*In David W. Pearson, Nigel C. Steele, Rudolf Albrecht (eds.), Proceedings of the Sixth International Conference on Neural Networks and Genetic Algorithms (ICANNGA'03), pp. 168-174, Roanne, France, 23-25 April, Springer, 2003.*

# An Immune System-Based Genetic Algorithm to Deal with Dynamic Environments: Diversity and Memory

Anabela Simões[1,2], Ernesto Costa[2]
[1] Dept. of Informatics and Systems Engineering, Coimbra Polytechnic, Quinta da Nora, 3030 Coimbra, Portugal
[2] Centre for Informatics and Systems of the Univ. of Coimbra, Pinhal de Marrocos, 3030 Coimbra, Portugal
E-mail: abs@isec.pt; ernesto@dei.uc.pt

**Abstract.** The standard Genetic Algorithm has several limitations when dealing with dynamic environments. The most harmful limitation as to do with the tendency for the large majority of the members of a population to convergence prematurely to a particular region of the search space, making thus difficult for the GA to find other solutions when changes in the environment occur. Several approaches have been tested to overcome this limitation by introducing diversity in the population or through the incorporation of memory in order to help the algorithm when situations of the past can be observed in future situations. In this paper, we propose a GA inspired in the immune system ideas in order to deal with dynamic environments. This algorithm combines the two aspects mentioned above: diversity and memory and we will show that our algorithm is also more adaptable and accurate than the other algorithms proposed in the literature.

## 1 Introduction

When using evolutionary techniques in order to cope with dynamic environments it is necessary to overcome some limitations inherent to traditional evolutionary algorithms. In fact, the long-term success of any biologic evolutionary system is assured by maintaining diversity of individuals within a population. Genetic diversity allows a population to adapt to modifications in the environment. In the classical GA it is difficult to maintain diversity because the algorithm assigns exponentially increasing number of trials to the observed best parts of the search space [9]. Consequently, the GA has strong convergence properties. When dealing with dynamic environments strong convergence can be problematic, because the GA is unable to respond effectively to modifications in the environment. To avoid premature convergence of all individuals of the population towards the optimum, several approaches have been used: hypermutation [1], Random Immigrants [10], new genetic operators [15]. Other approaches tested with non-stationary environments used the incorporation of memory mechanisms in the GA in order to help it, when some situation saw in the past is observed again [2], [3], [11].
In this work we propose a new algorithm, inspired in some ideas present in the natural immune system (IS), and we will test its performance in a classical dynamic optimization problem. The proposed GA will be referred as ISGA (Immune System-Based Genetic Algorithm). The application of ideas of the immune system in dynamic environments is not completely new and other approaches can be found in [6] and [7].

The IS exist to protect us from the dysfunction of our own cells and against the action of exogenous infectious microorganisms. The IS has two important characteristics. First, it is able to respond to infinity many pathogens by its capability to build also infinity many diverse agents adapted to kill each type of pathogen. The mechanism used combine gene libraries with a process of clonal selection, which includes a phase of somatic hypermutation. Second, the IS is able to build up a memory of previous pathogens and how to fight them, so the next time they invade the body the response will be faster and stronger.
In our work we will incorporate these two main ideas of the IS in a standard GA. We tested the ISGA with a biologically inspired genetic operator called transformation already used with success in problems dealing with dynamic environments [17], against a standard GA enhanced with memory (MGA) using one-point, two-point or uniform crossover. We will see that transformation is able to promote high levels of diversity and consequently the results were better than the ones obtained with crossover. The results also show that the proposed ISGA has a typical behavior of primary and secondary response: as the time goes on, the reaction to the changes becomes faster and stronger. We also compared ISGA with other three approaches: the GA used only with transformation (ETGA) [16], the Hypermutation GA (HMGA) proposed by Cobb [1] and the Random Immigrants GA (RIGA) proposed by Grefenstette [10].

## 2 The Immune system

The IS is a complex, distributed and multi-layered system, which includes cells, molecules and organs that constitute an identification mechanism capable of recognize and eliminate foreign molecules called antigens.
The human body maintains a large number of immune cells. Some belong to the innate IS, e.g. the macrophages, while others are part of the acquired, or adaptive IS and are called lymphocytes. There are mainly two types of lymphocytes, the **B-cells** and the **T-cells**, which cooperate but play different roles in the immune response. The B-cells can be further decomposed into plasma B-cells and memory B-cells. The same happens with the T-cells, which can be partitioned into helper T-cells and killer T-cells. The main functions of the B-cells are the production and secretion of antibodies as a response to exogenous organisms. Each B-cell produces a specific antibody,

which can recognize and bind to a specific pathogen. In order to do their job correctly the B-cells replicates by a process called **clonal selection**. This process is similar to the evolution of a population by means of a genetic algorithm using only mutation: only those cells that have high affinity with an antigen proliferate. Therefore, the B cells that have antibodies, which bind to the pathogen, are selected and cloned. Nevertheless, during cloning some variations may occur due to a process of somatic **hypermutation**. This may increase the affinity between the antibody and the antigen, making the B-cell more adapted to bind to the antigen.

When an antigen enters an organism for the first time, only a few number of B-cells can recognize it. Those cells are then stimulated to produce antibodies specific to the antigen (*primary response*). But, the IS can remember patterns that have been seen previously. This is possible because some cells, including the B-cells, become memory cells, which persist in the circulation and are capable of recognizing enemies when and if they attack again. Therefore, the next time the same pathogen invades the organism, the response of the IS will be much faster and strong (*secondary response*). We may say then the IS has **learning** capabilities based on the **memory** cells.

If each antibody recognizes only a specific antigen, how is possible that a huge number of pathogen can be recognized by the IS? The answer to this question remains in the **diversity** of the antibodies. A heavy and a light chain form each antibody molecule, and a variable and a constant region create each one of these chains. These variable and constant regions of the light chain are created by the concatenation of modular chunks of genes aggregated in **gene libraries** called V, J and C regions (see Fig. 1). The heavy chain is formed by an analogous process [4].

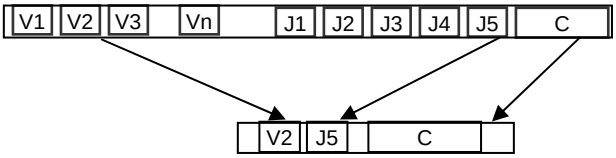


**Fig. 1.** Concatenation of genetic chunks to create an antibody molecule

It will be those ideas of gene libraries, clonal selection (including somatic hypermutation) and memory B-cells that we will translate and include into the standard GA. They are responsible for promoting diversity and ascribing memory capabilities to our modified GA.

## 3 Computational Implementation

As we said, in this work it is not our intention to mimic the functioning of the IS. We only used some of the main characteristics of the IS in order to enhance the GA so it will be more efficient when dealing with dynamic environments. In the next sections we will describe the mapping between those aspects of the IS we will use and the modified GA.

### 3.1 The immune system genetic algorithm

Our starting point is to view the environment as an antigen, and the changes in the environment as the appearance of a different antigen. We will have a process that detects whenever a change occurs. In practice, this will be achieved when degradation in the average fitness of the population is observed. The detecting mechanism is equivalent to the role of the helper T-cells.

But let's see how the GA behaves when the environment is stationary. First of all in the ISGA there are two populations. The first one is a set of individuals (the antibodies of plasma B-cells) that evolve in a process similar to clonal selection: the individuals that have the best match to the optimum (antigen) are selected and cloned. During the cloning phase every individual has a chance of being modified by a mechanism that remembers the somatic hypermutation of B-cells. This mechanism is based in a biologically inspired event called **transformation** and will be explained later. The second population is formed by a collection of individuals, which were the best ones at different moments in the past when they belong to the first population (they are the antibodies of memory B-cells). Each individual of the second population has attached a value which corresponds to the average of the first population's fitness is some particular situation, when that individual had good performance.

When a change occurs and is detected, it probably exists in the domain of the memory B-cells one that has proximity with the new conditions. This proximity is measured by the average of plasma B-cells population's fitness and the value attached to the memory B-cells. The most suitable memory B-cell is then activated, cloned and reintroduced in the population of plasma B-cells, replacing the worst ones.

As we see the two populations communicate with each other.

At the beginning of the simulation, we randomly create a set of M libraries, each one containing N gene segments of size L. The libraries are kept constant during the entire evolutionary process (Fig 2).

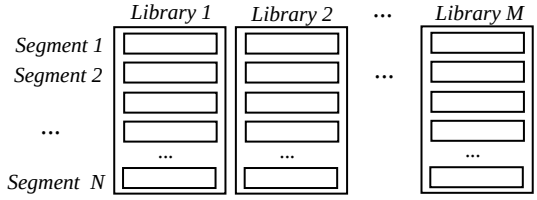


**Fig. 2.** Organization of the gene segments in libraries

These gene libraries will be used for the creation of the first memory B-cells and in the process of somatic hypermutation. In our experiments we used three libraries with eight segments of size four each and one library with eight segments of size five. The concatenation of one element of each library creates a binary string of size 17 (the length of our B-cells). The number and size of these gene libraries is problem

dependent. This idea of gene libraries was also used by [12].

### 3.2 Transformation

Somatic hypermutation is implemented by a biologically inspired mechanism called transformation. Biological transformation consists in the transfer of small pieces of extra cellular DNA between organisms. These strains of DNA, or gene segments, are extracted from the environment and added to recipient cells [14].
The process of using transformation can be described as follows: the ISGA starts with an initial population of individuals (plasma B-cells) and an initial pool of gene libraries both created at random. In each generation we select individuals to be transformed and we modify them using one gene segment of one gene library chosen at random. To transform each individual we now choose, also randomly, a point in the selected individual. The gene segment is incorporated in the genome of the individual, replacing the genes after the transformation point, previously selected. Form more details see [15].

## 4 Experimental Setup

In order to test de adequacy of ISGA to dynamic environments we made two types of experiments. In the first one, we compared ISGA with the standard GA enhanced by the memory B-cells (MGA), to see if transformation is a more effective, diversity preserving mechanism; in the second one, we compared ISGA also with three other algorithms: the Triggered Hypermutation GA (HMGA), the Random Immigrants GA (RIGA) and the enhanced transformation GA (ETGA) proposed by [1], [10] and [15], respectively. Now the goal was to see if the double effect of diversity and memory could give better results. In all cases we used the dynamic version of the 0/1 Knapsack Problem (DKP).

### 4.1 The 0/1 dynamic knapsack problem

The well-known single-objective 0/1 knapsack problem is defined as follows: given a set of $n$ items, each with a weight $W[i]$ and a profit $P[i]$, with $i = 1, ..., n$, the goal is to determine which items to include in the knapsack so that the total weight is less than some given limit (C) and the total profit is as large as possible. In the classical 0/1 knapsack problem, the capacity of the bag is kept constant during the entire run. In the dynamic knapsack problem (DKP) the weight limit can change over time between different values.
We used as a test function a 17-object 0/1 knapsack problem with oscillating weight constraint, proposed by [8]. The vectors of values and weights used for the knapsack problem are exactly the same as that used by the authors. The penalty function for the infeasible solutions is defined by: $Pen=K(\Delta W)^2$, where $\Delta W$ is the amount which the solution exceeds the weight constraint and K=20. A solution is considered infeasible if the sum of the weights of the items exceeds the knapsack capacity.
Goldberg and Smith used the DKP to compare the performance of a haploid GA and a diploid GA with fixed dominance map and a diploid GA with a triallelic dominance map. In [10] it is referred that their experimentation used variation of the knapsack capacity between two different values every 15 generations. This problem was already used by other authors to test evolutionary approaches in dynamic environments [11], [13].
In this work we enlarged the number of case studies: we used three types of changes in the capacity of the knapsack: periodic changes between two values (C1=104 and C2=60) and between three values (C1=60, C2=104 and C3=80) and non-periodic changes between 3 different capacities (C1=60, C2=80 and C3=104). Each trial allowed 10 cycles with cycle lengths of 30, 100, 200 and 300 generations.
When the changes in the environment are non-periodic we run the algorithms during 2000 generations and selected randomly several moments of change. In these moments the capacity of the knapsack was altered to a different value chosen among the same three values used in the periodic situation: 60, 80 and 104.

### 4.2 The parameters of the algorithms

The set of parameters used by the different algorithms were the following.
The three standard GA enhanced with memory (MGA) used 70% of crossover rate and 0.1% of mutation rate.
The ISGA used only transformation with a probability of 90%. Moreover the gene segment length is equal to 4 or 5, depending on the selected gene library. A detailed explanation of the reason for these values is given in [16].
The HMGA uses a baseline mutation rate (usually very low) when the algorithm is stable and increases the mutation rate (to a high value) whenever there is degradation in the performance of the time-averaged best performance [1]. We implemented this mechanism with a baseline mutation rate of 0.1% and whenever degradation is observed we increased the mutation rate to 10%, 20% or 30%. The best results were achieved by a hypermutation rate of 10% (the results presented refer to this value).
The RIGA replaces a fraction of a standard GA's population each generation, as determined by the replacement rate, with randomly generated values. This mechanism views the GA's population as always having a small flux of immigrants that wander in and out of the population from one generation to the next. This strategy effectively concentrates mutation in a subpopulation while maintaining a traditionally low (i.e., 0.001) mutation rate in the remainder of the population [10]. We tested the Random Immigrants GA with a replacement rate of 10%, 20% and 30%. The best results were achieved with the value 10% (the results presented refer to this value) Both HMGA and RIGA were run with one-point crossover with a probability of 70%.
The ETGA was run with the set of parameters obtained by the empirical study presented in [16]. The chosen

values were: no mutation rate, transformation rate equal to 90%, replacement rate of 50% and gene segment length of size 5.
All the algorithms used populations with 100 individuals and were repeated 30 times. In the ISGA and MGA we used an additional population of 20 memory B-cells. The results reported in the next section are the average values of the 30 runs.

### 4.3. Performance Measures

In order to evaluate the performance of the five approaches (seven algorithms) solving the dynamic 0/1 KP, we used two well known measures, usually employed in non-stationary problems. Those measures are the **accuracy** and the **adaptability**. They are based on a measure proposed by De Jong [5], the **off-line performance**, but evaluate the difference between the value of the current best individual and the optimum value, instead of evaluating just the value of the best individual. Accuracy (Acc) is the difference between the value of the current best individual in the population of "just before change" generation and the optimum value averaged over the entire cycle. Accuracy measures the capacity to recover to the new optimum before a new modification occurs. Adaptability (Ada) is the difference between the value of the current best individual of each generation and the optimum value averaged over the entire cycle. Adaptability measures the speed of the recovery.
The smaller measured values for accuracy and adaptability the better results. If the accuracy reaches a zero value it means that the algorithm found the optimum every time before a change occurs. If adaptability is equal to zero it means that the best individual in the population was at the optimum for all generations, i.e., the optimum was never lost by the algorithm.
These two measures can be mathematically defined by:

$$Acc = \frac{1}{K}\sum_{i=1}^{K} Err_{i,n-1} \qquad (1)$$

$$Ada = \frac{1}{K}\sum_{i=1}^{K}\left[\frac{1}{r}\sum_{j=0}^{r-1} Err_{i,j}\right] \qquad (2)$$

Where **K** is the number of changes during the run; **r** is the number of generations between two consecutive changes; $\mathbf{Err_{i,j}}$ is the difference between the value of the current best individual in the population of $j^{th}$ generation after the last change ($j \in [0, r-1]$) and the optimum value for the fitness after the $i^{th}$ change ($i \in [0, K-1]$).

## 5 Results

In this section, due to the constraints of space, we will present part of the obtained results. First we compare the ISGA with the MGA with one point crossover (MGA-Cx1). After that, we present the performance measures obtained with the proposed approach (ISGA), MGA and the other approaches already used by us (ETGA) and by other authors (HMGA and RIGA).

### 5.1 Transformation versus Crossover: the problem of diversity

As described in the paper, the proposed approach based on IS ideas incorporated a sort of memory mechanism in the GA in order to improve its efficiency when dealing with dynamic environments. Nevertheless, this mechanism of memory was not completely successful if we cannot preserve the diversity in the population. In fact, the MGA with one point crossover couldn't reach the best solution when changes occur. On the other hand ISGA, as the time passes, is always improving until reach the best solution. This behavior that can be seen in Figure 3 corresponds to the primary response and secondary response of the IS. Figures 3 and 4 show the behavior of the two algorithms when changes occur between to values every 15 generations.

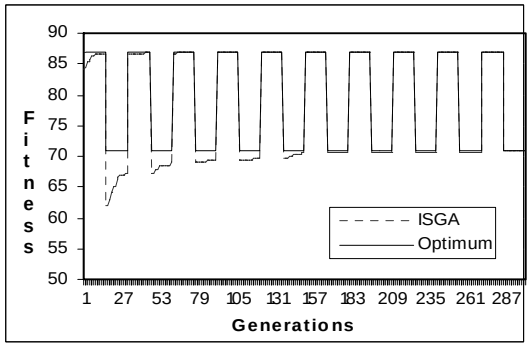


**Fig. 3.** ISGA: changes between 2 values, cycle=30

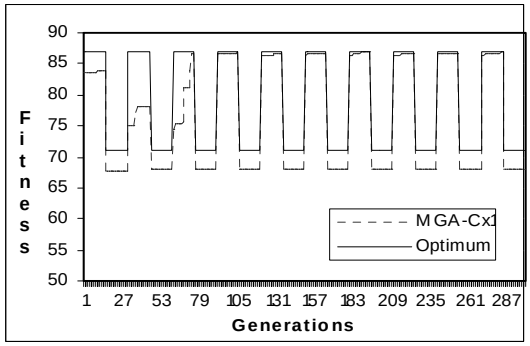


**Fig. 4.** MGA-Cx1: changes between 2 values cycle=30

For larger cycles, with changes every 150 generations, ISGA after the first change is able to recognize all the changes in the environment and the adaptation is very rapid. MGA-Cx1 had a poor performance because it memorizes one of the values but can't improve it until reach the best solution.
The behavior of ISGA and MGA-Cx1 when changes occur between three different values in periodic intervals is very similar to the previous cases. ISGA performs much better than MGA-Cx1. Figures 5 and 6 show the obtained results for cycles of 30 generations.
For non-periodic changes between three different values, the results were not as good as the ones obtained in the case of periodic changes. Nevertheless, ISGA was also better than MGA-Cx1. In this case, MGA-Cx1 had some situations where it was not able to find the new solution when a change occurred. Figures 7 and 8 show the achieved solutions.

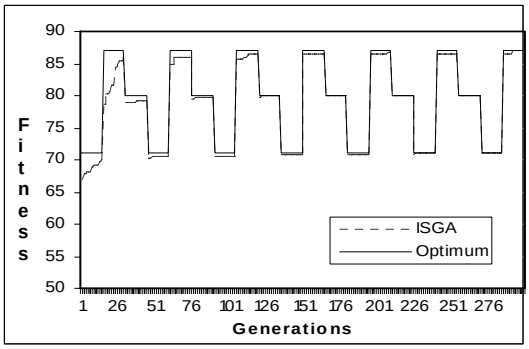


**Fig. 5.** ISGA: changes between 3 values, cycle=30

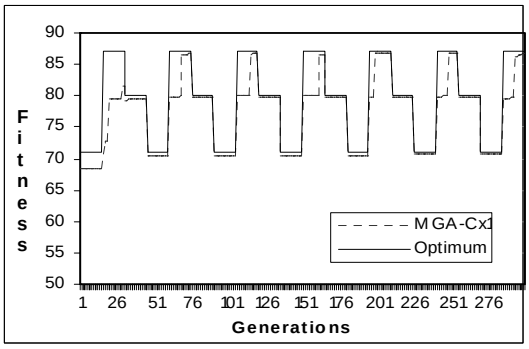


**Fig. 6.** MGA-Cx1: changes between 3 values, cycle=30

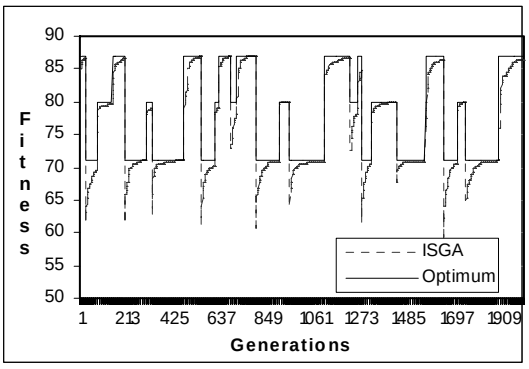


**Fig. 7.** ISGA: changes between 3 values, non-periodic

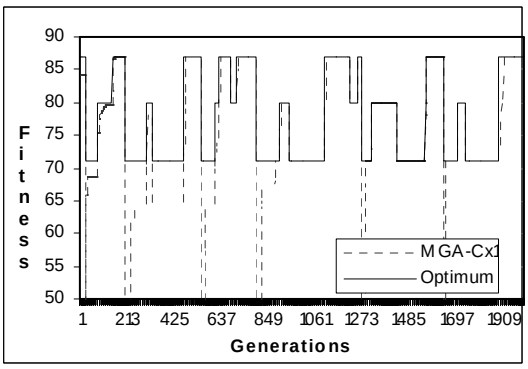


**Fig. 8.** MGA-Cx1: changes between 3 values, non-periodic

The main reason for the worst performance of MGA-Cx1 seems to be the fact that the population's diversity is loss and the memory mechanism by itself isn't enough to readapt the solutions to the new conditions of the environment. We used a standard measure of diversity defined by (3) and we can see that transformation can preserve diversity at very high.

$$Div(Pop) = \frac{1}{LP(P-1)} \sum_{i=1}^{P} \sum_{j=1}^{P} HD(p_i, p_j) \qquad (3)$$

where **L** is the length of the chromosome, **P**, the population size, $\mathbf{p_i}$, the $i^{th}$ individual in the population and **HD** the hamming distance.

Figure 9 shows the values measured for ISGA and MGA-Cx1 in the case of periodic changes between two values every 15 generations. As we can see, the MGA-Cx1 with mutation a rate equal to 0.1% cannot preserve the diverity in the population..

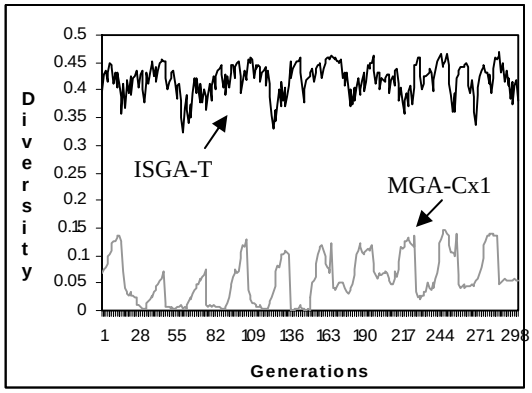


**Fig. 9.** Diversity in the population preserved by ISGA-T and MGA-Cx1

### 5.2 Immune system GA: comparing with other approaches

To evaluate the ISGA more deeply, we test it against other algorithms. We re-implemented the well-known HMGA and RIGA. We also used the GA with transformation, but no memory, enhanced by the appropriated choice for the parameters as studied in [16] (ETGA). To compare the results obtained by the proposed approach (ISGA) and also MGA (mutation rate of 0.1%), with these well known algorithms we used the measures introduced in section 4.4. Table 1 shows the accuracy and adaptability values measured during all the generations. As we can see, the ISGA was the approach that obtained, in general, the best results for periodic or non-periodic changes, with smaller or larger cycle lengths.

Moreover, in order to see how the accuracy and adaptability were in the end of the generational process (to see the effect of primary and secondary response) we present in table 2 the values obtained in the last cycle of each experiment. As we can see, ISGA reached always a zero value. This means that the new optimum was always achieved after the change occurs. MGA-Cx in some situations also achieved similar values.

**Table 1.** Accuracy and Adaptability obtained with ETGA, HMGA, RIGA, ISGA and MGA during the entire generational process

| | | | ETGA | HM 10% | RI 10% | ISGA | MGA-Cx1 | MGA-Cx2 | MGA-CxU |
|---|---|---|---|---|---|---|---|---|---|
| Periodic 2 values | Cycle = 30 | Acc | 2.20 | 1.25 | 1.78 | **0.58** | 2.22 | 4.20 | 3.68 |
| | | Ada | 5.01 | 3.26 | 4.37 | **0.78** | 2.73 | 4.24 | 3.70 |
| | Cycle = 100 | Acc | 0.29 | 0.23 | 0.59 | **0.06** | 4.14 | 3.64 | 3.94 |
| | | Ada | 2.35 | 0.98 | 2.58 | **0.16** | 4.24 | 3.83 | 4.08 |
| | Cycle = 200 | Acc | 0.05 | 0.23 | 0.50 | **0.06** | 0.49 | 3.64 | 4.14 |
| | | Ada | 1.11 | 0.61 | 1.68 | **0.17** | 1.16 | 3.70 | 4.14 |
| | Cycle = 300 | Acc | 0.02 | 0.23 | 0.45 | **0.04** | 3.28 | 0.79 | 0.14 |
| | | Ada | 0.73 | 0.52 | 1.27 | **0.15** | 3.43 | 1.05 | 0.21 |
| Periodic 3 values | Cycle = 30 | Acc | 1.45 | 0.70 | 1.08 | **0.38** | 0.72 | 1.28 | 2.16 |
| | | Ada | 3.53 | 1.84 | 2.74 | **0.68** | 2.02 | 1.85 | 3.59 |
| | Cycle = 100 | Acc | 0.18 | 0.36 | 0.22 | **0.09** | 0.57 | 6.68 | 1.10 |
| | | Ada | 1.30 | 2.18 | 1.16 | **0.31** | 1.25 | 7.52 | 4.22 |
| | Cycle = 200 | Acc | **0.02** | 0.42 | 0.15 | 0.04 | 3.64 | 1.48 | 0.28 |
| | | Ada | 0.61 | 1.39 | 0.65 | **0.19** | 4.59 | 4.45 | 1.59 |
| | Cycle = 300 | Acc | **0.01** | 0.49 | 0.22 | 0.05 | 0.58 | 1.34 | 0.16 |
| | | Ada | 0.42 | 1.35 | 0.58 | **0.18** | 1.79 | 3.38 | 1.04 |
| NonPeriodic 3 values | - | Acc | 0.77 | 0.54 | 0.93 | **0.53** | 1.27 | 1.27 | 0.98 |
| | | Ada | 2.02 | 1.13 | 2.22 | 2.29 | 5.71 | 5.26 | 4.03 |

**Table 2.** Accuracy and Adaptability obtained with ETGA, HMGA, RIGA, ISGA and MGA in the last cycle

| | | | ETGA | HM 10% | RI 10% | ISGA | MGA-Cx1 | MGA-Cx2 | MGA-CxU |
|---|---|---|---|---|---|---|---|---|---|
| Periodic 2 values | Cycle = 30 | Acc | 1.90 | 0.73 | 0.47 | **0.00** | 3.00 | 4.00 | 4.00 |
| | | Ada | 4.01 | 2.25 | 0.76 | **0.00** | 3.00 | 4.00 | 4.00 |
| | Cycle = 100 | Acc | 0.97 | 0.20 | 0.83 | **0.00** | 4.00 | 4.00 | 4.00 |
| | | Ada | 3.40 | 0.72 | 2.18 | **0.00** | 4.00 | 4.00 | 4.00 |
| | Cycle = 200 | Acc | 0.23 | 0.20 | 0.73 | **0.00** | 0.13 | 4.00 | 4.00 |
| | | Ada | 1.29 | 0.69 | 1.77 | **0.00** | 0.13 | 4.00 | 4.00 |
| | Cycle = 300 | Acc | 0.10 | 0.40 | 0.50 | **0.00** | **0.00** | 0.27 | **0.00** |
| | | Ada | 0.81 | 0.62 | 1.14 | **0.00** | **0.00** | 0.27 | **0.00** |
| Periodic 3 values | Cycle = 30 | Acc | 1.73 | 0.60 | 1.93 | **0.00** | 0.30 | **0.00** | **0.00** |
| | | Ada | 5.72 | 2.40 | 5.61 | **0.00** | 4.24 | 1.63 | 4.73 |
| | Cycle = 100 | Acc | 0.07 | 0.20 | 0.47 | **0.00** | **0.00** | 5.00 | 3.00 |
| | | Ada | 1.78 | 0.61 | 2.32 | **0.00** | 0.84 | 6.64 | 4.16 |
| | Cycle = 200 | Acc | 0.00 | 0.30 | 0.10 | **0.00** | 4.00 | **0.00** | **0.00** |
| | | Ada | 1.03 | 0.54 | 1.00 | **0.00** | 4.50 | 0.98 | 0.57 |
| | Cycle = 300 | Acc | 0.00 | 0.20 | 0.40 | **0.00** | **0.00** | **0.00** | **0.00** |
| | | Ada | 0.44 | 0.38 | 0.98 | **0.00** | 1.40 | 0.96 | 0.85 |
| NonPeriodic 3 values | - | Acc | 0.60 | 0.10 | 0.70 | **0.00** | **0.00** | **0.00** | **0.00** |
| | | Ada | 2.03 | 0.46 | 2.12 | **0.00** | **0.00** | **0.00** | **0.00** |

## 6 Conclusions

In this paper we proposed an immune system-based GA to deal with dynamic environment. We tested it in different situations, namely periodic and non-periodic changes and different cycle lengths. The main characteristic of this ISGA is the ability to remember past situations with faster and stronger reactions obtained as time goes on, e.g. a kind of secondary response typical of the natural immune system. Moreover, using transformation, as a hypermutation operator, ISGA is able to keep the diversity in the population.

On the overall, when we compare ISGA with the other algorithms (MGA, ETGA, RIGA and HMGA), it always achieved the best accuracy and adaptability values.

As the results are very promising, the next step in our work is to make ISGA more close to the natural IS (for instance, we want to start with a plasma B-cells population created using also the gene libraries), and to test it in other problems involving dynamic environments.

## References


1. H. Cobb (1990). *An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments.* Technical Report AIC-90-001.
2. J. Branke (1999). *Memory Enhanced Evolutionary Algorithm for Changing Optimization Problems*. In Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1875-1881, IEEE.
3. J. Branke (2002). *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.
4. D. P. Clark, L. D. Clark (1997). *Molecular Biology made simple and fun*. Cache River Press.
5. K. A. De Jong (1975). Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Dissertation, Department of Computer and Communication Science, University of Michigan.
6. A. Gaspar, P. Collard (1999). *From GAs to Artificial Immune Systems: Improving Adaptation in Time Dependent Optimization.* Proc. of the 1999 Congress on Evolutionary Computation, pp. 1859-1866, IEEE.
7. A. Gaspar, P. Collard (2000). *Immune approaches to experience acquisition in time dependent optimization*. In A. S. Wu (ed.), GECCO'2000 Workshop Proceedings.
8. D. E. Goldberg, R. E. Smith (1987). *Nonstationary Function Optimization using Genetic Algorithms with Dominance and Diploidy*. In J. J. Grefenstette (ed.), Proceedings of the 2nd International Conference on Genetic Algorithms, pp. 59-68. Laurence Erlbaum Associates.
9. D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization and Machine* Learning. Addison-Wesley Publishing Company, Inc.
10. J. J. Grefenstette (1992). *Genetic Algorithms for Changing Environments*. In R. Maenner, B. Manderick (eds.), Parallel. Problem Solving from Nature 2, pp. 137-144. North Holland.
11. A. Hadad, C. Eick (1997). *Supporting Poliploidy in Genetic Algorithms using Dominance Vectors*. In P. Angeline et al. (eds.) Proc. of the 6th International Conf. on Ev. Programming, vol. 1213 of LNCS. Springer.
12. R. R. Hightower, S. Forrest, A. S. Perelson (1995). *The Evolution of Emergent Organization in Immune System Gene Libraries.* In Proc. of the 6th Int. Conf. on Genetic Algorithms, pp. 344-350, Morgan Kaufmann.
13. K. P. Ng, K. C. Wong (1995). *A New Diploid Scheme and Dominance Change Mechanism for Non-stationary Function Optimization.* In Proc. of the 6th Int. Conf. on Genetic Algorithms, pp. 159-166. Morgan Kaufmann.
14. P. J. Russell (1998). *Genetics*. 5th edition, Addison-Wesley.
15. A. Simões, E. Costa (2001). *On Biologically Inspired Genetic Operators: Transformation in the Standard Genetic Algorithm*. In Lee Spector et. Al (eds), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), pp. 584-591, Morgan Kaufmann.
16. A. Simões, E. Costa (2003a). *Improving the Genetic Algorithm's Performance when Using Transformation.* Proceedings of ICANNGA 2003.
17. A. Simões, E. Costa (2003b). *A Comparative Study Using GAs to Deal with Dynamic Environments*. Proceedings of ICANNGA 2003.