Evolutionary Optimization in Non-Stationary Environments

Krzysztof Trojanowski[*] and Zbigniew Michalewicz [*†]

Abstract

As most real-world problems are dynamic, it is not sufficient to "solve" the problem for the some (current) static scenario, but it is also necessary to modify the current solution due to various changes in the environment (e.g., machine breakdowns, sickness of employees, etc). Thus it is important to investigate properties of adaptive algorithms which do not require re-start every time a change is recorded.

In this paper such non-stationary problems (i.e., problems, which change in time) are considered. We describe different types of changes in the environment. A new model for non-stationary problems and a classification of these problems by the type of changes is proposed.

We apply evolutionary algorithms to non-stationary problems. We extend evolutionary algorithm by two mechanisms dedicated to non-stationary optimization: redundant genetic memory structures and a diversity maintenance technique — *random immigrants* mechanism. We report on experiments with evolutionary optimization employing these two mechanisms (separately and together); the results of experiments are discussed and some observations are made.

1 Introduction

Most optimization algorithms assume static objective function; they search for a near-optimum solution with respect to some fixed measure (or set of measures), whether it is maximization of profits, minimization of a completion time for some tasks, minimization of production costs, etc. However, real-world applications operate in dynamic environments, where it is often necessary to modify the current solution due to various changes in the environment (e.g., machine breakdowns, sickness of employees, etc). Thus it is important to investigate properties of adaptive algorithms which do not require re-start every time a change is recorded.

Let us consider an electric company. Periods of work and rest in the industry (e.g., day and night periods, periods of five days of work and two days of rest, summer holidays, heating during the winter) mixed with some occasional changes (like special days of the year, e.g., Christmas holidays), natural anomalies (early frost, long summer, floods, etc.) and unpredictable events (e.g., breakdowns) make demands of energy varying in time. Fortunately power system in every country has an over-supply of the produced energy with respect to the demand but anyway the system of power control needs a flexible optimization algorithm to control and manage energy sources efficiently.

Let us consider a typical factory. When the list of tasks and the list of resources, which are necessary to realize these tasks, change in time, the optimization of task schedule is in fact a real-time optimization with a varying optimization function and a varying set of constraints.

Let us consider also a navigation problem. Navigation is a simultaneous path-planning and movement to the goal along the path. For non-stationary environments once optimized path could be useless if an unexpected object (e.g., unknown obstacle) is detected in the environment.

We can generalize these three examples to a class of optimization tasks of the same type: these are non-stationary problems which change in time. We are interested in solving these problems with evolutionary computation techniques. It would be interesting to investigate, which extensions of evolutionary algorithms

[*]Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland

[†]Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA

are useful in these scenarios. In this paper we discuss the nature of non-stationary problems especially from the point of view of evolutionary search process.

The paper is organized as follows. Section 2 defines a model of a stationary/non-stationary problem. Section 3 presents a brief review of existing approaches to non-stationary problem optimization. Section 4 gives a detailed description of methods for non-stationary problem optimization. To create a non-trivial demanding environments, a particular test-case generator was proposed: it is described in Section 5. The structure of evolutionary algorithm and its parameters are discussed in Section 6, together with the issue of evaluating the experimental results. Sections 7 and 8 report on results of various experiments, whereas Section 9 concludes the paper.

2 The model

Most real-world optimization problems can be modeled by specifying the variables of the problem together with their domains, the objective function to be optimized, and a set of constraints the solution must satisfy. We assume that the problem has n decision variables, $\vec{x} = (x_1, \ldots, x_n)$. There is also an additional discrete variable t, which plays the role of time (note, that time is continuous, however, any changes are usually recorded in discrete intervals).

Thus, a model $\mathcal{M}$ of a problem P can be expressed as:

$$\mathcal{M}(P) = (\mathcal{D}, \mathcal{F}, \mathcal{C})$$

where:

- $\mathcal{D}$ – domain of the variables of the problem. It is a $(n+1)$-dimensional search-space:

  $$\mathcal{D} = \prod_{i=1}^{n} \langle q_i(t), r_i(t) \rangle$$

  where $x_i \in \langle q_i(t), r_i(t) \rangle$ for $1 \le i \le n$ ($\langle q_i(t), r_i(t) \rangle$ is a range for the i-th variable at time t).

- F – an evaluation function (implied by the problem), possibly extended by the time dimension:

  $$F = f(\vec{x}, t) \ : \ \mathcal{D} \to \mathbb{R}.$$

- $\mathcal{C}$ – set of constraints:

  $$\mathcal{C} = \{c_i(\vec{x}, t) \le 0\}, \ i = 1, \ldots, v(t).$$

Let us discuss these components in more detail. Note that domains are usually divided into two categories: (1) continuous and (2) discrete domains. If both types of variables are present in the the problem, we deal with so-called mixed programming problem. There are two forms of changes of the domain during the search process:

1. the intervals for domain variables can change, and
2. the number of dimensions of the search space can change — some variables may disappear or a new variables can be introduced.

Changes of the number of dimensions and changes of intervals for domain variables usually require a re-start of the search procedure. Note that a change of interval seems to be just a change in a constraint, however, there is an important difference: if a solution is represented as a binary string, such a change may imply a change in the length of binary string (due to precision requirement). In this paper we assume that the domains are constant in time, i.e., $q_i(t) = q_i$ and $r_i(t) = r_i$.

The model discussed above represents both stationary and non-stationary problems. The difference is in the role of time in the evaluation function and constraints. If the variable of time t is present in the formula of evaluation function, i.e., $F = f(\vec{x}, t)$ or in constraint inequalities (i.e., $c_i = c_i(\vec{x}, t)$ for some i), then the model represents a non-stationary problem, otherwise we deal with a stationary one.

Note that all problems represented in the model can be divided further into six categories, since there are two categories of objective function and three categories of the set of constraints:

- the objective function may or may not depend on the time variable t, and
- the set of constraints $\mathcal{C}$ can be empty, non-empty and time independent, and non-empty and time dependent.

Table 1 provides a classification of all possible cases. Note that the first and the second class of problems are the stationary cases, i.e., they represent situations where the problem do not have any time context. These two classes have been investigated heavily in the EA community during the last twenty years.

Table 1: Classification of changes in a process. The symbol $\emptyset$ denotes the case where the set of constraints is empty; *static* — there are no changes in time; *var* — there are some changes in time

No.	*objective function*	*constraints*
1	*static*	$\emptyset$
2	*static*	*static*
3	*static*	*var*
4	*var*	$\emptyset$
5	*var*	*static*
6	*var*	*var*

In 1975 De Jong [8] studied usefulness of evolutionary techniques to five different evaluation functions which belonged to the first class of problems (i.e., the objective function is constant; the set of constraints is empty). Since that time a large group of researchers continued studying properties of evolutionary algorithms, proposed new operators of selection and variability, new population maintenance techniques, and other extensions of basic evolutionary algorithm for many different but stationary problems from the first class of problems. For the second class of problems many constraint-handling methods (e.g., methods based on preserving feasibility of solutions, penalty functions, repair algorithms, specialized operators, etc.) were proposed [26]. Clearly, the largest effort of the researchers of evolutionary computation community has been focused exclusively on these two classes of problems.

However, as discussed in Introduction, most real-world applications operate in dynamic environments, which are modeled by classes 3–6. Most researchers who investigated the use of evolutionary algorithms for non-stationary problems (see Section 3) concentrated on class 4, whereas other classes better represent real world problems. In this paper we discuss test cases from classes 3 and 4.

3 Evolutionary approaches to non-stationary problem optimization

Extensions of evolutionary algorithm to consider changes which may occur during the search process and to track the optimum efficiently in spite of this changes have been studied by several researchers over the last few years. These extensions can be grouped into three general categories:

- **Maintenance of the population diversity level.** The presence of many potential solutions during the evolutionary search seems to be a useful feature in optimization in changing environments. As long as some level of the population diversity is uphold we could expect the algorithm to adapt easier to changes. Hence maintaining diversity of the population could increase search performance of the algorithm.

  Among many maintaining population diversity techniques we can select:

  - sharing and crowding techniques [16, 4],
  - techniques based on the concepts of temperature and entropy [27, 28],
  - techniques based on the concept of the age of individuals [11],
  - a random immigrants mechanism [5, 17],
  - a mechanism of variable range local search around the current locations [39].

- **Adaptation and self-adaptation mechanism.** Dynamical adjustment of the algorithm to the non-stationary environment is a feature of the efficient optimization. So adaptive and self-adaptive techniques constitute significant extensions of evolutionary algorithms [1, 3, 9].

- **Redundancy of genetic material.** One of the most important abilities in adaptation to changes is reasoning from previous experiences. So it might be worthwhile to investigate memory structures as possible extensions to evolutionary algorithms.

  One of the earliest forms of memory (although not used for non-stationary optimization) was *Tabu Search* strategy [13, 14]. Also a considerable number of ideas to incorporate past experience was proposed in connection with evolutionary algorithms; we can classify them into several types [37]:

  - **numerical memory** — where the modification of algorithm parameters is performed using experience of previous generations [35, 30, 31, 36, 37, 42]. This type of memory has a form of additional numerical parameters. They are updated every generation using the results of the previous search. Their influence on the search process is realized by modification of the behavior of search operators: mutation or crossover.
  - **symbolic memory** — where the algorithm gradually learns from the individuals in the populations and thus constructs beliefs about the relevance of schemas (Machine Learning theory is exploited) [38]. The symbolic type of memory encodes some knowledge in its structures which have a form of rules used to guide search operators.
  - **exact memory** — where existing structures are enhanced by additional genes, chromosomes (diploidy) or groups of chromosomes (polyploidy) [6, 7, 15, 18, 24, 25, 27, 32, 33, 34, 41, 43]. The memory is utilized during the search process and between the search tasks as well. Change of the current active chromosome of the individual by the data from memory is controlled by some dominance functions which behavior depends on the type of stored data (chromosomes or just single genes), the structure of memory (linear, hierarchical, etc.) and a form of access — it is common for the whole population, an individual or a single gene only.

4 Tested extensions of the evolutionary algorithm

In our research we extended the evolutionary algorithm by *three* different mechanisms improving efficiency of evolutionary search: random immigrants mechanism, sharing, and memory structures.

The mechanism of random immigrants [5, 17] works on the entire population. In case of a change in evaluation function, a large part of population is replaced by randomly generated individuals. The number of replaced individuals in the population is controlled by the parameter called *replacement rate*.

An algorithm with sharing changes the evaluation function formula. The fitness of an individual $\vec{x}$ equals:

$$f'(\vec{x}) = f(\vec{x})/f_s(\vec{x})$$

where $f(\vec{x})$ is an original evaluation function and:

$$f_s(\vec{x}) = \sum_{\vec{y}} sh(dist(\vec{x}, \vec{y}))$$

where $dist(\vec{x}, \vec{y})$ — an Euclidean distance between $\vec{x}$ and $\vec{y}$ points and:

$$sh(dist(\vec{x}, \vec{y})) = \begin{cases} 1 - \left(\frac{dist(\vec{x},\vec{y})}{\delta_{sh}}\right)^{\alpha} & \text{if } dist(\vec{x}, \vec{y}) < \delta_{sh} \\ 0 & \text{otherwise} \end{cases}$$

α and δ_{sh} are the constants — parameters of equation.

There are various techniques based on memory structures, which may vary on (1) memory content, (2) process of remembering, and (3) process of recalling. In this paper we report on results for one set of choices made for these categories; we discuss them in turn.

• **Memory structure and content.** An individual consists of an *active chromosome*, which represents a solution, and a *memory buffer*, which may contain several chromosomes inherited from the individual's ancestors. The size of the memory buffer is constant during the time of evolutionary process.

• **Process of remembering.** Individuals of the first generation have empty memory buffers. Then, each time after a new individual is generated, if it is included in the next generation of population, the active chromosome of its parent (or a better parent — in case there are two parents) is added to its memory buffer. In addition, it will inherit the chromosomes in the memory buffer of its parent or better parent. Thus, what is remembered (i.e., the content of memory buffers) increases as the generation number increases. When the memory buffer is full the chromosome to delete is selected to make room for a new one. Each memory buffer is a FIFO queue such that when it is full, the oldest chromosome is deleted to make room for a new one.

• **Process of recalling.** In our implementations memory is recalled every time the change appears in the environment. Note that this is the time when all individuals in the current population are re-evaluated (to take into account the effect of the change). During this re-evaluation process, the chromosomes in the memory buffer of an individual are re-evaluated together with its active chromosome. After the re-evaluation, the best chromosome from the memory (if it is better than the active chromosome of the individual), replaces the active chromosome, which in turn is placed in the memory.

5 Testing environment

It is important to create a non-trivial demanding environment which could be a good test-bed for the experiments. The proposed test-case generator creates such a search space which can be used for various experiments.

The general idea is to divide the search space $\mathcal{D}$ into a number of disjoint subspaces $\mathcal{D}_k$ and to define a unimodal function f_k for every $\mathcal{D}_k$. Thus the objective function G is defined on $\mathcal{D}$ as follows:

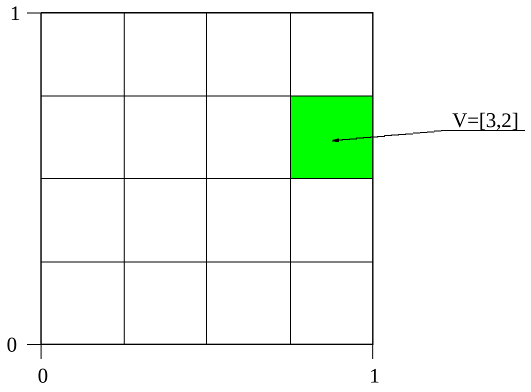
$G(\vec{x}) = f_k(\vec{x})$ iff $\vec{x} \in \mathcal{D}_k$.

The number of subspaces $\mathcal{D}_k$ corresponds to the total number of local optima of function G. This allows to create demanding and complex problems easy to control.

The search space is defined as

$$\mathcal{D} = \prod_{i=1}^{n} [0, 1),$$

where n is a number of dimensions. The domain of each of n variables is divided into w disjoint and equal sized segments of the length $\frac{1}{w}$. Thus the total number of cubes in the search space is equal w^n. For two-dimensional search space it is a patchy landscape displayed in Figure 1 on the left.



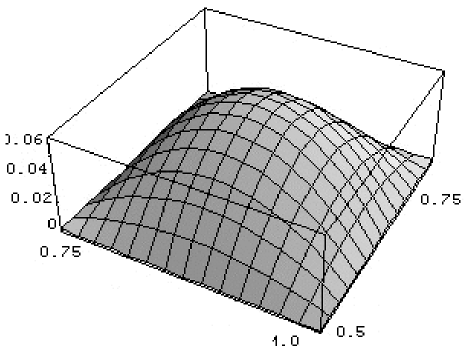


Figure 1: An example search space — a patchy landscape with $n = 2$ and $w = 4$. One cube is selected (on the left). An evaluation function of a single cube for 2-dimensional search space (on the right).

Every cube (i.e., subspace $\mathcal{D}_k$) in the space is identified by the vector $\vec{V} : [v_1, \cdots, v_n]$ where v_i identifies position of the cube by the i-th dimension (of course, $0 \leq v_i \leq w - 1$ for $i = 1, 2, \ldots, n$). The boundaries of a cube defined by a vector $[v_1, \cdots, v_n]$ are $[q_i, r_i)$ for the i-th dimension, where:

$$q_i = \frac{v_i}{w} \qquad r_i = \frac{v_i + 1}{w}.$$

The selected cube in Figure 1 is identified by a vector $[3, 2]$ and its lower and upper boundaries for both dimensions are:

$$\begin{cases} q_1 = \frac{3}{4} = 0.75 & q_2 = \frac{2}{4} = 0.5 \\ r_1 = \frac{3+1}{4} = 1 & r_2 = \frac{2+1}{4} = 0.75 \end{cases}$$

For every cube of the space an evaluation function is defined:

$$f_{\vec{V}}(x_1, \cdots, x_n) = a_{\vec{V}} \prod_{i=1}^{n} (r_i - x_i)(x_i - q_i)$$

with maximum in the middle point of the cube $\vec{V}$ (Figure 1 on the right) equal to $a_{\vec{V}}(\frac{1}{2w})^{2n}$.

A n-dimensional matrix H_a (with w elements along each dimension) defines the values of $a_{\vec{V}}$. For example for a given 2-dimensional environment ($n = 2$) with $w = 4$ we considered the following matrix H_a:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

This environment has 16 global optimum points of the same height as all 16 cubes in the matrix has the same value (values $a_{\vec{V}}$ are equal to each other). Then the environment created by the test-case generator is presented in Figure 2 on the left.

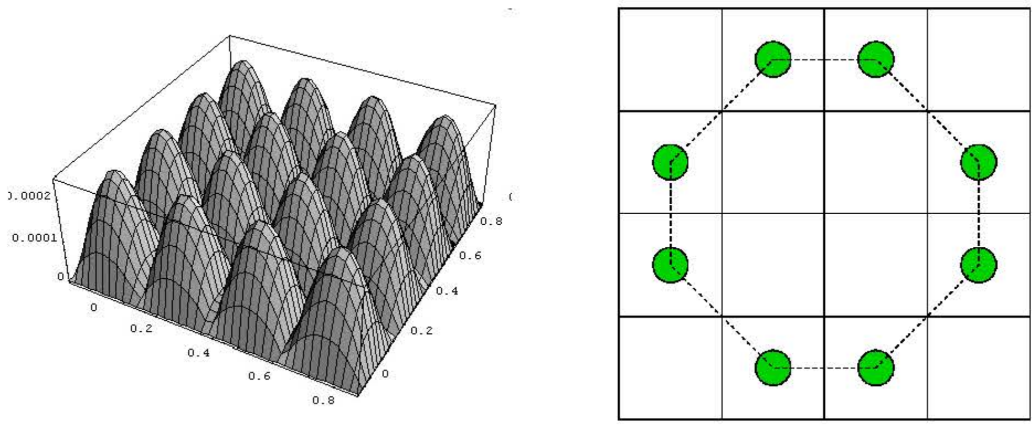


Figure 2: An evaluation function of an example environment in 2-dimensional search space (on the left). An example set of non-stationary cubes in the 2-dimensional environment (cubes marked by circles) (on the right).

To convert the search space into a non-stationary one, it is sufficient to change some values $a_{\vec{V}}$ in the matrix H_a in time. The list of subspaces $\mathcal{D}_k$ with changing values is a sequence. For the above sample environment, let us consider a list of cubes which create a contour, called a *path of changes* (Figure 2 on the right). The values in the matrix will change (every τ generations): the values which are on the path of change, would rotate clockwise.

Such a time-dependent matrix $H_a(t)$ together with a path of changes allows to model many types of changes. For example, to experiment with a "mouse under the carpet" scenario, the values $a_{\vec{V}}$ for these non-stationary cubes are listed in boldface; an index represents the position of the cube in the list:

$$\begin{bmatrix} 1 & \mathbf{1_7} & \mathbf{1_6} & 1 \\ \mathbf{1_8} & 1 & 1 & \mathbf{1_5} \\ \mathbf{2_1} & 1 & 1 & \mathbf{1_4} \\ 1 & \mathbf{1_2} & \mathbf{1_3} & 1 \end{bmatrix}$$

After τ generations, there is a anti-clockwise rotation of path of change values; this would result in the following values:

$$\begin{bmatrix} 1 & \mathbf{1_7} & \mathbf{1_6} & 1 \\ \mathbf{1_8} & 1 & 1 & \mathbf{1_5} \\ \mathbf{1_1} & 1 & 1 & \mathbf{1_4} \\ 1 & \mathbf{2_2} & \mathbf{1_3} & 1 \end{bmatrix}$$

After the following τ generations, there is another change. These changes cause a movement of optimum coordinates in the environment: from the cube $[0, 1]$ after τ generations they moved to the cube $[0, 2]$, then to cube $[1, 3]$ and so on. After eight such changes (as they are 8 subspaces listed on the path of change), each of them of τ generations, the values in the search space return to the original ones. This completes one *cycle* of changes.

The form of changes described above is just one of many possibilities offered by the test-case generator. We can propose different paths of changes (their topology and length) as well as the sequence of values (on the path and outside the path). Modifications can have periodical or non-periodical nature. Additionally we can mixed periodical changes on the path with random changes which would happen outside the path. The results of these experiments are discussed in the full version of the paper.

6 Description of AMIGA and estimation criteria

For empirical research an evolutionary algorithm called AMIGA (IndividuAl Memory aIded Genetic Algorithm) was implemented:

• Individual representation: chromosomes are Gray coded bit strings.

• Initial population: it is a randomly selected group of individuals. In case of changes of the evaluation function the population is neither re-initialized nor repaired. It is re-evaluated only.

• Operators: two classic operators were applied in the algorithm. They were 1-point crossover and bit-flip mutation.

• Selection: a tournament selection with a tournament size of 2 was applied. Applied selection was of a generational type with elitism.

• Parameters: a set of parameters defined the algorithm. Some of them were static and did not change over time. These were:

– number of bits per dimension in the individual: $n_{bpd} = 12$,

– crossover and mutation probabilities: $p_c = 0.9$ and $p_m = 0.04$,

– population size: $N = 80$.

The termination-condition used was a fixed number of generations, which depends on the length of the cycle of changes (for periodical type of changes). We tried to have at least 4 full cycles in every experiment.

When a problem is stationary (neither an evaluation function nor a set of constraints change in time) it is relatively easy to compare the results of various algorithms. However, when a problem is non-stationary, the situation is more complex, as it is necessary to measure not the final result (which does not exist in the continuous process of tracking the moving optimum), but rather the search process itself (e.g., its reactions to different types of changes).

In evolutionary computation community some measures of obtained results have been proposed; these measures exploited the iterational nature of the search process and the presence of continuously modified and improved population of solutions. One of the first measures were on-line and off-line performance proposed by De Jong in 1975 [8]:

- **off-line performance** — is the best value in the current population averaged over the entire run. It represents the efficiency of the algorithm in the given time of run.
- **on-line performance** — is the average of all evaluation of the entire run. It shows the impact of the population on the focus of the search.

These two measures, although designed for static environments, were employed in experiments with non-stationary ones [2, 17, 39, 40].

In other publications authors visually compared graphs of the best objective function value measured during the entire search process (or graphs of the mean value obtained from series of experiments) [1, 3, 5, 4, 7, 11, 15, 17, 23, 27, 28, 29, 40]. In some papers graphs of average values of all individuals or of the worst individual in the population were also analyzed [5, 15, 7, 27, 28]. Both these methods were based on the measures of off-line and on-line performance.

An interesting measure based on the off-line performance was an *adaptation performance* described in [27]. It was evaluated according to the formula:

$$I = \frac{1}{T_{max}} \sum_{i=1}^{T_{max}} \frac{f_{best}(t)}{f_{opt}(t)}$$

where:

T_{max} — the length of the entire search process,
$f_{best}(t)$ — the fitness of the best individual in the population at the time t,
$f_{opt}(t)$ — the fitness of the optimum point in the search space at the time t.

This formula was later modified slightly to:

$$I = \frac{1}{T_{max}} \sum_{i=1}^{T_{max}} \alpha \frac{f_{best}(t)}{f_{opt}(t)} \qquad \alpha = \begin{cases} 1, & \text{if } f_{best}(t) = f_{opt}(t) \\ 0.5, & \text{if } f_{best}(t) < f_{opt}(t) \end{cases}$$

In [10] two benchmarks measuring relative closeness of the best found solution to the global optimum were proposed: Optimality Op and Accuracy Ac. Optimality Op represents closeness of the value of the best obtained solution $f(\vec{x}_0)$ to the value of optimum f_{opt}. For maximization and minimization problems we have following formulas respectively:

$$Op_{max}(\vec{x}_0) = \frac{f(\vec{x}_0) - f_{min}}{f_{max} - f_{min}} \qquad Op_{min}(\vec{x}_0) = \frac{f_{max} - f(\vec{x}_0)}{f_{max} - f_{min}}$$

Accuracy Ac represents the relative closeness of a solution found to the global optimum solution $\vec{x}_{opt}$ and it is defined with following formula:

$$Ac(\vec{x}_0) = 1 - \frac{|\vec{x}_{opt} - \vec{x}_0|}{\vec{x}_{max} - \vec{x}_{min}}$$

Although authors did not use these measures to non-stationary optimization evaluation, the closeness to the optimum during the search process is an interesting value which seems to be helpful in comparisons between applications and easy to control in experiments.

Another measure was based on the observation of the population distribution. In [28, 40] authors controlled population entropy which is a measure of disorder in the population. Forms of the entropy evaluation depended on the demands of the applied algorithm. For example in [28] the entropy was evaluated in a locus-wise manner i.e. it was evaluated separately for every locus in the individual in comparison to locuses on that position in all other individuals in the population.

For results estimations of non-stationary optimization process we proposed the following two measures: *Accuracy* — Acc and *Adaptability* — Ada. They are based on a measure proposed by De Jong [8]: *off-line performance* but evaluate difference between the value of the current best individual and the optimum value instead of evaluation of the value of just the best individual.

• Accuracy — 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:

$$Acc = \frac{1}{K} \sum_{i=1}^{K} (err_{i,\tau-1})$$

• Adaptability — a difference between the value of the current best individual of each generation and the optimum value averaged over the entire run:

$$Ada = \frac{1}{K} \sum_{i=1}^{K} \left[\frac{1}{\tau} \sum_{j=0}^{\tau-1} (err_{i,j}) \right]$$

where: τ — number of generations between changes when the environment was static; $err_{i,j}$ — a difference between the value of the current best individual in the population of j-th generation after the last change ($j \in [0, \tau - 1]$), and the optimum value for the fitness landscape after the i-th change ($i \in [0, K-1]$); K — the number of changes of the fitness landscape during the run.

Clearly, the smaller the measured values are (for both Accuracy and Adaptability), the better the result is. In particular, a value of 0 for Accuracy means that the algorithm found the optimum every time before the landscape was changed (i.e., τ generations were sufficient to track the optimum). On the other hand, a value of 0 for Adaptability 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 are helpful in evaluating the quality of the search process. For example, results with low values for Accuracy and larger values for Adaptability can be interpreted that the algorithm loses the optimum after a change is made, but the time interval between changes is long enough to recover.

7 Non-stationary evaluation function — results

In the following we discuss results of experiments performed for environments where the evaluation function landscape changes in time and the set of constraints is empty (class 4, see Table 1). These environments were generated by the test-case generator described in previous section.

7.1 Selection of population diversity maintenance technique

Two population diversity maintenance approaches were selected to compete:

— evaluation function with sharing,

— random immigrants mechanism.

To compare these techniques, a series of experiments were performed for different values of parameters and two different environments with periodical changes of global optimum coordinates. The differences between them were in the length of period of changes and the number of non-stationary cubes. The paths of changes of the environments are presented in Figure 3.

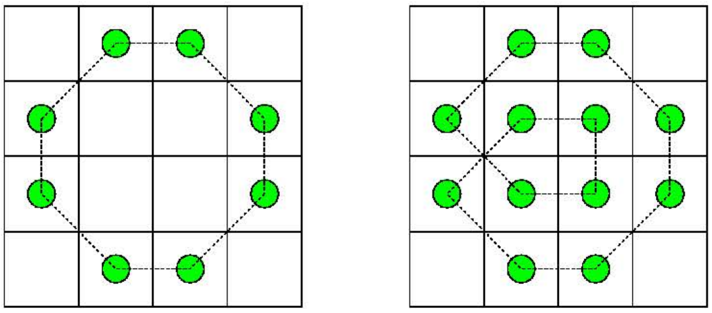


Figure 3: An environment no. 1 (on the left) and an environment no. 2 (on the right)

The matrices M_a of the environments no. 1 (on the left) and no. 2 (on the right) are presented in Figure 4. Values of non-stationary cells in the matrices are displayed with bold letters.

For every environment, we made two series of 50 experiments. In the first one, we used evaluation function with sharing for different values of sharing factor δ_{sh}. Because of regularities in the landscape of generated environments we made the value of δ_{sh} a function of w — the number of cubes per dimension:

$\delta'_{sh}(w) = \frac{\delta_{sh}}{w}$

$$
(a) \quad \begin{bmatrix} 0.100 & \mathbf{0.146} & \mathbf{0.000} & 0.100 \\ \mathbf{0.500} & 0.100 & 0.100 & \mathbf{0.146} \\ \mathbf{0.546} & 0.100 & 0.100 & \mathbf{0.500} \\ 0.100 & \mathbf{1.000} & \mathbf{0.546} & 0.100 \end{bmatrix}
$$

$$
(b) \quad \begin{bmatrix} 0.100 & \mathbf{0.067} & \mathbf{0.250} & 0.100 \\ \mathbf{0.000} & \mathbf{0.650} & \mathbf{0.500} & \mathbf{0.500} \\ \mathbf{0.467} & \mathbf{0.067} & \mathbf{0.250} & \mathbf{0.650} \\ 0.100 & \mathbf{1.000} & \mathbf{0.467} & 0.100 \end{bmatrix}
$$

Figure 4: Matrices H_a for environments no. 1 (matrix a), and no. 2 (matrix b). Non-stationary values are listed in boldface.

Therefore for experiments with environments created with test-case generator the shared fitness value of an individual was evaluated with a following value of $sh(dist(\vec{x},\vec{y}))$:

$$
sh(dist(\vec{x},\vec{y})) = \begin{cases} 1 - \left(\frac{dist(\vec{x},\vec{y})}{\delta_{sh}} w\right)^{\alpha} & \text{iff } dist(\vec{x},\vec{y}) < \frac{\delta_{sh}}{w} \\ 0 & \text{otherwise} \end{cases}
$$

This way we could easily observe the efficiency of sharing respectively to the number of cubes being in the range of sharing. E.g., for $\delta'_{sh} = 1$ an individual is evaluated in context of all those individuals which are placed not further than the length of one cube — $dist(\vec{x},\vec{y}) < \frac{1}{w}$. Additionally we assumed that for all experiments $\alpha = 1$.

The second group of experiments with random immigrants mechanism was performed for different values of replacement rate. The environments were changed every $\tau = 5$ generations. The values in the list were moved by one position anti-clockwise. As 8 values in the matrix of environment no. 1 are involved in the path of change, one cycle consisted of 8 changes, so a single cycle took 40 generations. The matrix of environment no. 2 had 12 values in the path so a single cycle took 60 generations. The results of experiments are presented as graphs in Figures 5, 6, 7, and 8.

Comparing the graphs it can be seen that random immigrants mechanism gave better results than sharing. There exist values of replacement rate for which the obtained Accuracy had smaller value than the Accuracy for any of values of sharing factor.

We can also observe that for regular landscapes of evaluation function sharing is sensitive to the distances between local optima and to the shape of path of changes therefore some knowledge about the distances between optima is necessary for tuning the δ_{sh} parameter.

On the other hand since δ_{sh} is the same for all individuals the results should be much better for regular environments where distance from the peak to every its neighbor peak is equal and should be some worse when the peaks are not equidistant or when the distance is estimated incorrectly. These observations indicate random immigrants mechanism as more efficient and flexible, and therefore more useful for further experiments and comparisons with redundant genetic material strategy.

7.2 Population behavior during experiments

Results of experiments presented below show that presence of memory can increase the algorithm's efficiency although this is not a straight dependency. In some cases enlargement of memory buffers does not give better results. It is very important what was remembered in the memory. The contents depends of the length of time interval between changes, type of changes, etc.

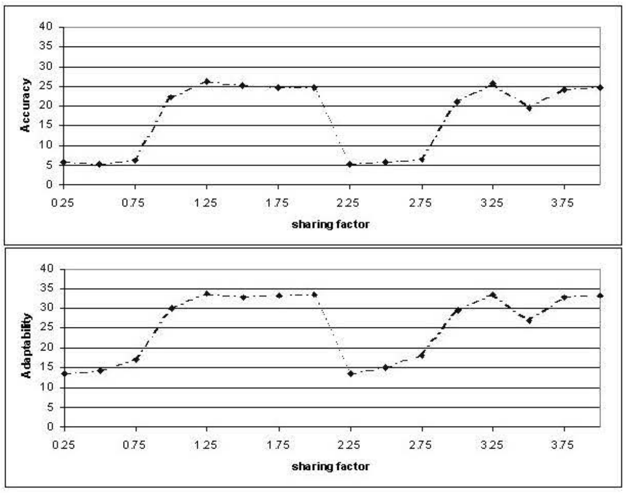


Figure 5: Results of experiments with sharing for environment no. 1 — Accuracy and Adaptability

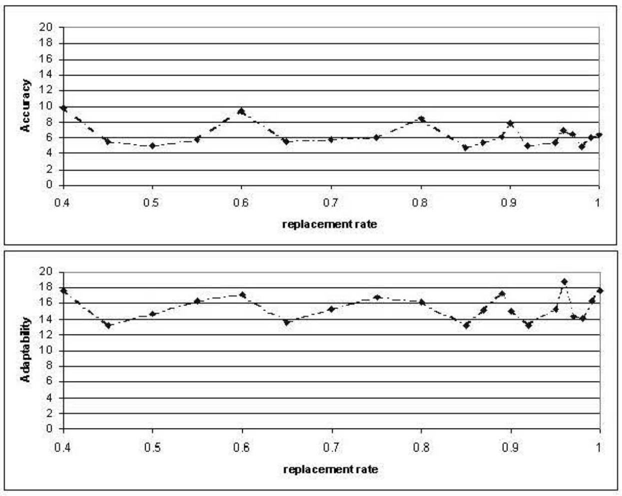


Figure 6: Results of experiments with random immigrants for environment no. 1 — Accuracy and Adaptability

Below an influence of memory on behavior of individuals in the population is discussed. It is illustrated on the experiments for the environment no. 1 from previous section and for two types of evolutionary algorithm extensions: random immigrants mechanism and redundant genetic material approach. Matrix H_a in the experimental environment has the values as in the environment no. 1 in previous experiments:

$$\begin{bmatrix} 0.100 & \mathbf{0.146} & \mathbf{0.000} & 0.100 \\ \mathbf{0.500} & 0.100 & 0.100 & \mathbf{0.146} \\ \mathbf{0.546} & 0.100 & 0.100 & \mathbf{0.500} \\ 0.100 & \mathbf{1.000} & \mathbf{0.546} & 0.100 \end{bmatrix}$$

and was changed identically.

Four versions of evolutionary algorithms are discussed for such an environment: a pure evolutionary algorithm, evolutionary algorithm with a memory, evolutionary algorithm with random immigrants mechanism, and evolutionary algorithm with memory *and* random immigrants mechanism.

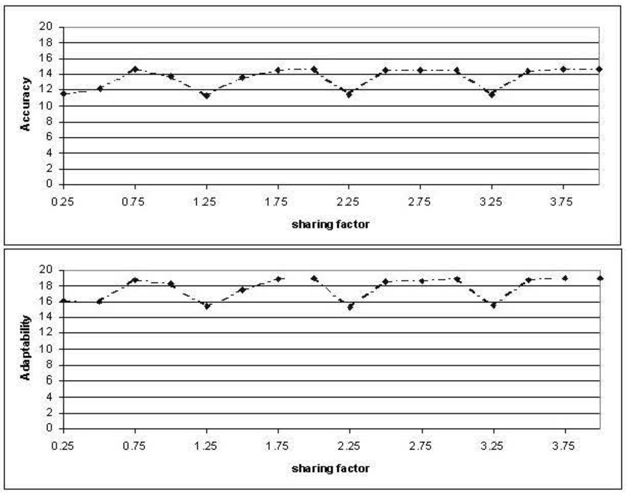


Figure 7: Results of experiments with sharing for environment no. 2 — Accuracy and Adaptability

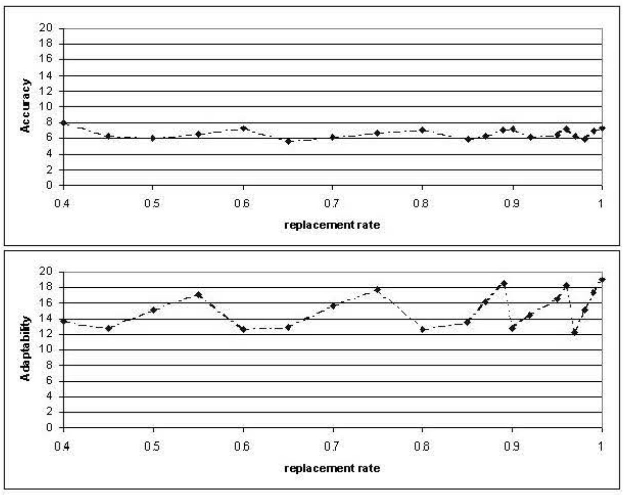


Figure 8: Results of experiments with random immigrants for environment no. 2 — Accuracy and Adaptability

In the first case, an evolutionary algorithm was not enriched by any additional mechanisms dedicated to non-stationary environments. In the second case individuals were equipped with memory buffers of size 20. The memory was employed in case of changes in the environment. The third case included experiments with alternative extension of the algorithm. It was a diversity maintenance technique of random immigrants mechanism. It was applied to the population just after the change in the environment. 85% of population was exchanged average (random immigrants mechanism is controlled by a *replacement rate* which defines the probability of change of an individual into randomly generated another one). In the fourth case algorithm was enriched by two mechanisms: memory and the random immigrants mechanism. After the change in the environment the memory was first recalled. Next, active chromosomes in the individuals were modified by random immigrants mechanism. A modified individual has an active chromosome changed but the chromosome in the memory structures are left unchanged.

It is interesting to examine the distribution of individuals in the population just after a change is made in the landscape. Note that the positions of individuals after the change influence Adaptability (or the number of generations in which the algorithm finds the optimum or/and the closeness of the best individual to the optimum).

Figures 9–12 present typical snapshots (two snapshots per case) of the population just after the change (these snapshots were made after a few cycles of changes already went through). Individuals are visualized as small black diamonds. A triangle represents new coordinates of the optimum just after change.

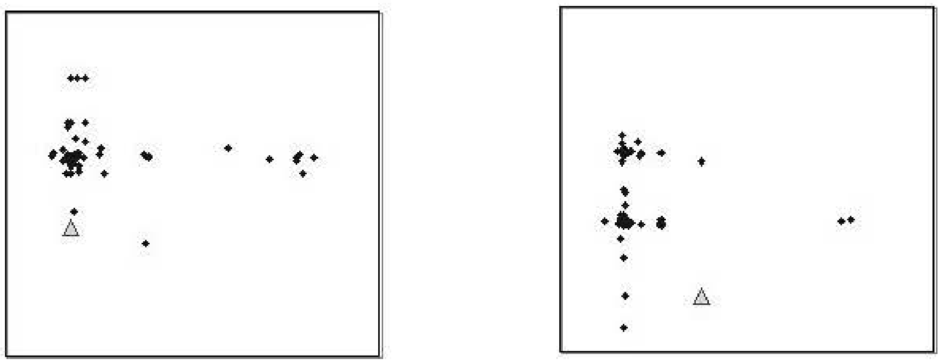


Figure 9: Two views of the population just after the change in the environment no. 1 for the algorithm without memory and without random immigrants mechanism.

Figure 9 represents the case where algorithm was not extended by any additional mechanisms. It can be seen that the population is clustered around the points which were optima before the change. Sometimes they are clustered around two points because only a part of population could manage to migrate to the new area of space since the previous change. On every snapshot, there is a small subset of individuals distributed in the whole space. This set of individuals has a very significant role in looking for new coordinates of global optimum. They are exploited in searching through other areas of space. The population on the left has better situation than the one on the right. One of individuals is quite close to the new optimum and it can attract the rest of the population to the new area of the search space. On the right hand picture, there are no individuals around the new optimum.

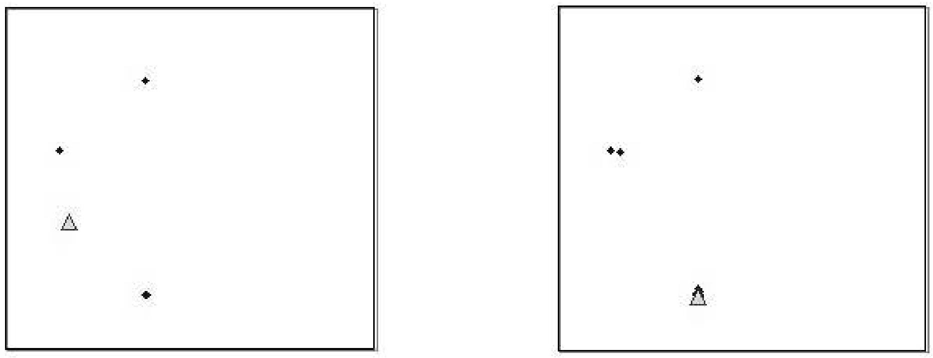


Figure 10: Two views of the population just after the change in the environment no. 1 for the algorithm with memory but without random immigrants mechanism.

Figure 10 represents the case where the algorithm is equipped with memory structures. Here the population is also clustered but it is clustered around the new point of optimum while in the previous case — around the points which were global optima quite a few changes ago. This is the result of presence of some solutions in the memory structures which were collected during the search process till now. They mostly represent good solutions from previous positions of optimum. It happens that some positions are not represented in the memory. Then the other good local optimum is recalled. Anyway the memory always has something (better or worse) to prompt and if the current values of individuals are worse than the values of remembered ones then the population is moved to new positions. This explains why in the snapshot on the left the population is not clustered around the current global optimum but around the point which was the optimum a few changes ago and now is one of local optima only.

The other difference is in the absence of the small set of individuals distributed in the search space and that most of individuals are similar or identical to each other. This low diversity of population decreases potentiality of searching through the space. The whole responsibility and hope for skipping by the population to the other area of space and finding the new position of optimum is taken by the mutation operator which now has to hit this position or at least close around it.

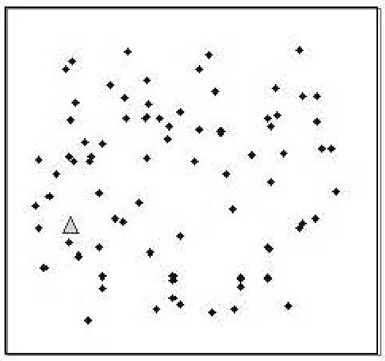
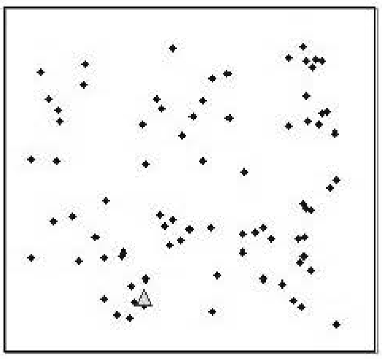



Figure 11: Two views of the population just after the change in the environment no. 1 for the algorithm without memory but with the random immigrants mechanism.

Figure 11 represents a case where the algorithm was equipped by the random immigrants mechanism. Here the population consists of diverse individuals representing solutions from all areas of search space but they have no information about new coordinates of optimum. Random immigrants mechanism is a good mechanism for small changes where new coordinates of optimum are close to the previous ones or where the shape of evaluation function is not very complex (e.g., for environments with one optimum only or with many local optima but without any complex changes during the search process). Otherwise the individuals which were not mutated do not represent any significant information and then we start the search of a new optimum from scratch in fact.

The last case is presented in the Figure 12. Here the algorithm was extended by two mechanisms: memory and random immigrants mechanism. These snapshots are very similar to the ones presented in the Figure 11. The difference between the algorithms is visible much better in the Figure 13 in graphs (c) and (d) showing obtained results of Accuracy — a difference between the value of the current best individual in the population of the "just before the change" generation and the optimum value.

Here good traits of both mechanisms made a well co operating pair. A part of population is clustered around the coordinates of the optimum taken from memory. The rest of them are distributed in the whole search space to speed up searching in case when the remembered optimum is not a current global optimum.

Observations from the figures are confirmed by the values of Accuracy for these four experiments. Graphs of a difference between the value of the current best individual in the population of the "just before the change" generation and the optimum value are presented in the Figure 13.

Y-axis represents the difference between the value of the current best individual and the current optimum value (the smaller is the better) and X-axis is the number of change. There were 8 changes in the cycle so we can see 6 full cycles of changes in the graphs. There are four graphs for four experiment discussed above:

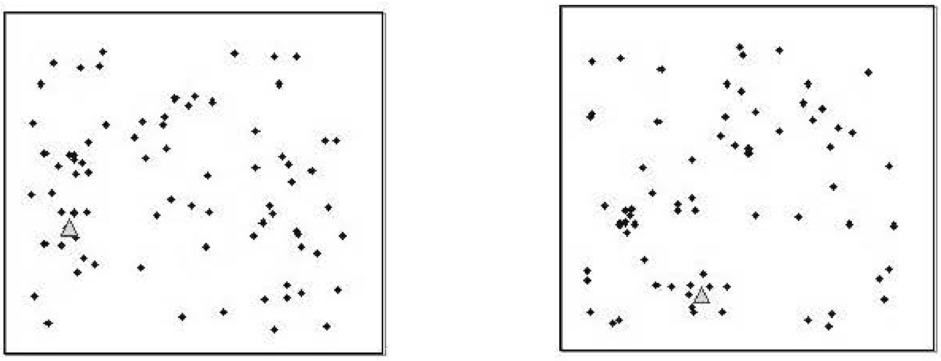


Figure 12: Two views of the population just after the change in the environment no. 1 for the algorithm with memory and with random immigrants mechanism.

- **a** — pure evolutionary algorithm (m- ri-),
- **b** — evolutionary algorithm with memory (m+ ri-),
- **c** — evolutionary algorithm with random immigrants mechanism (m- ri+), and
- **d** — evolutionary algorithm with memory and random immigrants mechanism (m+ ri–).

Note that for better visibility of the results the scale on the Y-axis for the graphs (a) and (b) is different than for the graphs (c) and (d).

In the graph (a) we have lots of peaks representing situations where algorithm could not find the new coordinates of optimum and the population clustered around any other local optimum. In the graph (b) there is smaller number of peaks than in graph (a). Memory helped to find new coordinates of optimum but not for every change. In cases where the new coordinates were not remembered the memory prompted other solutions better than the current values of individuals but which were not clustered around the global optimum. In the graph (c) the values are various. For some of the changes the algorithm finds an optimum while for the others does not or hits somewhere around the optimum only. There is no correlation between obtained values of the best individual and the coordinates of optimum point in the cycle. In the graph (d) there is a long sequence of hits very close to optimum. It is the best graph of those four. The value of Adaptability for this type of algorithms is also the best (see Table 2).

Table 2: Experimental results for the environment no. 1 for four versions of evolutionary algorithm and for restart strategy — Accuracy and Adaptability

	Accuracy	Adaptability
m- ri-	23.5958	32.5576
m+ ri-	16.9635	25.0845
m- ri+	4.8435	14.0464
m+ ri+	2.4375	7.1326
restart	6.5772	18.1417

For comparisons the values of Accuracy and Adaptability evaluated as an average values from a sequence of 50 experiments are presented in the Table 2. In the last row of the table, we have values for an experiment where after every change the whole population was exchanged and the algorithm started really

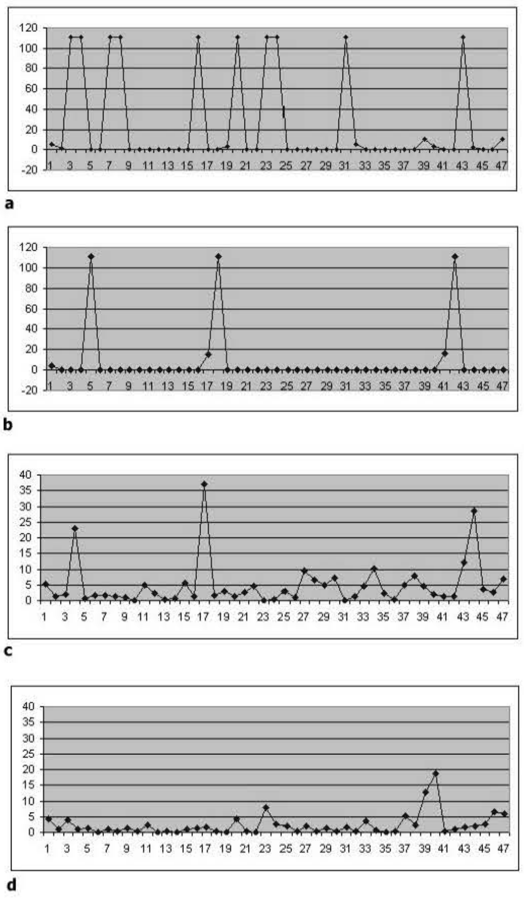


Figure 13: Values of the difference between the best individual in the population and the value of optimum solution just before the change in the environment. (a) algorithm without memory and without random immigrants mechanism, (b) algorithm with memory but without random immigrants mechanism, (c) algorithm without memory but with random immigrants mechanism, (d) algorithm with memory and with random immigrants mechanism. Scale on the Y-axis for the graphs (a) and (b) is different than for the graphs (c) and (d).

from scratch. The value of Accuracy in this row is worse than the value of Accuracy for experiments with random immigrants mechanism. That means that the small unchanged part of population left from generations before change has good influence on the efficiency of the algorithm.

7.3 Results for different types of changes

In the experiments discussed below, we studied dependencies between type of changes and the efficiency of a few strategies of evolutionary algorithms dedicated to non-stationary environments optimization. We compared and analyzed results of experiments on similar environments with cyclic and non-cyclic changes. All the results in tables in this subsection are the average values of 50 experiments. In cases with cyclic type of changes each experiment consisted of at least 6 cycles of changes. We made every experiment for four values of τ: 5, 10, 20 and 40 generations.

All environments were created by test-case generator and were based on 2-dimensional search space which consisted of 16 cubes (4 by 4). The differences between environments were in the selection of set of non-stationary cubes. Three different shapes of paths of changes were compared: (1) random (non-cyclic changes), (2) circle (cyclic changes), and (3) line (cyclic changes).

In the first of compared cases the shape of path was generated randomly at every change, i.e. the set of non-stationary cubes was not constant but changed in time. Every change a new set of 8 cubes was randomly selected, which was initialised with 8 following values:

0.000 0.146 0.500 0.546 1.000 0.546 0.500 0.146

In the rest of cubes the heights were set to the standard value which was ten times smaller than the height of global optimum over the entire search space. The number of non-stationary cubes in the environment was not static but changed in time. It was at least 8 cubes (if the set of selected cubes was the same as previous one) and at most 16 cubes (if the newly selected set was completely different than the previous one.

In the second environment, the path of changes was fixed. During every change, the same set of cubes was modified. Unmodified cubes had the same height which was — as in previous case — ten times smaller than the height of global optimum. The path of changes is presented in the matrix in Figure 14(a) where values of non-stationary cells are displayed with bold letters. These are initial values of the cells for time $t = 0$. They were moved anticlockwise by one position after every τ generations (it is the same environment as in the previous section).

$$(a)\ \begin{bmatrix} 0.100 & \mathbf{0.146} & \mathbf{0.000} & 0.100 \\ \mathbf{0.500} & 0.100 & 0.100 & \mathbf{0.146} \\ \mathbf{0.546} & 0.100 & 0.100 & \mathbf{0.500} \\ 0.100 & \mathbf{1.000} & \mathbf{0.546} & 0.100 \end{bmatrix}$$

$$(b)\ \begin{bmatrix} 0.100 & 0.100 & 0.100 & \mathbf{0.500} \\ 0.100 & 0.100 & \mathbf{0.000} & 0.100 \\ 0.100 & \mathbf{0.500} & 0.100 & 0.100 \\ \mathbf{1.000} & 0.100 & 0.100 & 0.100 \end{bmatrix}$$

Figure 14: Matrices H_a for compared environments no. 2 (matrix a), and no. 3 (matrix b). Non-stationary cubes are listed in boldface.

The last environment also had a fixed path of changes. The main difference between the previous environment and the current one was that the last cube in the path is not a neighbor of the first one — the path is a segment. The optimum from the last cube in the top right hand corner was moved to the bottom left hand corner. This was the case where the length of jumps of optimum was not constant. The shape of the path and its initial values are presented in the matrix in Figure 14(b).

For these three environments experiments were performed. We tested two types of algorithm extensions: the first — with memory and the second — with memory and random immigrants mechanism. We compared results for four sizes of the time interval between changes and for five sizes of the individual memory buffer. The *replacement rate* was 0.85.

Values $a_{\overline{V}}$ for stationary cubes were selected to be equal to 0.1 because we wanted to force the population to active search through the search space for the global optimum instead of staying around one of quite good stationary local optimum.

(A) Cyclic changes

In presented results (Tables 3, 4, 5, and 6 — environments no. 2 and 3) in most cases we can see positive impact of the presence of memory structures for environments with cyclic type of changes. The difference between the value of optimum and the value of the best individual in the population decreased

Table 3: Experimental results for three environments for $\tau = 5$ — Accuracy and Adaptability

	Accuracy					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	12.2035	4.4032	23.5958	4.8435	16.8089	4.2657
m= 1	13.3311	5.0800	20.3702	4.5661	24.0580	4.4577
m=10	14.5487	3.0935	19.3711	2.4851	26.7077	3.0737
m=20	12.3776	2.7568	16.9635	2.4375	12.1999	1.1559
m=40	9.3754	2.4043	12.0400	1.6720	6.9340	0.5551

	Adaptability					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	20.6059	14.1339	32.5576	14.0464	33.3517	13.9904
m= 1	22.4394	13.8237	29.7868	12.3413	41.1585	13.9146
m=10	20.6657	9.7435	27.1076	7.6566	39.5565	10.9795
m=20	17.0614	7.5859	25.0845	7.1326	18.1399	4.7926
m=40	13.1688	6.3277	17.9237	5.2074	10.1615	2.3838

Table 4: Experimental results for three environments for $\tau = 10$ — Accuracy above and Adaptability below

	Accuracy					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	9.1411	1.0683	7.1380	1.3284	9.7249	0.7337
m= 1	8.5526	1.1044	7.5751	0.9352	9.9857	0.6829
m=10	11.8247	1.1488	7.4025	0.6473	11.1865	0.7859
m=20	12.5128	0.9438	5.8378	0.8665	9.1448	0.5011
m=40	10.7871	0.8758	5.9347	0.6986	2.4629	0.1135

	Adaptability					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	20.1494	8.9553	19.3574	8.4935	35.6773	9.1151
m= 1	20.7621	8.0485	20.3126	7.6179	38.6913	9.3523
m=10	23.2790	7.5933	19.0508	5.8161	36.9165	8.6378
m=20	21.0737	6.1645	15.8290	5.8761	29.0826	6.8413
m=40	17.9509	5.3498	14.6871	4.9156	8.2435	2.8945

for experiments with larger memory buffer. Memory held good individuals collected there during the search process. The larger the memory was, the better results were obtained. Simultaneous application of memory and random immigrants mechanism additionally improved obtained results.

In the obtained results two additional effects were observed:

— local worsening: An effect of results worsening appeared for environments with cyclic type of changes

Table 5: Experimental results for three environments for $\tau = 20$ — Accuracy above and Adaptability below

	Accuracy					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	8.7503	0.4112	0.9885	0.0340	1.6311	0.0209
m= 1	8.2180	0.2371	1.3290	0.2132	1.2154	0.0232
m=10	7.6998	0.4157	1.4106	0.1027	2.3956	0.0152
m=20	8.2422	0.5027	1.2419	0.0137	0.9071	0.0128
m=40	8.3188	0.7726	1.0899	0.0596	1.2996	0.0130

	Adaptability					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	18.3924	4.3193	12.0455	4.0082	21.8467	4.7305
m= 1	18.9217	4.0209	12.4993	3.9327	22.6649	4.9852
m=10	18.4058	4.3389	12.4713	3.5715	23.6661	5.1084
m=20	18.5685	3.9488	11.9974	3.4459	21.9957	5.0748
m=40	17.0796	3.8599	9.6934	2.9409	18.5041	4.4567

Table 6: Experimental results for three environments for $\tau = 40$ — Accuracy above and Adaptability below

	Accuracy					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	4.3399	0.2888	0.0943	0.0000	0.0002	0.0000
m= 1	5.6473	0.1927	0.0118	0.0000	0.0000	0.0000
m=10	5.1178	0.2886	0.0281	0.0000	0.0000	0.0000
m=20	5.4507	0.2889	0.0003	0.0000	0.0001	0.0000
m=40	4.5507	0.0961	0.0001	0.0000	0.0003	0.0000

	Adaptability					
	e.1		e.2		e.3	
	ri-	ri+	ri-	ri+	ri-	ri+
m= 0	11.7376	2.2605	5.9614	1.9314	11.2836	2.3877
m= 1	13.1919	2.2180	6.6661	1.7968	11.3569	2.6079
m=10	13.3908	2.2810	5.6996	1.7700	11.2915	2.4717
m=20	13.0291	2.3220	5.5330	1.8055	11.5583	2.5264
m=40	12.7945	1.9874	5.9390	1.9267	10.9274	2.5222

where the memory buffer of a small size was applied (Tables 3, 4, and 5 — environment no. 3). But the continuous increase of memory buffer size leaded to improvement of the results. The effect of local worsening disappeared for experiments with longer period of time between changes. It was compensated by the exploring properties of evolutionary algorithm which had more time to find a global optimum in spite of unfavorable starting distribution of individuals in the population.

— long period between changes: It can be seen (Table 6) that for all environments 40 generations is enough time to find an optimum for the algorithm with or without memory. The Accuracy is very close to zero, as there is almost no difference between the optimum and the best individual in the population.

(B) Non-cyclic changes

For non-cyclic changes random immigrants mechanism improved obtained results but memory structures worsened them (Tables 3, 4, 5, and 6 — environment no. 1).

This is a case where the memory should not be applied. The reason of this new worse behavior of the algorithm is a particular property of memory mechanism. The memory can hold references to many different points of space which were optima in previous changes but not to all of them. When optimum is moved to the position which is absent among remembered solutions memory is still trying to prompt something. When the remembered position is better than the current one the individual is moved to that point immediately even if it is not a global optimum but one from the set of local ones.

For environment with non-cyclic changes there was also an improvement of results for longer periods between changes which means that algorithm was still trying to find an optimum during the given time of static shape of environment.

8 Non-stationary constraints — results

All the experiments presented in this section were performed for the class of environments where the evaluation function landscape is static in time but the set of constraints exists and changes in time.

To introduce constraints into the problem we extended the definition of an environment by a set of constraint inequalities:

$$c_j \geq 0 \qquad j = 1, \ldots, v.$$

They divided the search space into feasible and infeasible regions. We wanted the regions to be changed during the search process so we changed constraint inequalities, i.e., $c_j = c_j(\vec{x}, t)$. Changes were performed in the environment periodically: every τ generations.

Our aim was to create demanding environments with non-stationary constraints where the global optimum coordinates change in time although the evaluation function is static. To achieve this we carefully selected constraint inequalities formulas and values in matrices H_a defining evaluation function landscape. They were generated to modify the global optimum coordinates at every change. After every change a new better feasible solution appeared or the current global optimum was turned into infeasible one. Additionally to make the optimization task more difficult the current global optimum was always situated on the boundary of the feasible region.

In our research we added some extensions to the evolutionary algorithm and observed how the extensions can influence on the obtained results. We made experiments with two types of extensions: random immigrants mechanism and individual level memory structures.

We made experiments with cyclic and non-cyclic changes of the feasible part of the search space. All of them were performed for a set of linear constraints.

In cyclic changes a single cycle consisted of several changes after which the shape of the feasible and infeasible regions was the same as at the beginning. For example at the beginning of the search process following changes of the feasible region of the search space decreased the size of this region. After some changes the region was small and in the next changes started to increase its size. When the feasible part was as at the beginning one *cycle* of changes was completed. We repeated the cycles of changes a few times in every experiment.

Additionally we made some experiments with non-cyclic changes decreasing the size of the feasible region of the search space. In that case only a half cycle of changes described above was performed which only decreased the size of the feasible region.

The experiments and the results are discussed below. All the results in tables are the average values of 50 experiments. In cases with cyclic type of changes one experiment consisted of at least 6 cycles of changes. We made every experiment for four values of τ: 5, 10, 20 and 40 generations and for five different sizes of memory buffer: 0, 1, 10, 20 and 40 entries.

8.1 Testing environments

We made experiments with four different environments with non-stationary set of constraints dividing the search space into feasible and infeasible regions. Feasible regions in all experiments were represented by convex areas.

Environment A is a simple case with only one constraining inequality where the full cycle of changes consists of 10 steps. The size of the feasible area changes from 1/8 (12,5%) to 7/8 (87,5%) of the search space. Environment B is defined with two constraining inequalities and full cycle of changes consists of 14 steps The size of the feasible area changes from 15/128 (11,7%) to 123/128 (96%) of the search space. Environment C is also defined with two constraining inequalities and full cycle of changes is the longest and takes 16 steps. The size of the feasible area changes from 49/256 (19,1%) to 123/128 (96%) of the search space. Environment D represents more difficult case where the feasible part of the search space is represented by two disjoined regions. A full cycle of changes for the environment D consists of 14 steps and the size of feasible area changes from 5/128 (3,9%) to 123/128 (96%) of the search space.

Matrices H_a and constraint inequalities for these environments are presented in Figures 15, 17, 19, and 21. Example views of the search space with marked feasible and infeasible regions of the search space are presented in Figures 16, 18, 20, and 22.

The variable t' in constraint inequalities represents the number of change. For a given t-th generation we can evaluate the number of last performed change with the following formula:

$$t' = \frac{t - (t \bmod \tau)}{\tau}$$

where t is the current number of generation. All the experiments are based on 2-dimensional search space consisted of 16 cubes (4 by 4).

$$\begin{bmatrix} 1 & 1 & \mathbf{4}_2 & 1 \\ \mathbf{2}_1 & 1 & 1 & \mathbf{8}_4 \\ 1 & \mathbf{6}_3 & 1 & 1 \\ 1 & 1 & \mathbf{10}_5 & 1 \end{bmatrix} \qquad x_2 \geq x_1 + \frac{-2+|(t' \bmod 9)-4|}{w}$$

Figure 15: Matrix H_a and constraint inequality for environment A. Values $a_{\vec{V}}$ of cells being global optimums for following changes of constraints are in boldface. Indices at these values indicate order of changes for increasing feasible part of the search space.

Figures of values of a feasible part of the search space capacity ratio ρ and the gradient of change $\mathcal{G}(t)$ for experiments with environments A, B, C, and D are presented in Figure 23.

8.2 Results of experiments with cyclic changes

Results of experiments with cyclic changes of the feasible part of the search space presented in Tables 7, 8, and 9 show that memory structures can increase the algorithm's efficiency but for small values of τ

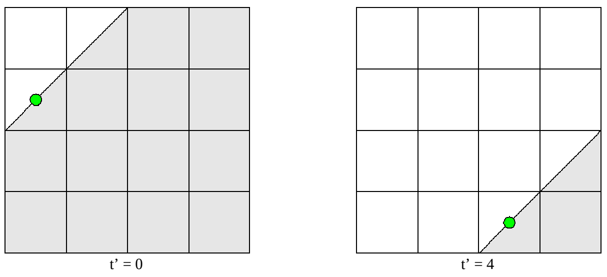


Figure 16: Two views with minimum and maximum size of the infeasible part of the search space of environment A. The infeasible region is marked as a grey area. A current global optimum is represented by a circle.

$$\begin{bmatrix} \mathbf{8.57}_2 & 1.00 & 1.00 & \mathbf{2.00}_7 \\ \mathbf{5.71}_4 & \mathbf{2.86}_6 & 1.00 & 1.00 \\ 1.00 & 1.00 & \mathbf{4.28}_5 & \mathbf{7.14}_3 \\ 1.00 & 1.00 & 1.00 & \mathbf{10.00}_1 \end{bmatrix} \quad \begin{cases} c_1(\vec{x}) : x_2 \geq x_1 - \frac{|(t' \mathrm{mod} 13)-6|+1}{2w} \\ c_2(\vec{x}) : x_2 \leq x_1 + \frac{|(t' \mathrm{mod} 13)-6|}{2w} \\ c(\vec{x}) = c_1(\vec{x}) \wedge c_2(\vec{x}) \end{cases}$$

Figure 17: Matrix H_a and constraint inequality for environment B. Values $a_{\vec{V}}$ of cells being global optimums for following changes of constraints are in boldface. Indices at these values indicate order of changes for decreasing feasible part of the search space.

only. In every table results for $\tau = 5$ are better for algorithm with memory structures than for algorithm without them. Although with increasing value of τ the influence of memory changes and it rather decreases the quality of obtained results (apart from the environment C where the improvement is observed for all values of τ).

As in experiments with non-stationary evaluation function the results of experiments with random immigrants mechanism were better than the results without it.

In table 10 with results of environment D with random immigrants mechanism it can be seen that one of Accuracy results is extremely large. The reason of this was the situation where none of individuals was feasible and an infeasible individual was evaluated as the best. This indicates that although the results of the algorithm with random immigrants extension are better the risk of loosing feasible part of the search

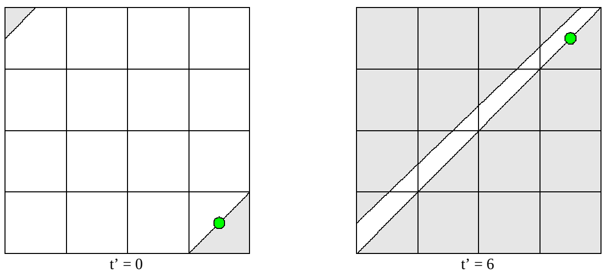


Figure 18: Two views with minimum and maximum size of the infeasible part of the search space of environment B. The infeasible region is marked as a grey area. A current global optimum is represented by a circle.

$$\begin{bmatrix} \mathbf{18.75}_1 & \mathbf{13.75}_3 & 1.00 & \mathbf{3.75}_7 \\ 1.00 & \mathbf{8.75}_5 & 1.00 & 1.00 \\ \mathbf{11.25}_4 & 1.00 & \mathbf{2.50}_8 & 1.00 \\ \mathbf{16.25}_2 & 1.00 & \mathbf{6.25}_6 & 1.00 \end{bmatrix} \quad \begin{cases} c_1(\vec{x}) : x_2 \geq x_1 + \frac{|(t' \mathrm{mod} 15)-7|-1}{2w} \\ c_2(\vec{x}) : x_2 \leq -x_1 + \frac{7-(|(t' \mathrm{mod} 15)-7|-1)}{2w} \\ c(\vec{x}) = c_1(\vec{x}) \wedge c_2(\vec{x}) \end{cases}$$

Figure 19: Matrix H_a and constraint inequality for environment C. Values $a_{\vec{V}}$ of cells being global optimums for following changes of constraints are in boldface. Indices at these values indicate order of changes for decreasing feasible part of the search space.

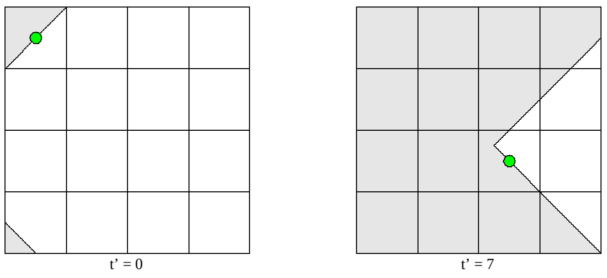


Figure 20: Two views with minimum and maximum size of the infeasible part of the search space of environment C. The infeasible region is marked as a grey area. A current global optimum is represented by a circle.

space is growing too. The algorithm could have some troubles especially when the feasible part has complex shape as in the environment D where the feasible part consisted of two regions.

8.3 Results of experiments with non-cyclic changes

The results of experiments with changes decreasing the feasible part of the search space for the environment B and C are presented in Tables 11 and 12.

As in previous experiments with cyclic changes obtained results showed that presence of memory can increase the algorithm's efficiency but for small values of τ only. Additionally in both cases, we can see an effect of local worsening. This effect is presented in experiments with small size of memory buffer, when some valuable individuals are quickly forgotten. These are individuals which are diversified over the entire search space and they were generated for the initial population. Instead of them other group of

$$\begin{bmatrix} \mathbf{2.86}_2 & 1.00 & 1.00 & \mathbf{10.00}_7 \\ \mathbf{5.71}_4 & \mathbf{8.57}_6 & 1.00 & 1.00 \\ 1.00 & 1.00 & \mathbf{7.14}_5 & \mathbf{4.28}_3 \\ 1.00 & 1.00 & 1.00 & \mathbf{2.00}_1 \end{bmatrix} \quad \begin{cases} c_1(\vec{x}) : x_2 \geq x_1 + \frac{|(t' \mathrm{mod} 13)-6|+1}{2w} \\ c_2(\vec{x}) : x_2 \leq x_1 - \frac{|(t' \mathrm{mod} 13)-6|}{2w} \\ c(\vec{x}) = c_1(\vec{x}) \vee c_2(\vec{x}) \end{cases}$$

Figure 21: Matrix H_a and constraint inequality for environment D. Values $a_{\vec{V}}$ of cells being global optimums for following changes of constraints are in boldface. Indices at these values indicate order of changes for increasing feasible part of the search space.

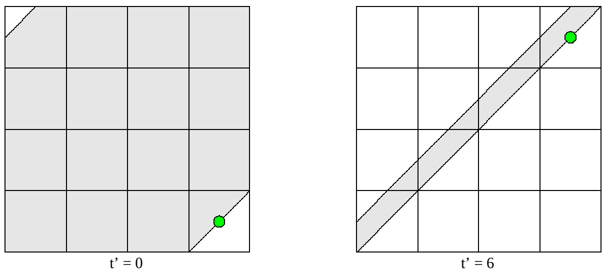


Figure 22: Two views with minimum and maximum size of the infeasible part of the search space of environment D. The infeasible region is marked as a grey area. A current global optimum is represented by a circle.

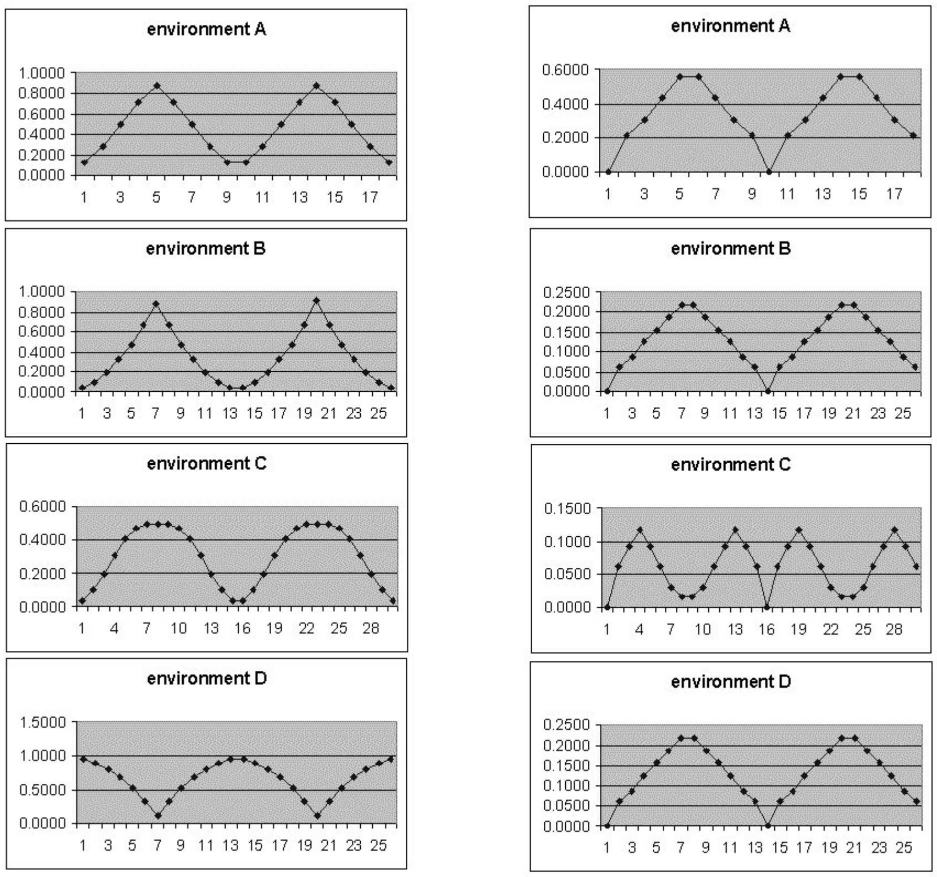


Figure 23: Values of a feasible part of the search space capacity ratio ρ (on the left), and of the gradient of change $\mathcal{G}(t)$ (on the right) for experiments with environments A, B, C, and D.

individuals is remembered which is converged around lately found optima. Those optima are infeasible when the memory is recalled because every change the feasible part of the search space is decreasing. Such an recalled group of individuals certainly does not improve the algorithm.

The rest of results confirmed previous observation that both extensions are better than only one of them.

Table 7: Experimental results for environment A with cyclic changes — *without* (upper tables) and *with* (lower tables) random immigrants mechanism

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	24.1720	19.1523	17.1165	16.0686	25.5360	22.6059	30.2336	22.6825
m= 1	23.2413	18.5917	17.1412	15.7720	31.7588	53.7640	26.8776	22.7757
m=10	21.9088	19.2049	18.1388	16.3769	23.8949	30.1423	22.2735	23.1781
m=20	21.2503	18.4504	17.9351	16.2385	22.9793	22.5333	22.0134	21.1899
m=40	18.0279	18.5105	18.7428	17.1618	40.7112	21.7120	22.6240	21.3980

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	12.9722	7.9036	6.4823	6.2395	18.8547	13.5586	10.6246	8.7765
m= 1	13.1297	8.8261	6.6658	7.0289	18.4213	13.7494	10.5950	9.4034
m=10	12.6759	8.9501	7.4465	6.8665	16.6713	13.3880	11.0038	9.1419
m=20	12.0111	9.4362	7.7467	6.6848	15.6170	13.7554	11.3908	9.2466
m=40	9.4184	8.8338	8.3457	7.2639	12.5244	12.5166	11.5241	9.6738

Table 8: Experimental results for environment B with cyclic changes — *without* (upper tables) and *with* (lower tables) random immigrants mechanism

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	7.4496	4.2754	4.0610	4.5950	11.3207	7.7606	7.1268	7.2998
m= 1	6.6640	3.9652	4.5315	5.0115	9.9375	7.4372	7.3130	7.6557
m=10	5.7606	4.1328	4.7764	5.0925	8.3011	6.7484	7.4557	7.7017
m=20	6.1377	4.6582	4.8754	5.1435	7.9273	6.6384	7.2656	7.6726
m=40	5.4937	5.8597	4.7298	5.2549	6.9645	7.2683	6.9454	7.5965

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	7.7580	3.2530	2.1220	1.7090	14.2243	8.7411	5.8299	4.2685
m= 1	7.0594	3.2330	2.2026	1.8089	13.4167	8.3765	5.9718	4.2687
m=10	5.7453	3.3573	2.3455	1.8552	10.6468	7.5932	5.8947	4.2301
m=20	4.6894	3.1700	2.4481	2.0204	9.0455	7.2969	5.8159	4.3425
m=40	4.1220	2.4613	2.1828	1.9683	7.7490	5.8439	5.2849	4.2563

9 Summary

In this section we discussed experiments with non-stationary environments.We tested a few strategies which include or exclude a diversity maintenance mechanism and a redundant genetic material. We made two main groups of experiments:

- The evaluation function landscape changed in time and the set of constraints was empty (the class no. 4 from classification proposed in Section 2),

Table 9: Experimental results for environment C with cyclic changes — *without* (upper tables) and *with* (lower tables) random immigrants mechanism

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	26.0563	20.0799	15.9908	11.5186	32.2006	26.2790	21.5202	17.0339
m= 1	27.3530	19.2586	15.9746	11.2060	33.3356	25.2621	21.4723	16.4425
m=10	24.7618	19.6083	15.3293	11.3702	29.9558	24.8508	20.1568	16.1374
m=20	23.6284	19.0728	15.4956	11.1010	28.3897	24.0946	20.5401	16.0839
m=40	23.7549	17.6008	15.4634	11.9392	28.0348	22.3615	20.2813	16.4780

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	17.7694	10.1137	5.9728	4.3132	27.4256	18.4195	12.7544	9.4236
m= 1	16.6335	10.4913	6.6597	3.6538	25.4761	17.8397	12.8481	8.5812
m=10	14.5487	9.5082	6.5060	4.5299	21.1519	16.0133	12.3483	9.2125
m=20	13.3011	9.1219	6.6091	4.5265	19.0526	14.8654	12.0144	9.1199
m=40	11.5403	8.5713	6.3925	4.9001	16.6594	13.7780	11.4563	9.0804

Table 10: Experimental results for environment D with cyclic changes — *without* (upper tables) and *with* (lower tables) random immigrants mechanism

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	24.8662	18.5187	14.8566	15.6481	33.1594	37.8718	30.0428	27.2793
m= 1	22.4651	17.8735	15.9541	14.7989	40.0738	33.9599	24.4282	21.0898
m=10	20.8169	18.1240	16.6639	14.8439	131.4890	28.0258	27.9650	19.5137
m=20	20.2965	18.5505	16.6420	14.3067	76.1384	37.8050	26.1793	17.8270
m=40	18.8820	19.1804	17.0707	14.6443	69.3086	32.7580	19.4330	19.1167

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	9.8023	5.0431	3.1259	2.3849	24.6873	32.0076	17.8759	11.7471
m= 1	9.3574	4.8953	2.9487	2.4292	13.7163	26.5154	12.6240	11.0568
m=10	7.2801	4.5890	3.1629	2.5099	15.8527	18.0866	22.3021	11.6146
m=20	6.8510	4.5673	3.7115	29.8252	14.9644	15.3360	31.2635	35.5059
m=40	5.6726	4.3908	3.4566	2.8493	18.4273	19.8187	16.2057	11.8980

- The evaluation function landscape was static in time but the set of constraints existed and changed in time (the class no. 3).

Section 7 described experiments with non-stationary evaluation function. We compared the behavior of population for the evolutionary algorithm with and without two different extensions dedicated to non-stationary problem optimization. We made experiments in one environment with cyclical changes of optimum coordinates. The results given by two proposed measures, Accuracy and Adaptability, and observations of individuals' distribution on the search space indicated that each of tested strategies controlled and improved

Table 11: Experimental results for environment B with non-cyclic changes — *without* (upper tables) and *with* (lower tables) random immigrants mechanism

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	6.3478	3.1386	0.9861	0.1652	10.8374	8.8748	6.2803	3.5082
m= 1	6.8960	3.6968	1.2465	0.4939	11.4571	9.9673	5.8690	3.9523
m=10	6.0248	3.7881	1.4946	0.2530	10.3831	9.3391	6.1346	3.8501
m=20	6.2169	4.1387	1.2193	0.3683	9.9083	9.8351	7.1102	4.3121
m=40	5.6419	3.2686	2.6687	0.7304	9.8175	8.4998	6.8090	4.1014

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	7.8964	2.7562	0.7139	0.6301	13.1268	8.9695	4.8759	3.2122
m= 1	7.4561	3.2197	1.6083	0.8639	12.6942	7.9575	5.6525	3.3344
m=10	6.1271	2.5709	0.9843	0.5742	10.4505	7.3667	5.0346	3.4608
m=20	5.8578	2.5063	1.2512	0.2739	9.0887	6.5624	5.2143	3.0250
m=40	5.2874	2.6441	1.9176	0.8469	8.7712	6.6222	5.0837	3.4227

Table 12: Experimental results for environment C with non-cyclic changes — *without* (upper tables) and *with* (lower tables) random immigrants mechanism

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	14.3863	8.3151	7.3330	3.4545	20.8555	15.1931	12.0690	7.5415
m= 1	15.8537	8.2427	6.3611	6.4486	22.0372	14.9632	11.5217	9.9522
m=10	14.4098	7.2456	7.1221	3.2806	20.2487	13.9904	11.7496	7.6266
m=20	13.6556	8.5994	5.3471	5.1406	19.4796	15.0338	9.6536	9.2481
m=40	10.8245	9.5401	6.3716	4.5104	15.9198	14.0723	10.3780	8.5031

	Accuracy				Adaptability			
τ :	5	10	20	40	5	10	20	40
m= 0	18.8156	7.7415	3.6971	1.9020	28.6003	16.9907	11.7570	7.0057
m= 1	15.0837	8.0022	3.8054	2.7083	23.0277	15.3970	9.9306	7.3431
m=10	12.3109	5.7396	4.2164	1.1373	18.0849	13.0661	10.7981	6.6793
m=20	11.5600	6.1781	2.9295	1.1769	17.9858	12.6995	8.8713	5.6556
m=40	10.8352	5.2289	2.6689	1.7460	16.5785	10.9338	8.2416	5.9121

different aspects of evolutionary search process. For the type of changes considered here, the best choice is a cooperation of both strategies.

Section 8 described experiments with non-stationary set of constraints (for cyclic and non-cyclic changes). Experiments showed that memory structures can improve the results but for small values of τ only. As in experiments with non-stationary evaluation function the results of experiments with random immigrants mechanism were better than the results without it. This confirmed our previous observations that the cooperation of both strategies represents the best approach. The experiments showed also that although the results of the algorithm with random immigrants extension are better, the risk of loosing feasible part of the search space is growing too, especially when the feasible part has more complex shape. Hence tuning of replacement rate of random immigrants mechanism is often required.

One of the points of our research was also to demonstrate that environments created by the test-case generator are sufficiently diverse and challenging, and might be appropriate for further research on non-stationary optimization. With this test-case generator we can experimentally compare properties of the evolutionary algorithms and their extensions for many different types of changes, e.g.,

- for different numbers of dimensions of the search space (controlled by parameter n),
- for different numbers of local optima in the environment (controlled by parameter w),
- for different lengths and contours of the path of changes,
- for different changing strategies of the matrix H_a,
- for different lengths of time interval between changes (controlled by parameter τ),
- for cyclical and random types of parameters changes,
- *etc.*

We can also study dependencies between changed parameters, e.g., whether a large number of local optima or a long path of changes can be compensated by a large value of τ, large size of a memory buffer, or a large replacement ratio.

Acknowledgments

The research reported in this paper was supported by the ESPRIT Project 20288 Cooperation Research in Information Technology (CRIT-2): "Evolutionary Real-time Optimization System for Ecological Power Control".

References

[1] Angeline, P., "Tracking Extrema in Dynamic Environments", Proc. of the Sixth Int. Conf. on Evolutionary Programming - EP'97, vol. 1213 in LNCS, Springer, 1997, pp 335-346.

[2] Bäck, T., "On the Behavior of Evolutionary Algorithms in Dynamic Environments", Proc. of the 5nd IEEE Int. Conf. on Evolutionary Computation - ICEC'98, IEEE Publishing, pp 446-451.

[3] Bäck, T., Schutz, M., "Intelligent Mutation Rate Control in Canonical Genetic Algorithm", Proc. of the 9th Int. Symposium – ISMIS'96, vol. 1079 in LNAI, Springer, 1996, pp 158-167.

[4] Cedeno, W., Vemuri, V., R., "On the Use of Niching for Dynamic Landscapes", Proc. of the 4th IEEE Int. Conf. on Evolutionary Computation - ICEC'97, IEEE Publishing, Inc., pp 361-366.

[5] Cobb., H., G., Grefenstette, J., J., "Genetic Algorithms for Tracking Changing Environments", Proc. of the 5th IEEE Int. Conf. on Genetic Algorithms - V ICGA'93, Morgan Kauffman, pp 523-530.

[6] Corne, D., Collingwood, E., Ross, P., "Investigating Multiploidy's Niche", Evolutionary Computing: AISB Workshop, vol. 1143 in LNCS, Springer, 1996, pp 189-197.

[7] Dasgupta, D., McGregor, D. R., "Non-Stationary Function Optimization using the Structured Genetic Algorithm", 2PPSN: Parallel Problem Solving from Nature, Elsevier Science Publishers B. V., 1992, pp 145-154.

[8] De Jong, K., A., "An Analysis of the Behavior of a Class of Genetic Adaptive systems", (Doctoral Dissertation, University of Michigan), Dissertation Abstract Int. 36(10), 5140B. (University Microfilms No 76-9381).

[9] Eiben, A., E., Hinterding, R., Michalewicz, Z., "Parameter Control in Evolutionary Algorithms", Technical Report TR98-07, Department of Computer Science, Leiden University, Netherlands, 1998.

[10] Feng, W., Brune, T., Chan, L., Chowdhury, M., Kuek, C., K., Li, Y., "Benchmarks for Testing Evolutionary Algorithms", The Third Asia-Pacific Conf. on Measurement & Control, Dunhuang, China, 1998.

[11] Ghosh, A., Tsutsui, S., Tanaka, H., "Function Optimization in Non-Stationary Environment using Steady-State Genetic Algorithms with Aging of Individuals", Proc. of the 5th IEEE Int. Conf. on Evolutionary Computation - ICEC'98, IEEE Publishing, Inc., pp 666-671.

[12] Glover, F., Kochenberger, G., "Critical Event Tabu Search for Multidimensional Knapsack Problems", Proc. of the Int. Conf. on Metaheuristic for Optimization, Kluwer, 1995, pp 113-133.

[13] Glover, F., *Tabu Search — Part I*, ORSA Journal on Computing, Vol.1, No.3, pp.190–206, 1989.

[14] Glover, F., *Tabu Search — Part II*, ORSA Journal on Computing, Vol.2, No.1, pp.4–32, 1990.

[15] Goldberg, D., E., Smith, R., E., "Non-Stationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy", Proc. of the 2nd IEEE Int. Conf. on Genetic Algorithms - II ICGA'87, Lawrence Erlbaum Associates, pp 59-68.

[16] Goldberg, D., E., Richardson, J., "Genetic Algorithms with Sharing for Multimodal Function Optimization", Proc. of the 2nd IEEE Int. Conf. on Genetic Algorithms - II ICGA'87, Lawrence Erlbaum Associates, pp 41-49.

[17] Grefenstette, J., J., "Genetic algorithms for changing environments", Parallel Problem Solving from Nature, Elsevier Science Publishers B. V., 1992, pp 137-144.

[18] Hadad, B., S., Eick, C., F., "Supporting Polyploidy in Genetic Algorithms Using Dominance Vectors", Proc. of the Sixth Int. Conf. on Evolutionary Programming - EP'97, vol. 1213 in LNCS, Springer, 1997, pp 223-234.

[19] Homaifar, A., Lai, S., H., Y., Qi, X., "Constrained Optimization via Genetic Algorithms", *Simulation* 62(4), 1994, pp 242-254.

[20] Joines, J., Houck, C., "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems", Proc. of the 1st IEEE Int. Conf. on Evolutionary Computation - ICEC'94, IEEE Publishing, Inc., pp 579-584.

[21] Kwasnicka H., "Redundancy of Genotypes as the Way for Some Advanced Operators in Evolutionary Algorithms - Simulation Study", *VIVEK A Quarterly in Artificial Intelligence*, Vol. 10, No. 3, July 1997, National Centre for Software Technology, Mumbai, pp 2-11.

[22] Le Riche, R., G., Knopf-Lenoir, C., Haftka, R., T., "A Segregated Genetic Algorithm for Constrained Structural Optimization", Proc. of the 6th IEEE Int. Conf. on Genetic Algorithms - VI ICGA'95, Morgan Kauffman, pp 558-565.

[23] Lewis, J., Hart, E., Ritche, G., "A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems", 5PPSN: Parallel Problem Solving from Nature, vol. 1498 in LNCS, Springer, 1998, pp 139-148.

[24] Louis, S., J., Johnson, J., "Solving Similar Problems using Genetic Algorithms and Case–Based Memory", Proc. of the 7th IEEE Int. Conf. on Genetic Algorithms - VII ICGA'97, Morgan Kauffman, pp 283-290.

[25] Louis, S., J., Li, G., "Combining Robot Control Strategies Using Genetic Algorithms with Memory", Proceedings of the Sixth International Conference on Evolutionary Programming - EP'97, vol. 1213 in LNCS, Springer, 1997, pp 431-441.

[26] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 3-rd edition, Springer-Verlag, New York, 1996.

[27] Mori, N., Imanishi, S., Kita, H., Nishikawa, Y., "Adaptation to a Changing Environments by Means of the Memory Based Thermodynamical Genetic Algorithm", Proc. of the 7th IEEE Int. Conf. on Genetic Algorithms - VII ICGA'97, Morgan Kauffman, pp 299-306.

[28] Mori, N., Kita, H., Nishikawa, Y., "Adaptation to Changing Environments by Means of the Feedback Thermodynamical Genetic Algorithm", 5PPSN: Parallel Problem Solving from Nature, vol. 1498 in LNCS, Springer, 1998, pp 149-157.

[29] Ng, K., P., Wong, K., C., "A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization", Proc. of the 6th IEEE Int. Conf. on Genetic Algorithms - VI ICGA'95, Morgan Kauffman, pp 159-166.

[30] Obuchowicz, A., "Adaptation in a Time-Varying Landscape Using an Evolutionary Search with Soft Selection", Proceedings of the 3rd National Conference: Evolution Algorithms and Global Optimization - KAEiOG'99, Warsaw University of Technology Press, 1999, pp 245-251.

[31] Peyral,M., Ducoulombier, A., Ravise,C., Schoenauer, M., Sebag,M., "Mimetic Evolution", Proceedings of the 3rd European Conference AE'97, vol. 1363 in LNCS, Springer, 1998, pp 81-94.

[32] Puppala, N., Sen, S., Gordin, M., "Shared memory based Cooperative Coevolution", Proc. of the 5th IEEE Int. Conf. on Evolutionary Computation - ICEC'98, IEEE Publishing, Inc., pp 570-574.

[33] Ramsey, C., L., Grefenstette, J., J., "Case-Based Initialization of Genetic Algorithms", Proceedings of the 5th IEEE International Conference on Genetic Algorithms - V ICGA'93, Morgan Kauffman, pp 84-91.

[34] Ryan, C., J.J., Collins, "Polygenic Inheritance – A Haploid Scheme That Can Outperform Diploidy", 5PPSN: Parallel Problem Solving from Nature, vol. 1498 in LNCS, Springer, 1998, pp 178-187.

[35] Reynolds R., G., Chung C., J., "Knowledge-based Self-adaptation in Evolutionary Programming using Cultural Algorithms", Proc. of the 4th IEEE Int. Conf. on Evolutionary Computation - ICEC'97, IEEE Publishing, Inc., pp 71-76.

[36] Sebag, M., Schoenauer, M., Peyral, M., "Revisiting the Memory of Evolution", *Fundamenta Informaticae*, Vol.35, 1998, pp. 125-162.

[37] Sebag, M., Schoenauer, M., Ravise, C., "Toward Civilized Evolution: Developing Inhibitions", Proc. of the 7th IEEE Int. Conf. on Genetic Algorithms - VII ICGA'97, Morgan Kauffman, pp 291-298.

[38] Sebag, M., Schoenauer, M., Ravise, C., "Inductive Learning of Mutation Step-Size in Evolutionary Parameter Optimization", Proc. of the Sixth Int. Conf. on Evolutionary Programming - EP'97, vol. 1213 in LNCS, Springer, 1997, pp 247-261.

[39] Vavak F., Fogarty T.C., Jukes K., "Learning the Local Search Range for Genetic Optimization in Non-Stationary Environments" , Proc. of the 4th IEEE Int. Conf. on Evolutionary Computation - ICEC'97, IEEE Publishing, Inc., pp 355-360.

[40] Vavak, F., Fogarty, T., C., "Comparison of Steady State and Generational Genetic Algorithm for Use in Non-Stationary Environments" Proc. of the 3rd IEEE Int. Conf. on Evolutionary Computation - ICEC'96, Nagoya University, IEEE Publishing, Inc., pp 192-195.

[41] Voigt, H., M., Lange, J., M., "Local Evolutionary Search Enhancement by Random Memorizing", Proceedings of the 5th IEEE International Conference on Evolutionary Computation - ICEC'98, IEEE Publishing, Inc., pp 547-552.

[42] White, T., Oppacher, F., "Adaptive Crossover Using Automata", 3PPSN: Parallel Problem Solving from Nature, vol. 866 in LNCS, Springer, 1994, pp 229-238.

[43] Yoshida, Y., Adachi, N., "A Diploid Genetic Algorithm for Preserving Population Diversity – pseudo–Meiosis GA", 3PPSN: Parallel Problem Solving from Nature, vol. 866 in LNCS, Springer, 1994, pp 482-491.