Clonal Selection Principle Based Approach to Non-Stationary Optimization Tasks

Krzysztof Trojanowski

Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
e-mail: trojanow@ipipan.waw.pl

1 Introduction

The immune system of mammals includes a set of defence mechanisms, and among them a primary and a secondary immune responses are the most sophisticated. The primary one is responsible for recognition and destruction of intruders called *pathogens* which appeared in the organism for the first time. The recognition of pathogens of a previously unknown type is based on a quick adaptation of the system to a new type of patterns. In the case of detection of pathogens recognized as dangerous for the mammal the immune system has to build new detectors which will be able to find and eliminate intruders as soon as possible. The secondary response is based on memory and the ability to remember previously recognized pathogens and this way it reacts much faster than the primary one. Both mechanisms complement each other making an efficient tool for recognition and for the elimination of the different microbes, viruses, etc.

The process of adaptation of cells of the immune system (i.e. white blood cells called lymphocytes) to the new patterns is called clonal selection, and it is based on the reaction of the lymphocytes which are generated in the bone marrow (thus called B-cells) to the matching pathogens [10]. The B-cell matches the pathogens with its receptors called antibodies. On the surface of the B-cell there are hundreds of thousands of such receptors. When an intruder appears in the organism, detectors of the system (and among them B-cells) are stimulated to defence. Those cells which match the pathogens bind to them and eliminate them from the organism. The B-cells which are the most complementary to the matched pathogens are *activated* i.e. they are cloned and undergo hypermutation to create new receptors better able to match the pathogens. This way they explore the space of pattern shapes. The number of clones is directly proportional to the B-cell matching specificity while the strength of mutation is inversely proportional.

The clonal selection paradigm briefly described above was an inspiration for a set of heuristic approaches to optimization since the end of 90's [10]. Adaptation of the pattern recognition mechanizm to optimization task is simple. Instead of an explicit population of antigens to be recognized an evaluation function is used to evaluate antigens' matching specificity. The better fitness value is assigned to the antigen, the higher is its matching. The remaining rules of the iterated algorithms stay unchanged. There is a reference set of antibodies called population. Antibodies with high matching are activated i.e. generate

clones. The clones undergo mutation. After mutation those clones where fitness is better than their predecessors, enter the reference set and the process continues.

In our research we investigate the use of the clonal selection principle based approaches for non-stationary optimization tasks. Selected approaches proved to be efficient in stationary optimization however it is interesting what efficiency could be expected in case of non-stationary one. We did our tests using two known test-case generators.

2 Compared algorithms

In this paper we study properties of four algorithms based on the clonal selection principle: Artificial Immune Iterated Algorithm (AIIA) [9], two versions of B-Cell Algorithm (BCA) [6] and a Simple Artificial Immune System (Sais) [5]. All of them implement non-deterministic iterated process of search and all of them work with a population of solutions called antibodies or B-cells. Antibodies represent candidate solutions to the problem, i.e. vectors of coordinates of points in an n-dimensional real valued search space. Entries of the vectors can be coded as bit strings or stored as real values. Every algorithm starts with a population of randomly generated solutions from the search space and performs the process of iterated improvement of the solutions by the execution of the main loop of the algorithm.

We also assume that the optimization system "knows" when its environment has changed. However none of the algorithms starts from scratch but each reevaluates its population of solutions and continues the search process.

2.1 Artificial Immune Iterated Algorithm (AIIA)

The detailed description of AIIA can be found in [9]. The pseudo-code of the main loop of AIIA is given in Figure 1. The symbol $\mathbf{x}_i$ represents i-th antibody, $\mathbf{xc}_i(k)$ – k-th mutated clone of the i-th antibody, $f(\mathbf{x}_i)$ – fitness to the antigen of the i-th antibody, and $\mathbf{xc}_i^*$ is the best mutated clone of the i-th antibody, i.e. $f(\mathbf{xc}_i^*) \geq f(\mathbf{xc}_i(k)), \bigvee k \in \{1, \ldots c\}$ where c is the number of clones.

1. *Fitness evaluation.* For each antibody $\mathbf{x}_i$ in the population P
   compute its fitness i.e. the value of the objective function $f(\mathbf{x}_i)$.
2. *Clonal selection.* Choose n antibodies with highest fitness to the antigen.
3. *Somatic hypermutation.*
   Make c mutated clones $\mathbf{xc}_j$ for each antibody $\mathbf{x}_j$, $j \in \{1, \ldots n\}$.
   The mutated clone $\mathbf{xc}_j^*$ with highest fitness replaces the original
   antibody if $f(\mathbf{xc}_j^*) > f(\mathbf{x}_j)$.
4. *Apoptosis.* Replace d weakest antibodies by randomly generated solutions.

Figure 1. Pseudo-code of the main loop of AIIA

AIIA has five control parameters: $|P|$ – population size, n – size of the subpopulation activated for *clonal selection* procedure, c – number of mutated clones of the antibody (in general the number of clones for each of activated antibodies could be different. However in experiments presented below we simplified this rule and the number of clones of each

antibody was the same), d – size of the subpopulation that undergo *apoptosis* procedure, and r_m – mutation range.

The solutions are represented as real valued vectors. Applied mutation operator originates from [4] where it was a component of aiNet optimization algorithm. Mutation of an i-th coordinate of a clone $\mathbf{xc}$ is performed as follows:

$$xc_i' = xc_i + \alpha N(0,1),$$
$$\alpha = (r_m/\beta) \cdot (domain\_width/2) \cdot \exp(-f'(\mathbf{x})).$$

where: xc_i – current value of the i-th coordinate of the solution $\mathbf{xc}$, xc_i' – new value of the i-th coordinate, $N(0,1)$ – Gaussian random variable of zero mean and $\sigma = 1$, r_m – mutation range ($0 < r_m \leq 1$), β – a weight factor which is set to 1 for all experiments, and $f'(\mathbf{x})$ is the fitness of a clone's predecessor $\mathbf{x}$ normalized in an interval [0,1] as follows:

$$f'(\mathbf{x}) = \frac{f(\mathbf{x}) - f_{min}}{(f_{max} - f_{min})},$$
$$f_{max} \geq f(\mathbf{x}_k), \bigvee k \in \{1, \ldots |P|\} \text{ and } f_{min} \leq f(\mathbf{x}_k), \bigvee k \in \{1, \ldots |P|\}.$$

2.2 A Simple Artificial Immune System (SAIS)

Outline of the Sais algorithm implemented for our experiments comes from [5]. The pseudo-code of the main loop of Sais is given in Figure 2.

1. *Fitness evaluation.* For each antibody $\mathbf{x}_i$ in the population P compute its exogenic activation i.e. the value of the objective function $f(\mathbf{x}_i)$.
2. Copy the antibodies from P into two reference sets: P_{ex} (i.e. a set of n antibodies with highest fitness value) and P_{en} (all the left antibodies not present in P_{ex}). For all the antibodies in P_{en} compute the endogenic activation.
3. *Clonal selection and somatic hypermutation.*
   3.1 Make c mutated clones $\mathbf{xc}_j$ for each antibody $\mathbf{x}_j$ from P_{ex}, $j \in \{1, \ldots n\}$. The mutated clone $\mathbf{xc}_j^*$ with highest fitness replaces the original antibody in P_{ex} if $f(\mathbf{xc}_j^*) > f(\mathbf{x}_j)$.
   3.2 Do selection and replication of antibodies in P_{en}.
4. *Recruitment.* A new population P is created as a result of tournament selection between current P and both P_{ex} and P_{en} .

Figure 2. Pseudo-code of the main loop of Sais

Sais has four control parameters: $|P|$ – population size, n – number of exo-activated B-cells, c – number of mutated clones of i-th B-cell and r_m – mutation range.

In our implementation of Sais, solutions are represented as real valued vectors unlike the original version described in [5]. Thus it was necessary to do some adaptations in the algorithm. We needed especially to redefine the mutation operator (which was the same as for AIIA) and specify the rules of selection and replication applied to the population of endo-activated B-cells.

The proces of management of exo-activated antibodies is clearly described in the Figure 2. However some additional explanations are needed for understanding the process

of management of endo-activated individuals. The level of endo-activation of a B-cell is evaluated as follows:

$$i^{en} = 1/n_P - d_i + 1,$$

where: n_P – number of different B-cells in the population; d_i – density of an i-th B-cell (a parameter assigned to each B-cell and evaluated every iteration).

The endo-activation level i^{en} depends on the density of i-th B-cell $\mathbf{x}_i$. This density is proportional to the number of other B-cells located not farther than k from $\mathbf{x}_i$, where k is an experimentally tuned parameter. In the course of *clonal selection and somatic hypermutation* a set of clones is generated from P_{en} with a tournament selection scheme. The clones do not undergo any mutation in this step and completely replace the population P_{en}. In the step of *recruitment* a new population P is created with tournament selection: antibodies from P_{ex} compete with their parents from P and respectively antibodies from P_{en} also compete with their parents.

2.3 B-Cell Algorithm (BCA)

Our version of the BCA algorithm originates from [6]. The pseudo-code of the main loop of BCA is given in Figure 3.

1. For each B-Cell $\mathbf{x}_i$ in the population P compute its fitness i.e. the value of the objective function $f(\mathbf{x}_i)$.
2. For each B-Cell $\mathbf{x}_i$ do
   - 2.1 Make c mutated clones $\mathbf{xc}_i$ and place in clonal pool.
   - 2.2 Randomize one clone in the clonal pool.
   - 2.3 Apply contiguous mutation to all the remaining clones.
   - 2.4 For each clone in the clonal pool compute its fitness.
   - 2.5 The mutated clone $\mathbf{xc}_i^*$ with highest fitness replaces the original B-Cell if $f(\mathbf{xc}_i^*) > f(\mathbf{x}_i)$.

Figure 3. Pseudo-code of the main loop of BCA

The algorithm has three control parameters: $|P|$ – population size, c – number of clones in the clonal pool, and n_b – number of bits for each of the coordinates because a binary representation of solution is applied here. Two versions of BCA with two types of binary coding were tested in our experiments: a 32-bit pure binary coding and a 64-bit double precision floating point coding (according to IEEE 754 standard). Contiguous mutation was implemented as described in [6] and applied to both versions.

3 How to compare the algorithms

Because of different rules of population management in the tested algorithms it was impossible to compare them just by setting the same values of common parameters. Instead, according to suggestions in [1] we decided to consider the algorithms as comparable when the number of fitness function evaluations between subsequent changes in the environment is similar for each of the algorithms. For comparisons with results published by other authors the number of evaluations equals approx. 5000. In our experiments we varied some of algorithms' parameters, however for each of parameters' settings the

constraint on maximum number of iterations was always satisfied. Full list of values of algorithms' fixed and tuned parameters is given in Table 1.

Table 1. Full list of values of algorithms' fixed and tuned parameters

Algorithm	fixed parameters	tuned parameters
AIIA	$\|P\| = 50$, $d = 30$	n, c, r_m
Sais	$\|P\| = 50$, $k = 0.05 \cdot (\textit{domain width})$	n, c, r_m
BCA	$n_b = 32, 64$	$\|P\|$, c

4 Plan of experiments and applied measures

Behavior of the algorithms was tested with six environments generated with two test-benchmarks. The first test-benchmark is a Test Case Generator (or TCG) proposed in [8][1]. We created four testing environments with TCG; two of them with cyclic changes and two with non-cyclic ones. In case of cyclic changes a single run includes 5 cycles of changes in the environment. In case of non-cyclic environments the total number of changes for each of them was 25. The second test-benchmark is a Moving Peaks Benchmark (or MPB) generator [2, 7]. Its description, sample parameters settings and a source code are available at the web page [1]. We created two testing environments with MPB called scenario 1 and 2 [1]. A six groups of experiments were performed: two of them with cyclically changing environments and four others with non-cyclically changing environments. Table 2 shows the settings of each of the groups.

Table 2. Parameters of groups of experiments with cyclically changing environments, i.e. TCG_{10c} and TCG_{20c} and with non-cyclically ones, i.e. TCG_{12nc}, TCG_{20nc}, MPB_5 and MPB_{50}

Environment	TCG_{10c}	TCG_{20c}	TCG_{12nc}	TCG_{20nc}	MPB_5	MPB_{50}
No. of iterations	1000	2000	500	500	500	500
Environment type	10×10	10×10	6×6	10×10	1	2
No. of varying optima	10	20	12	20	5	50

The first row called *Environment* shows the identifier of the testing environment. *No. of iterations* shows the number of iterations of the search process performed in a single experiment, i.e. in one run of the optimization algorithm. In case of cyclic changes this number is equal to: number of cycles of changes multiplied by the number of changes in a cycle (i.e. no. of varying optima) and by the number of iterations between changes, For example, for environment TCG_{10c} it is: 5×10×20. *Environment type* shows the size of the environment generated with TCG (i.e. the number of subspaces in the search space)

[1] Figures of sample environments generated with TCG are available at: http://www.ipipan.waw.pl/~stw/ais/environment/env.html

or the number of scenario in case of MPB. *No. of varying optima* shows the number of varying hills, peaks or cones.

To evaluate the results, we used a measure called offline error which represents the average deviation of the best individual from the optimum evaluated since the last change of the fitness landscape. Every time the solution's fitness is evaluated, an auxiliary variable is increased by the value which is the deviation of actually evaluated solution if the fitness is better than any other since the last change or the deviation of the best one since the last change otherwise. When the experiment is finished the sum in the variable is divided by the total number of evaluation and returned as the offline error. Every experiment was repeated 100 times and the mean is presented in the Figures.

5 Results of experiments

Figure 4 presents results of tuning for BCA with both types of coding, Figure 5 – for AIIA, and Figure 6 – for Sais.

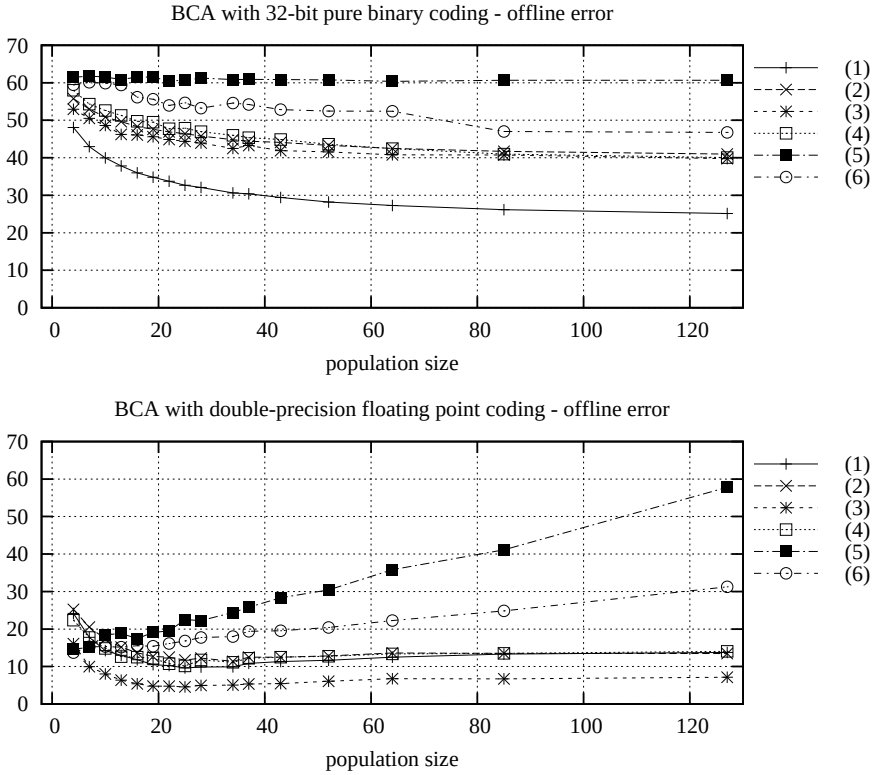

Figure 4. Offline error obtained during experiments with BCA with pure binary coding (BCA_{bc}) and with double precision floating point coding (BCA_{fpc}) for six testing environments. X axis represents population size

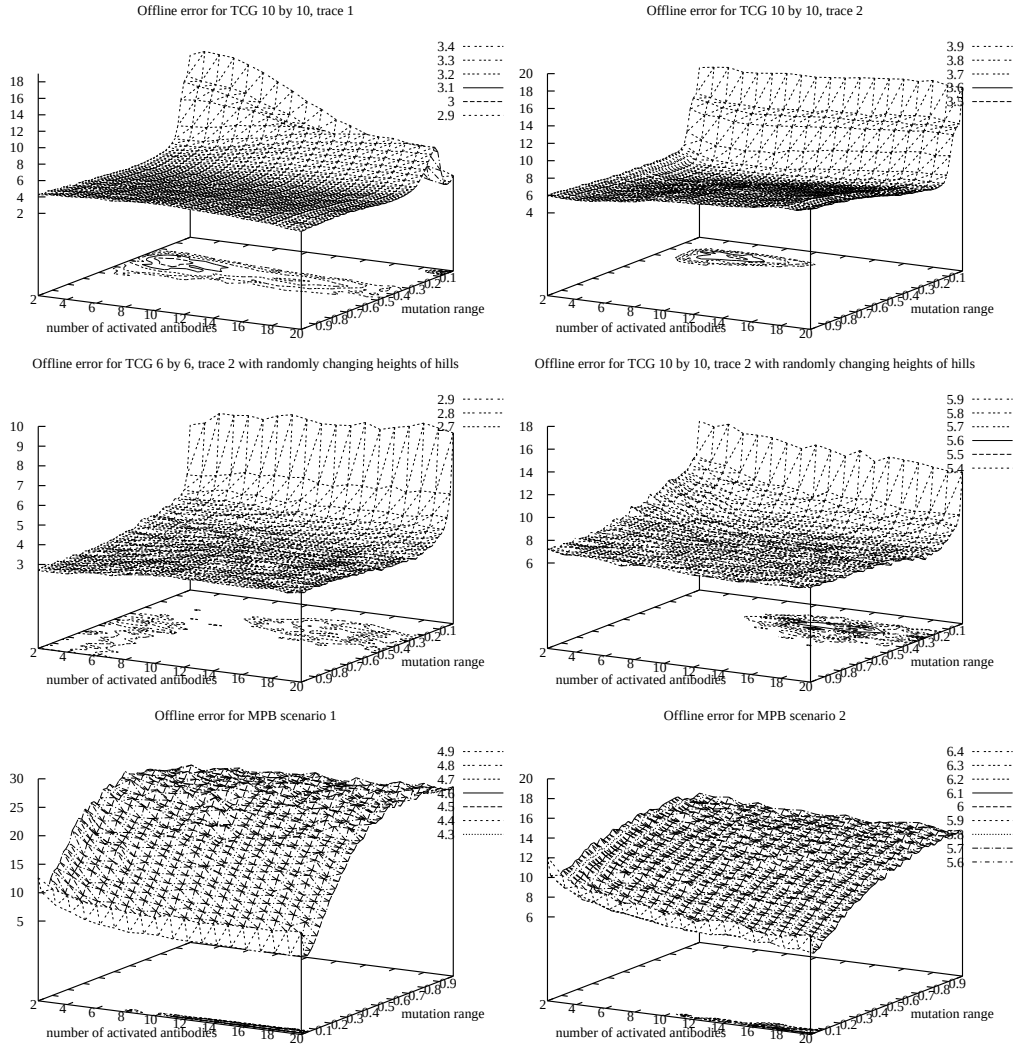

Figure 5. Offline error obtained during experiments with AIIA: mutation range vs. number of activated antibodies

For easier comparison of efficiency of tested algorithms we present Table 3 where the best mean values of offline error obtained during experiments are presented.

6 Results summary and conclusions

In spite of the fact that all the algorithms originate from the same clonal selection principle the results of tuning show presence of differences in the properties of the three algorithms. In case of cyclic changes (environments TCG_{10c} and TCG_{20c}) for AIIA the number of activated antibodies (n) should be rather small for better performance

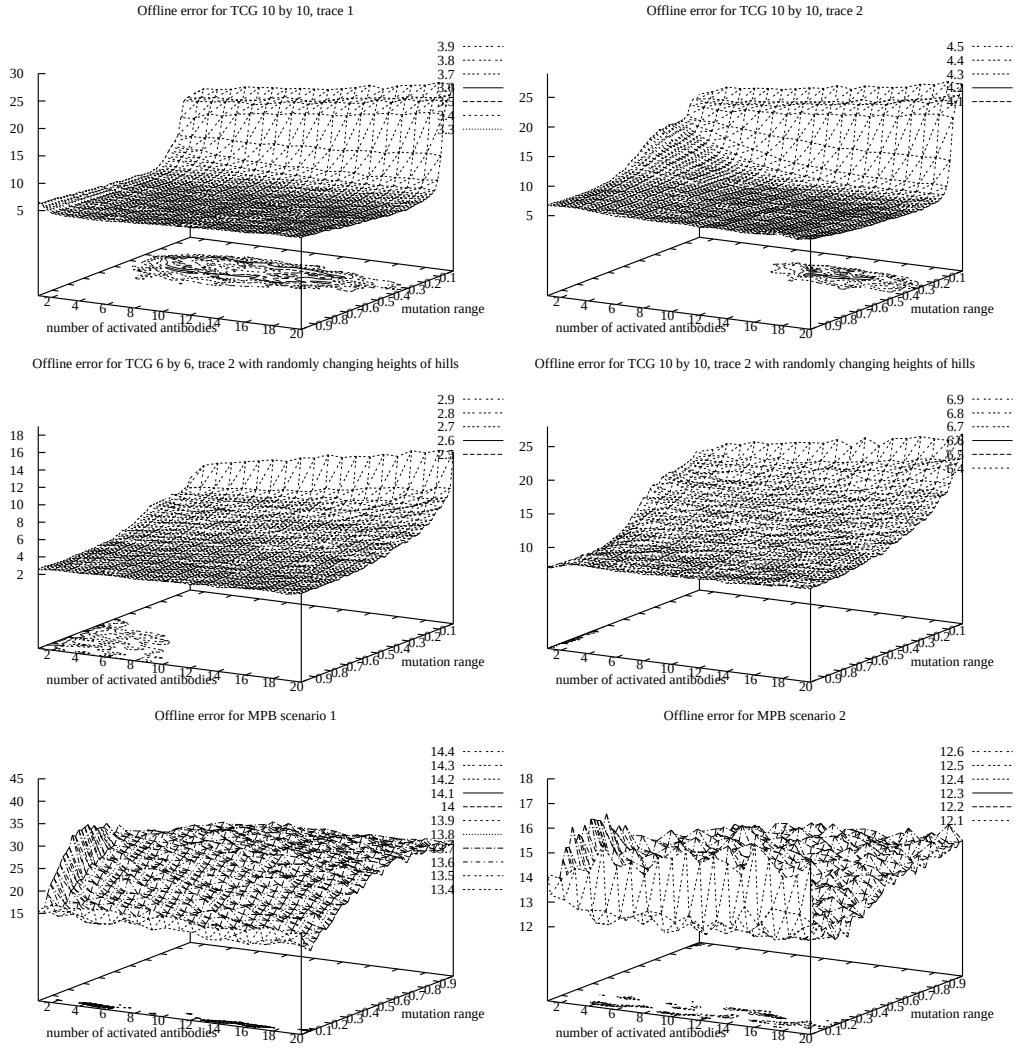

Figure 6. Offline error obtained during experiments with Sais with double precision floating point representation of solution coordinates: mutation range vs. number of activated antibodies

(admittedly for TCG_{10c} the best result was obtained for $n = 20$ and $r_m = 0.05$, whereas the second best was for $n = 4$ and $r_m = 0.35$; for TCG_{20c} – the best result was obtained for $n = 7$ and $r_m = 0.21$). For Sais the rule is not so clear because for TCG_{10c} the best result was obtained for $n = 7$ and $r_m = 0.39$, but for TCG_{20c} – the best result was obtained for $n = 14$ and $r_m = 0.23$. For BCA_{fpc} the optimal number of activated antibodies for all the testing environments is rather big (the best values are obtained for small population sizes) or represents even the maximal possible size in some cases.

The mutation range plays significant role in all the algorithms. It should be tuned respectively to the number and width of changing hills in the landscape for cyclic changes

Table 3. Best values of offline error obtained by the algorithms

Environment	TCG_{10c}	TCG_{20c}	TCG_{12nc}	TCG_{20nc}	MPB_5	MPB_{50}
AIIA	2.83	3.45	2.62	5.32	4.29	5.60
Sais	3.24	4.03	2.45	6.32	13.31	12.07
BCA_{bc}	25.13	40.99	39.74	40.07	60.39	46.74
BCA_{fpc}	9.63	11.36	4.55	10.17	14.65	13.72

and for non-cyclic changes as well. For both algorithms with real valued representation of solution coordinates the best results for cyclic changes were obtained when mutation range r_m was between 0.2 and 0.4. The two environments with non-cyclic changes based on TCG also required mutation range of similar values for AIIA (best results obtained for $r_m = 0.25$ in case of TCG_{12nc} and for $r_m = 0.31$ in case of TCG_{20nc}) but much wider mutation range for Sais ($r_m = 0.83$ for both environments). The situation is just opposite in case of the two environments based on MPB. The requested mutation range should be very small (for AIIA: $r_m = 0.05$ and $r_m = 0.03$ respectively and for Sais: $r_m = 0.07$ for both environments). It is interesting that in case of Sais there are two optimal configurations of parameters settings for MPB_{50}. One of them is with $r_m = 0.07$ and another one – with $r_m = 0.95$ both with small number of activated antibodies and large number of clones.

The best performance was achieved by the AIIA algorithm which returned the smallest value of the offline error for all the tested environments except one, i.e. the TCG_{12nc} test case where it was minimally outperformed by Sais. The worst results with non-stationary optimization were obtained by BCA, however the results obtained with an original 64-bit double precision floating point coding were much better than those obtained with a 32-bit pure binary coding.

Finally it is worth noting that the outcome of the clonal selection based approaches is comparable with the results obtained by other evolutionary approaches even supported by mechanisms dedicated for non-stationary optimization, like random immigrants or hypermutation, as well as by multi-objective optimization methods (for comparison see e.g. the best value of offline error obtained for MPB_{50} in [3]: the best value equals 8.87. For traditional GA – 11.05. In our experiments presented in Table 3 above the AIIA algorithm reached 5.60). This encourages us to study properties of other clonal selection based algorithms not mentioned in this paper. A subject of our further research could be searching for those common properties which are the most responsible for effective optimization in non-stationary optimization tasks.

7 Acknowledgements

The author would like to thank the anonymous reviewer for valuable comments.

Bibliography

[1] J. Branke. The moving peaks benchmark. URL: http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/movpeaks/.

[2] J. Branke. Memory enhanced evolutionary algorithm for changing optimization problems. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1875–1882. IEEE Press, Piscataway, NJ, 1999.

[3] L.T. Bui, J. Branke, and A. Abbass. Multiobjective optimization for dynamics environments. Technical Report TR-ALAR-200504007, ALAR Technical Report Series, 2005.

[4] L.N. de Castro and J. Timmis. An artificial immune network for multimodal function optimization. In *Proc. of the Congress on Evolutionary Computation*, volume 1, pages 699–674. IEEE Press, Piscataway, NJ, 2002.

[5] A. Gaspar and P. Collard. From GAs to artificial immune systems: Improving adaptation in time dependent optimization. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1859–1866. IEEE Press, Piscataway, NJ, 1999.

[6] J. Kelsey and J. Timmis. Immune inspired somatic contiguous hypermutation for function optimisation. In *Genetic and Evolutionary Computation Conference – GECCO 2003*, LNCS 2723, pages 207–218. Springer Verlag, 2003.

[7] R. W. Morrison and K. A. De Jong. A test problem generator for non-stationary environments. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1859–1866. IEEE Press, Piscataway, NJ, 1999.

[8] K. Trojanowski and Z. Michalewicz. Searching for optima in non–stationary environments. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proc. of the Congress on Evolutionary Computation*, volume 3, pages 1843–1850. IEEE Press, Piscataway, NJ, 1999.

[9] K. Trojanowski and S. T. Wierzchoń. Studying properties of multipopulation heuristic approach to non-stationary optimisation tasks. In M. A. Kłopotek, S. T. Wierzchoń, and K. Trojanowski, editors, *IIS 2003:Intelligent Information Processing and Web Mining*, Advances in Soft Computing, pages 23–32. Springer Verlag, 2003.

[10] S.T. Wierzchoń. Function optimization by the immune metaphor. *Task Quarterly*, 6(3):493–508, 2002.