Control of Immune Memory in Artificial Immune System

Krzysztof Trojanowski[1)], Sławomir T. Wierzchoń[1,2)]

[1)] Institute of Computer Science, Polish Academy of Sciences
01-237 Warszawa, ul. Ordona 21
e-mail: {trojanow,stw}@ipipan.waw.pl

[2)] Department of Computer Science, Białystok Technical University
15-351 Białystok, ul. Wiejska 45[a]

Introduction

As stated by [Wilke99], evolutionary scenarios – studied to understand different phenomena of living systems – can be divided into three large classes. When looking at a system from a single species standpoint, we can find that the species evolve (i) in a constant environment, (ii) in a changing environment whose properties cannot be altered by the species, or (iii) in a changing environment that reacts to feedback from the species. By species we mean a population of individual entities subjected to mutation and selection. Thus the world of species covers not only plants or animals but also self-replicating molecules, pure information carriers such as computer programs or populations of bitstrings used in artificial chemistry [Dittrich00], genetic algorithms, or artificial immune systems [Wierzchoń01]. The first of the three classes mentioned above is most-intensively studied: let us mention works on population genetics, e.g. [Gale90], quasispecies, [Eigen71], or standard genetic algorithms. Surprisingly, class (iii) has also been studied by many authors: let us mention ecological investigations, e.g. [Freedman80] or interspecies interactions analysed by the artificial life community, consult e.g. [Adami98].

Concerning the second class, we should mention [Wilke99] where Eigen's quasispecies model in a variable environment was intensively studied. Interestingly, the model proposed by Eigen in 1971 (and concerned with infinite populations) was one of the first exact mathematical studies of the interplay of mutation and selection on a molecular level. The recent analysis of genetic algorithm, [Nimwegen97] although dealing with finite populations has its roots in the quasispecies model. An alternative approach refers to the idea of immune system, consult [Wierzchoń01]. It was tested in the papers [TrojanowskiWierzchon02] and [TrojanowskiWierzchon02a], where an interested reader can find the algorithm and relevant literature. Our aim – inspired by the earlier paper [Wierzchoń01a] – was to trace the immune memory emergence and its usefulness in optimising. In [Trojanowski, Wierzchoń02a] it was shown, that the presence of memory can improve efficiency of the artificial immune system in searching for optima of non-stationary environments, however the main problem was a form of memory control, i.e. memory structure, and rules of remembering, reminding and forgetting. It should be mentioned here, that the idea of employing memory mechanisms in heuristic optimisation (but in the other heuristic than the immune system, i.e. in evolutionary algorithms) was already studied by Trojanowski and Michalewicz in e.g. [Trojanowski,Michalewicz99]. Memory proved its positive influence on the effectiveness of evolutionary optimisation in non-stationary environments. It introduces into the algorithm a primitive form of learning, which comes to be helpful especially in case of cyclically

changing environments. In this paper our goal was to find memory control rules, which would be appropriate for the immune system.

In this study we compare two methods of memory control. In both methods there exists a *memory buffer* in artificial immune system. The buffer is one and only one, it is common for all individuals in the system and may contain many individuals obtained during the search process. The size of the memory buffer is constant during the time of search.

Experiments

The two already mentioned methods have different population architectures. In the first one there is one population which exchanges information with memory buffer, and in the second one, there are two populations, which can write to memory but only one of them can read.

Single-population method

Here, the system consists of a population of antibodies, which perform search process over the search space and a memory buffer storing complete individuals remembered during the search. The population is initialised randomly once at the beginning of the search and continues its search after every change in optimised environment without any significant modification of a group of individuals. The only modification performed after change (which represents presentation of another antigen to the immune system) is concerned with exchange of a single individual with the memory buffer.

Double-population method

In this approach there are two population of antibodies: the first one, called *population of explorers*, represents primary response of an immune system, and the second one, called *exploitative population*, represents secondary response of the system. There is also a memory buffer, which represents memory cells of the system. This architecture of populations corresponds to the idea proposed by Branke in [Branke99].

After every change in the optimised environment, both populations start their search processes in parallel at the same time. The difference is in starting points of these populations. Population of explorers starts with randomly generated and therefore uniformly distributed over the search space population of individuals. Exploitative population takes the best individual form the memory (here "the best" means that all individuals from the memory buffer are re-evaluated with evaluation function after change, and then the best one is selected), and builds its population using this individual as a "seed". All individuals in exploitative population are created by light mutation of the seed. Then the search process starts. During the search process populations do not disturb each other, i.e. do not exchange any information. They just compete in the race to the optimum. Returned value of this searching pair is the better individual of the best individuals of the two populations.

Memory control

We tested two forms of memory control: for single-population architecture of a system and for double-population architecture. For both of forms, three mechanisms concerned with memory access are discussed: remembering, forgetting and recalling.

1) Memory control for a single-population architecture

a) Rules of remembering - at the first iteration of the search process memory buffer is empty. Then, at the end of every iteration the best individual in the population, which was also the best in previous iteration, is added to the memory buffer. Note that it is not enough to be the best individual just once, but an individual has to be the best in two iterations one by one, to be remembered. What is remembered (i.e. the content of memory buffer) increases as the generation number increases.
b) Rules of forgetting - when the buffer is full, the solution with minimal affinity value is deleted to make room for a new one.
c) Rules of recalling - memory is recalled every time the change appears in the environment. This is the time when all the individuals in the current population are re-evaluated (to take into account the effect of changes). Then, if any of the remembered individuals is better than the best individual in the population, then it will be replaced with the current one.

2) Memory control for a double-population architecture

a) Rules of remembering - at the first iteration of the search process memory buffer is empty. Then, after every change in the testing environment, the better individual of the best individuals of the two populations: population of explorers and exploitative population are added to the memory buffer. Thus, what is remembered (i.e. the content of memory buffer) increases as the generation number increases.
b) Rules of forgetting - when the buffer is full, the solution with minimal affinity value is deleted to make room for a new one.
c) Rules of recalling - memory is recalled every time the change appears in the environment to initiate the exploitative population. Then, best found individual takes role of a seed for exploitative population.

Testing environment

A set of experiments was performed. We did three groups of experiments with two types of environments. Our test-bed was a test-case generator proposed in [Trojanowski,Michalewicz99]. The generator creates a convex search space, which is a multidimensional hypercube. It is divided into a number of disjoint subspaces of the same size with defined simple unimodal functions of the same shape but possibly different value of optimum. In case of two-dimensional search space we simply have a patchy landscape, i.e. a chess-board with a hill in the middle of every field. Hills do not move but cyclically change their heights what makes the landscape varying in time. The goal is to find the current highest hill. In our experiments there was a sequence of fields with varying hills' heights. Other fields of the space were static. We did experiments with two-dimensional search space where the chess-boards were of size 4 by 4, i.e. with 16 fields, and of size 6 by 6, i.e. with 36 fields. Thus the search spaces consisted of 15 local optima and one global optimum in the first case, and of 35 local optima and one global optimum in the second one. We tested four shapes of the sequence of non-stationary fields presented in Figure 2. In the figure, values in cells are weights of unimodal fuctions of the respective fields, which control heights of the hills. In other words the function located at the (i,j)-field is of the form $f_{ij}(x,y) = w_{ij} \times g(x-a_i, y-b_j)$, where g is a fixed unimodal function and (a_i, b_j) is the central point of this field. Lower index at the value in the cell represents the position of the field in the sequence of presented optima. The environments #1 and #3 (left part of Figure 2) were test-beds for experiments with cyclic

changes, while the environments #2 and #4 (right part of Figure 2) were test-beds for experiments with both cyclic and acyclic changes. The aim of these experiments was to trace efficiency of primary (acyclic changes) and secondary (cyclic changes) immune response to the antigens (i.e. current optima). For experiments with cyclic changes a single epoch obeys 5 cycles of changes. In all the experiments each antigen has been presented through 10 iterations. Thus, in case of the environment #1 a single epoch took 200 iterations, in case of the environment #2 - 400 iterations, in case of the environment #3 - 300 iterations, and in case of the environment #4 - 600 iterations. Experiments with non-cyclic changes were based on the environments #2 and #4 and a single epoch included just one cycle of changes and took 80 and 120 iterations respectively.

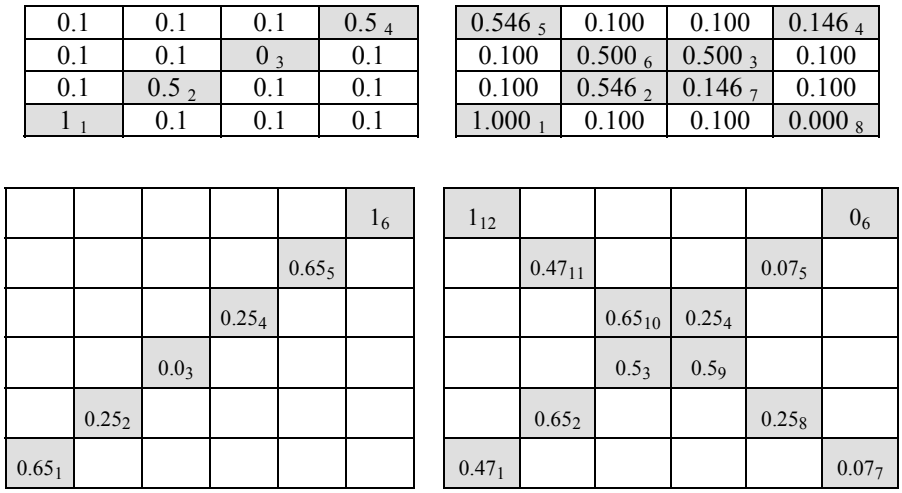


Figure 1. Environments #1 (top left), #2 (top right), #3 (bottom left) and #4 (bottom right) - shapes of the sequence of non-stationary fields in testing environments.

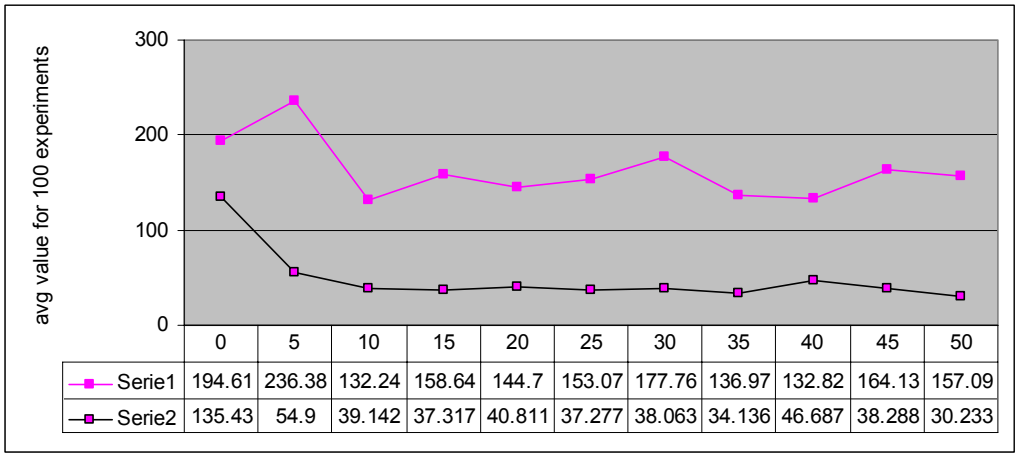
For the six environments described above we did series of experiments with external memory buffer of different sizes - from zero to 50. Every experiment was repeated through 100 epochs and in the later figures we always study average values of these 100 epochs. For the results estimation we used a measure proposed in [Trojanowski,Michalewicz99]: Accuracy. Accuracy is a difference between the value of the current best individual in the population of the "just before the change" generation and the optimum value averaged over the entire run. For this measure the smaller values are the better results.

Obtained results

Obtained values of Accuracy are presented in the figures. In every figure we compare two graphs for the two method described above.

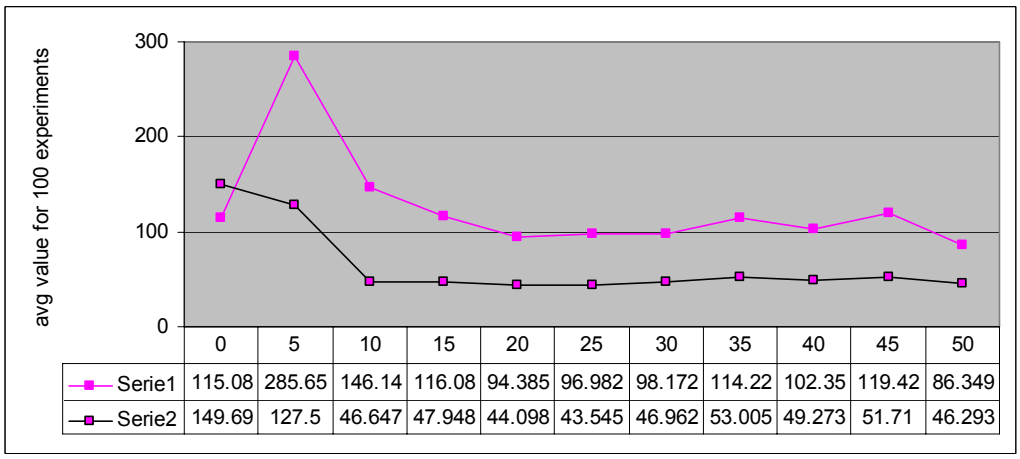
Cyclic changes

The results of cyclic changes are presented in Figure 2 (env. #1) Figure 3 (env. #2), Figure 4 (env. #3), and Figure 5 (env. #4).



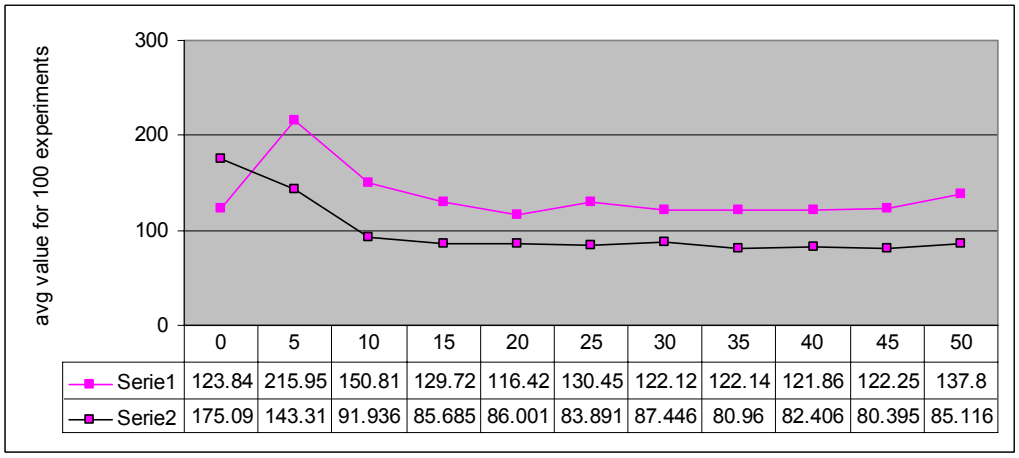
	0	5	10	15	20	25	30	35	40	45	50
Serie1	194.61	236.38	132.24	158.64	144.7	153.07	177.76	136.97	132.82	164.13	157.09
Serie2	135.43	54.9	39.142	37.317	40.811	37.277	38.063	34.136	46.687	38.288	30.233

Figure 2. Results for experiments with 11 sizes of memory buffer (from 0 to 50) performed with two methods: single-population (Serie 1) and double-population (Serie 2) approach for environment #1 (cyclic changes).



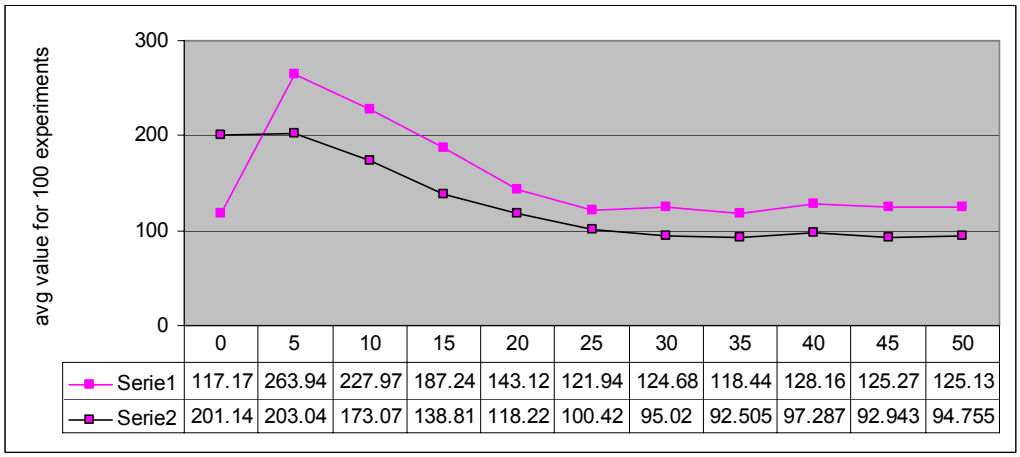
	0	5	10	15	20	25	30	35	40	45	50
Serie1	115.08	285.65	146.14	116.08	94.385	96.982	98.172	114.22	102.35	119.42	86.349
Serie2	149.69	127.5	46.647	47.948	44.098	43.545	46.962	53.005	49.273	51.71	46.293

Figure 3. Results for experiments with 11 sizes of memory buffer (from 0 to 50) performed with two methods: single-population (Serie 1) and double-population (Serie 2) approach for environment #2 (cyclic changes).



	0	5	10	15	20	25	30	35	40	45	50
Serie1	123.84	215.95	150.81	129.72	116.42	130.45	122.12	122.14	121.86	122.25	137.8
Serie2	175.09	143.31	91.936	85.685	86.001	83.891	87.446	80.96	82.406	80.395	85.116

Figure 4. Results for experiments with 11 sizes of memory buffer (from 0 to 50) performed with two methods: single-population (Serie 1) and double-population (Serie 2) approach for environment #3 (cyclic changes).

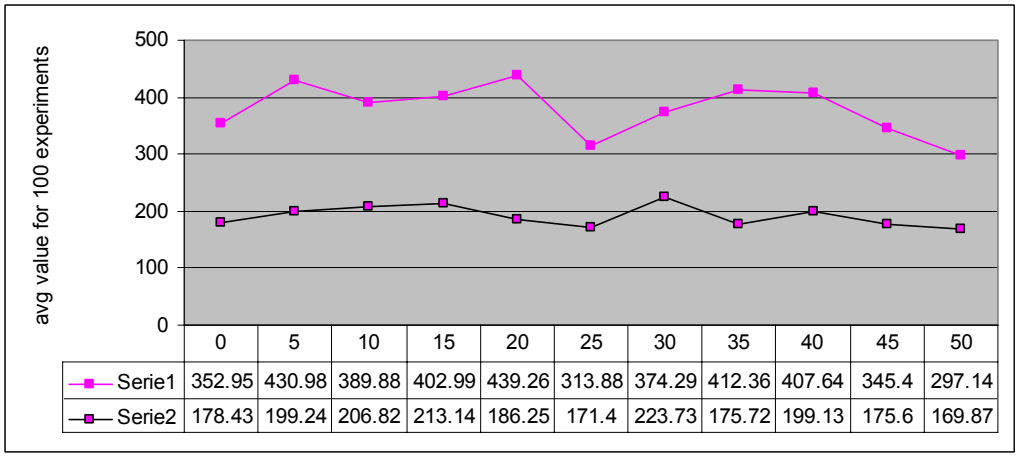


	0	5	10	15	20	25	30	35	40	45	50
Serie1	117.17	263.94	227.97	187.24	143.12	121.94	124.68	118.44	128.16	125.27	125.13
Serie2	201.14	203.04	173.07	138.81	118.22	100.42	95.02	92.505	97.287	92.943	94.755

Figure 5. Results for experiments with 11 sizes of memory buffer (from 0 to 50) performed with two methods: single-population (Serie 1) and double-population (Serie 2) approach for environment #2 (cyclic changes).

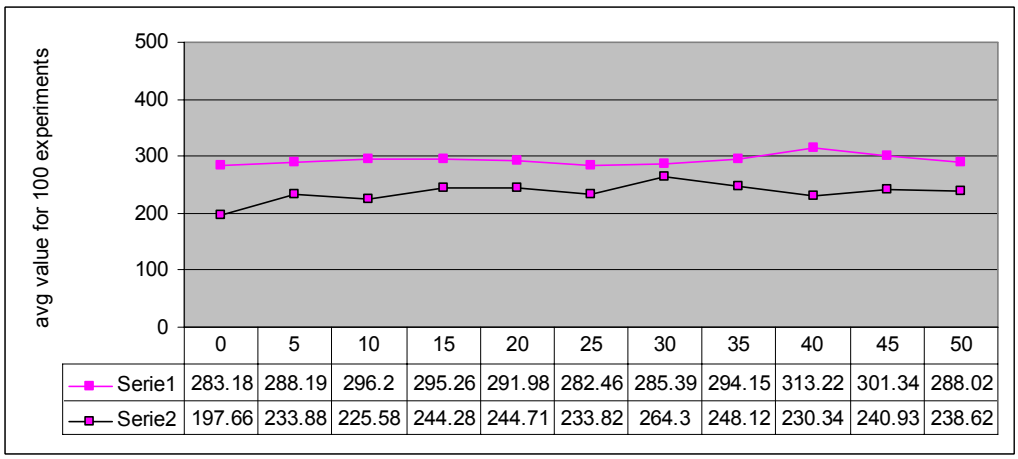
Non-cyclic changes

The results of non-cyclic changes are presented in Figure 6 (env. #2) and Figure 7 (env. #4).



	0	5	10	15	20	25	30	35	40	45	50
Serie1	352.95	430.98	389.88	402.99	439.26	313.88	374.29	412.36	407.64	345.4	297.14
Serie2	178.43	199.24	206.82	213.14	186.25	171.4	223.73	175.72	199.13	175.6	169.87

Figure 6. Results for experiments with 11 sizes of memory buffer (from 0 to 50) performed with two methods: single-population (Serie 1) and double-population (Serie 2) approach for environment #2 (non-cyclic changes).



	0	5	10	15	20	25	30	35	40	45	50
Serie1	283.18	288.19	296.2	295.26	291.98	282.46	285.39	294.15	313.22	301.34	288.02
Serie2	197.66	233.88	225.58	244.28	244.71	233.82	264.3	248.12	230.34	240.93	238.62

Figure 7. Results for experiments with 11 sizes of memory buffer (from 0 to 50) performed with two methods: single-population (Serie 1) and double-population (Serie 2) approach for environment #2 (non-cyclic changes).

Conclusions

Results obtained for the single-population approach applied to cyclically changing environments show, that too small memory buffer can significantly decrease quality of obtained results. With increasing size of memory buffer, we can observe that obtained results are getting better. Additionally, when the memory size exceeds a threshold value, which is different for different environments, the result values stabilise and become as good as the ones of experiments without memory or in some cases even slightly better. However, in general it is easy to come to convinction that memory cannot help much with non-stationary optimisation process.

Completely different results were obtained for experiments with cyclically changing environments and with the double-population approach. We can see that artificial immune system with even small memory buffer is better than the pure system. However, every type of changes in the environment has a kind of *limes* in memory buffer size, such that for algorithms with larger memory buffer, obtained results are almost the same while the computation cost is obviously higher. The value of this *limes*, i.e. optimal memory buffer size is the most probably concerned with a length of a cycle of changes. This conclusion can be confirmed by observation that for environment with a cycle of changes of length 12, the results stopped to be improved for memory buffer of size 30 (Figure 5) while for a cycle of changes of length 6 - for memory buffer of size 15 (Figure 4). However, a shape of a path of changes should be also taken into account in this considerations (compare also Figures 2 and 3).
Experiments with cyclically changing environment showed that the form of memory control plays key role in efficiency of the artificial immune system, and forces a form of architecture of populations in the system.

For experiments with non-cyclically changing environments, both approaches did not improve obtained results too much and sometimes even decreased their quality. Obtained graphs (Figures 6 and 7) show that differences between results for experiments with and without memory are not significant and do not show any dependency between memory buffer size and quality of obtained results.

Finally, we can also see, that in three of four environments with cyclic changes, the better results were obtained when algorithm continued search after change with the same population, as before the change. The results were worse when algorithm every time started from scratch (see Figures 2, 3, 4 and 5 - accuracy values for experiments with memory buffer of size zero).

References

[Adami98] Adami, C. *Introduction to Artificial Life*. Santa Clara: Telos, Springer-Verlag 1998

[Branke99] Branke, J., Memory Enhanced Evolutionary Algorithm for Changing Optimization Problems, Proc. of the 1999 Congress on Evolutionary Computation - CEC'99, IEEE Publishing, pp. 1875-1882

[Dittrich00] Dittrich, P. Zielgler, J., Banzhaf, W. Artificial chemistries. A review. *Artificial Life*, **7** (2001) 225 – 275

[Eigen71] Eigen, M. Selforganization of matter and the evolution of biological macromolecules. *Naturwissenschaften* **58** (1971)465-523

[Freedman80] Freedman, H.I. *Deterministic Mathematical Models in Population Ecology*. New York: Marcel Dekker 1080

[Gale90] Gale, J.S. *Theoretical Population Genetics*. London: Unwin Hyman 1990

[Trojanowski,Michalewicz99] Trojanowski,K., and Michalewicz, Z., Searching for optima in non-stationary environments, Proc. of the 1999 Congress on Evolutionary Computation - CEC'99, IEEE Publishing, pp.1843-1850

[Trojanowski,Wierzchoń02] Trojanowski K., Wierzchoń S. T., Memory Management in Artificial Immune System, presented at International Conference on Neural Nets and Soft Computing, ICNNSC 2002, Zakopane, Poland, proceedings to be printed in Physica-Verlag, Advances in Soft Computing series.

[Trojanowski,Wierzchoń02a] Trojanowski K., Wierzchoń S. T., Searching for Memory in Artificial Immune System, *Intelligent Information Systems 2002*. Proceedings of the IIS'2002 Symposium, Sopot, Poland, June 3-6, 2002. Heidelberg 2002, Physica-Verlag, Advances in Soft Computing.

[Nimwegen97] van Nimwegen, E., Crutchfield, J.P., and Mitchell, M. Statistical dynamics of the royal road genetic algorithm. SFI Working Paper 97-04-035, Santa Fe 1997

[Wierzchoń01] Wierzchoń, S.T. *Artificial Immune Systems. Theory and Applications* (in Polish). Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001. ISBN 83-87674-30-3, 282+vii pp.

[Wierzchoń01a] Wierzchoń, S.T. Algorytmy immunologiczne w działaniu: optymalizacja funkcji niestacjonarnych. XII Ogólnopolskie Konwersatorium nt. *Sztuczna Inteligencja – nowe wyzwania. SzI-16'2001*, Siedlce-Warszawa, 28 listopada 2001, Akademia Podlaska, PAN., WAT, s. 97-106

[Wilke99] Wilke, C.O. Evolutionary dynamics in time-dependent environments. Ph. D. Thesis, Dept. of Physics and Astronomy, Ruhr-Universität Bochum, Bochum 1999