memory. At later stages of the algorithm operation, when it converges to the optimum, new cells are very similar to the best memory cell with high probability. Whereas in modified opt-aiNet cells are more often created in the area near the boundaries of the function domain. Moreover the diversity operator parameters do not depend directly on the algorithm stage.

The new elements introduced in HIA result in quite different features of the two algorithms. Opt-aiNet never pretends to be fast in terms of number of function evaluations. Its strategy concentrates on finding as well as maintaining all the global and local optima and if applicable determining their number. Contrary to it HIA heads to find fast an exact value of the global optimum. In such a case a number of function evaluations is the most important metric taken into account while assessing quality of the solution.

4 Results

We have implemented the above algorithm in Java language. The performance of HIA was investigated with a use of benchmark functions published by Timmis *et al.* (2004) and Liu and Xu (2008). All the benchmark functions are multimodal and non-separable. They cover a very wide range of landscape types to allow for assessment of the algorithms performance in very different conditions. All the functions and their domain ranges are shown in the Table 1. HIA was executed until the optimum value was found and all the presented results are an average value calculated on basis of 10000 independent experiments, unless indicated dif-

TABLE 1: Functions for experiments.

Id	Function	Parameters
f1	$2(x - 0.75)^2 + \sin(5\pi x - 0.4\pi) - 0.125$	$x \in [0, 1]$
f2	$-\sum_{j=1}^{10} j \sin((j+1)x + 1)$	$x \in [-10, 10]$
f3	$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$x_1 \in [-5, 10], x_2 \in [-10, 10]$
f4	$\sum_{j=1}^5 j \sin((j+1)x_1 + j) \sum_{j=1}^5 j \sin((j+1)x_2 + j) + 0.5[(x_1 + 1.4513)^2 + (x_2 + 0.80032)^2]$	$x_1 \in [-10, 10], x_2 \in [-10, 10]$
f5	$\sum_{j=1}^5 j \sin((j+1)x_1 + j) \sum_{j=1}^5 j \sin((j+1)x_2 + j) + (x_1 + 1.4513)^2 + (x_2 + 0.80032)^2$	$x_1 \in [-10, 10], x_2 \in [-10, 10]$
f6	$\frac{x_1^4}{4} - \frac{x_2^2}{2} + \frac{x_1}{10} + \frac{x_2^2}{2}$	$x_1 \in [-10, 10], x_2 \in [-10, 10]$
f7	$\sum_{j=1}^5 j \sin((j+1)x_1 + j) \sum_{j=1}^5 j \sin((j+1)x_2 + j)$	$x_1 \in [-10, 10], x_2 \in [-10, 10]$
f8	$-x_1 \sin(4\pi x_1) + x_2 \sin(4\pi x_2 + \pi) - 1$	$x_1 \in [-2, 2], x_2 \in [-2, 2]$
f9	$-0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	$x_1 \in [-10, 10], x_2 \in [-10, 10]$

ferently. For the sake of clarity, if results do not contribute to any new trends, some entries are left blank in the presented tables.

First we have investigated algorithm effectiveness varying a number of clones $|c|$ produced by a parent cell and a suppression threshold σ . Both parameters have great influence on the opt-aiNet algorithm performance. Too small values of $|c|$ ($|c| < 30$ and $\sigma = 0.02$) led to slow convergence to the optimum value and rapid grow of the population size. Larger number of clones produced by one cell result in better exploration of the search area and smaller population size. Sigma parameter controls the accuracy of obtained optima values. The smaller sigma value the higher is the accuracy. Table 2 shows relation between number of clones produced by a parent cell and number of evaluations, necessary to find optimum values of benchmark functions by HIA algorithm. One-dimensional functions (f1, f2) or functions with many densely distributed global optima (f7, f8) require quite small number of clones — 3 or 4. In case of quite smooth functions with a few global optima the best results are obtained for $|c| = 6$. The largest number of clones produced by a parent cell is necessary in case of sophisticated functions with many local and one global optimum (f4, f5, f9). However, even for these functions best results are achieved when $|c|$ is quite small comparing to opt-aiNet algorithm and equals 10-12. In case of functions with easily found global optima exploration of the area can be limited, only functions with many local optima, with values close to the global one require extensive exploration by many clones. Further increase of the parameter $|c|$ leads to worse performance of the algorithm because of waste of resources.

Table 3 shows how number of evaluations depends on a suppression threshold sigma. Difference between quite "simple" functions or functions with densely distributed optima and "difficult" functions with many local optima is even more visible taking into consideration a sigma value. The former ones reveal best per-

TABLE 2: Effect of number of clones on number of function evaluations. Values of the other parameters as in Table 5. The first figure indicates number of function evaluations averaged for 1000 experiments and the second one standard deviation.

$ c $	2	3	4	5	6	7	9	10	12	15
f1	47	47	47	50	50	53	55	61	-	-
	48	47	41	41	42	34	40	42	-	-
f2	126	126	139	146	150	149	166	169	-	-
	140	132	159	151	131	131	117	119	-	-
f3	229	201	191	178	173	175	183	188	-	-
	288	219	183	160	137	127	123	136	-	-
f4	-	104573	70406	38908	28766	27285	22394	19227	14323	17092
	-	219355	171548	95149	57479	77747	104995	59702	47944	89078
f5	-	-	-	21954	-	14156	-	11973	7691	10692
	-	-	-	63089	-	32342	-	33999	20951	49093
f6	212	179	173	173	166	171	177	178	-	-
	271	174	188	187	154	151	138	148	-	-
f7	352	343	359	359	361	361	403	410	-	-
	441	332	416	417	219	250	295	263	-	-
f8	259	254	239	250	257	257	276	290	-	-
	225	212	170	185	175	161	164	174	-	-
f9	-	7916	-	7044	-	6852	6326	6268	6565	6778
	-	7497	-	6770	-	6838	5920	5675	6516	6576

formance for small values of sigma (0.0002) whereas the latter ones require higher sigma values (2 or 4). Such behavior of the algorithm is not astonishing since an increase of a suppression threshold value enlarges distances between memory cells and leads to more intensive exploration in HIA — new cells are more sparsely distributed.

We have also conducted some analysis concerning the special features of HIA. One of the HIA novelties lies in introduction of two types of mutation. The first one uses Gaussian distribution within an area limited by the distribution parameters whereas the second type concerns only one dimension and relies on uniform distribution. Table 4 shows how number of function evaluations for each benchmark function depends on a percentage of cases the second type of mutation takes place. Using only one type of mutation never leads to best performance of the algorithm. For most functions the best results are achieved when the second type of mutation is applied in 20% of cases. For unimodal functions without many local optima, f1 and f6 (Quartic), higher percentage (30%-50%) of uniformly generated mutants seems to be more efficient. As the second type of mutation increases exploration, making the mutation steps longer, it is a quite natural result.

5 Comparison with other solutions

The performance of HIA was compared with other immune algorithms. As benchmarking solutions we have used opt-aiNet, BCA, and CoAIN algorithms, introduced in the first section of this paper. The parameters used for HIA were the same for all the functions except for the suppression threshold parameter σ , which was different for functions f4, f5, and f9 (Table 5). Parameters for the other algorithms are given by Timmis *et al.* (2004). Values of the function minima are given in Table 7. As can be seen from the results included in the Table 6 HIA algorithm outperforms the other ones in terms of the number of evaluations and is equal to the others in terms of the quality of solutions. Worth noticing is a fact, that the numbers of function evaluations are not the best achievements of HIA and they can be improved by better parameters tuning for each function. In spite of this, for all benchmarking functions HIA compared against opt-aiNet, BCA, and CoAIN consistently employs fewer evaluations. The differences in some cases are dramatically high, for example for f7 function the HIA result is about 40 times better than that for BCA. What is of noticeable interest is the large standard deviation in the number of evaluations for each technique. It is rather not a surprising effect, taking into consideration that AIS algorithms are stochastic in nature.

6 Conclusions

This paper has presented a novel hybrid algorithm HIA, utilizing immune metaphor and heuristic limitation of search space, in order to solve optimization problem for multimodal functions. HIA has also introduced mixed mutation, using both Gaussian and uniform distributions. It was theoretically compared with opt-aiNet algorithm and experimentally also with BCA and CoAIN. The performance was

TABLE 3: Effect of suppression threshold on number of function evaluations. Values of the other parameters as in Table 5. The first figure indicates number of function evaluations averaged for 1000 experiments and the second one standard deviation.

σ	0.00002	0.0002	0.02	0.2	2	4	5
f1	52	52	59	76	201	-	-
	37	37	37	60	173	-	-
f2	153	152	189	259	282	-	-
	130	122	183	220	226	-	-
f3	178	174	177	185	433	-	-
	136	117	132	116	250	-	-
f4	-	-	-	24866	23735	-	25951
	-	-	-	36241	61053	-	70691
f5	-	-	-	17826	13924	-	17061
	-	-	-	28453	31670	-	44879
f6	174	171	177	184	267	-	-
	172	150	152	156	127	-	-
f7	389	364	555	991	1303	-	-
	377	231	434	1116	3437	-	-
f8	267	263	311	441	671	-	-
	177	170	211	354	532	-	-
f9	-	-	-	433727	9460	4793	5020
	-	-	-	736207	14496	5038	6511

TABLE 4: Effect of the second type mutation rate on number of function evaluations. The second column indicates number of clones, values of the other parameters as in Table 5. The first figure indicates number of function evaluations averaged for 1000 experiments and the second one standard deviation.

Id	c	0%	10%	20%	25%	33%	50%	75%	100%
f1	3	406	59	45	44	42	40	42	46
		1247	73	41	38	33	26	25	28
f2	3	92467	143	131	136	150	177	272	471
		321627	189	142	170	181	215	422	616
f3	6	54961	195	169	174	190	215	271	347
		806786	246	124	136	142	171	253	330
f4	7	27426	26525	32734	28187	36373	46268	89003	-
		99538	74735	119927	75966	113340	120158	268054	-
f6	6	53376	240	168	167	151	158	192	239
		201604	348	157	155	103	88	115	168
f7	6	65021	371	350	381	409	574	1605	6346
		259713	275	201	595	473	768	5469	13559

TABLE 5: Algorithm parameters

Parameter	Value
Initial population size	1
Suppression threshold	0.0002 for f1, f2, f3, f6, f7, f8; 2 for f4, f5; 4 for f9
Number of clones generated	7
Percent of the second type of mutation	20%

illustrated for nine functions. The algorithm demonstrated to be capable of finding optima with much smaller number of function evaluations, comparing to the other immune techniques. The promising results of HIA obtained for multimodal func-

TABLE 6: Performance of CoAIN, opt-ai-Net, BCA, and HIA.

Id	Number of function evaluations			
	CoAIN	opt-aiNet	BCA	HIA
f1	78 ± 160	6717 ± 538	3016 ± 2252	51 ± 25
f2	528 ± 445	41419 ± 25594	1219 ± 767	154 ± 108
f3	1347 ± 652	6346 ± 4656	4921 ± 31587	174 ± 100
f4	233819 ± 148696	363528 ± 248161	46433 ± 31587	15785 ± 19414
f5	168726 ± 122349	346330 ± 255980	426360 ± 32809	13134 ± 19659
f6	1382 ± 567	54703 ± 29701	2862 ± 351	149 ± 112
f7	18636 ± 13538	50875 ± 45530	14654 ± 5277	331 ± 176
f8	2959 ± 1425	14048 ± 13209	-	287 ± 177
f9	10436 ± 5853	40226 ± 36063	-	7069 ± 5699

TABLE 7: Minimum of the functions

Minimum								
f1	f2	f3	f4	f5	f6	f7	f8	f9
-1.12	-12.03	0.40	-186.73	-186.73	-0.35	-186.73	-4.25	-1

tion optimization encourage to use the algorithm for other optimization or even dynamic optimization problems, what will be a subject of the authors' further research.

References

- P. S. ANDREWS and J. TIMMIS (2006), On Diversity and Artificial Immune Systems: Incorporating a Diversity Operator into aiNET, in *Neural Nets*, volume 3931 of *Lecture Notes in Computer Science*, pp. 293–306, Springer.
- V. CUTELLO, G. NARZISI, G. NICOSIA, and M. PAVONE (2005), An Immunological Algorithm for Global Numerical Optimization, in *Artificial Evolution: 7th International Conference, Evolution Artificielle, EA 2005*, volume 3871 of *Lecture Notes in Computer Science*, pp. 284–295.
- L. N. DE CASTRO and J. TIMMIS (2002), An Artificial Immune Network for Multimodal Function Optimization, in *Proc. of the IEEE Congress on Evolutionary Computation*, volume 1, pp. 699–674, IEEE Press, Piscataway, NJ.
- L. N. DE CASTRO and F. J. VON ZUBEN (2001), aiNET: An Artificial Immune Network for Data Analysis, in H. ABBAS, R. SARKER, and C. NEWTON, editors, *Data Mining: A Heuristic Approach*, pp. 231–259, Idea Group Publishing.
- L. N. DE CASTRO and F. J. VON ZUBEN (2002), Learning and Optimization Using

- the Clonal Selection Principle, *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.
- J. KELSEY and J. TIMMIS (2003), Immune Inspired Somatic Contiguous Hypermutation, in *Genetic and Evolutionary Computation Conference, GECCO 2003*, volume 2723 of *Lecture Notes in Computer Science*, pp. 13–28, Springer.
- L. LIU and W. XU (2008), A Cooperative Artificial Immune Network with Particle Swarm Behavior for Multimodal Function Optimization, *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.
- J. TIMMIS, EDMONDS, and J. C. KELSEY (2004), Assessing the Performance of Two Immune Inspired Algorithms and a Hybrid Genetic Algorithm for Function Optimisation, in *Congres on Evolutionary Computation Conference, CEC 2004*, volume 1 of *Lecture Notes in Computer Science*, pp. 1044–1051, IEE Press.
- J WALKER and S. GARRETT (2003), Dynamic Function Optimisation: Comparing the Performance of Clonal Selection and Evolutionary Strategies, in *International Conference on Artificial Immune Systems, ICARIS 2005*, volume 2787 of *Lecture Notes in Computer Science*, pp. 273–284, Springer.

