Intelligent Information Systems VIII
Proceedings of the Workshop held in
Ustroń, Poland, June 14-18 , 1999

Evolutionary Algorithms for Non–Stationary Environments

Krzysztof Trojanowski,[1] Zbigniew Michalewicz,[2,1]
trojanow@ipipan.waw.pl, zbyszek@uncc.edu

[1] Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland
[2] Department of Computer Science, University of North Carolina,
Charlotte, NC 28223, USA

Abstract. Most real-world applications operate in dynamic environments. In such environments often it is 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 non-stationary problems (i.e., problems, which change in time) are discussed. 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. A brief review of existing applied measures of obtained results is also presented.

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 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. In section 2 a model of a problem is proposed. The model includes stationary and non-stationary problems as well. Section 3 provides a discussion of types of changes, whereas Section 4 presents a brief review of existing approaches to non-stationary problem optimization. In Section 5 existing applied measures of obtained results are discussed. Some conclusions are drawn in section 6.

2 The Model

Most real-world optimization problems can be modelled 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, $\overline{x} = (x_1, \ldots, x_n)$. There is also an additional discrete variable t, which plays the role of time.[1]

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:

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

[1] Note, that time is continuous, however, any changes are usually recorded in discrete intervals.

where $x_i \in \langle q_i(t), r_i(t)\rangle$ for $1 \leq i \leq n$ ($\langle q_i(t), r_i(t)\rangle$ 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:

$$F = f(\overline{x}, t) \ : \ \mathcal{D} \rightarrow \boldsymbol{R}.$$

- $\mathcal{C}$ – set of constraints:

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

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 in time — some variables may disappear or a new variables can be introduced.

Changes of the number of dimensions as well as changes of intervals for domain variables modify the nature of the problem. In that situation it is usually necessary to re-start the search procedure and to tune the optimization tool to the new problem after a change has occurred. So in further text we do not discuss this form of changes and we assume that the domains is 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., $\mathcal{F} = f(\overline{x}, t)$ or in constraint inequalities (i.e., $c_i = c_i(\overline{x}, t)$ 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.

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 were investigated heavily in the EA community.

In 1975 De Jong 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

Table 1. Classification of changes in a process. The symbol ∅ 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*	∅
2	*static*	*static*
3	*static*	*var*
4	*var*	∅
5	*var*	*static*
6	*var*	*var*

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 [24]. 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. Thus in the rest of the paper we concentrate on these.

3 Discussion of types of changes

The model of a problem discussed in the previous section introduced a time variable t. At any time there might be a change in the evaluation function or in the set of constraints. However, it is important to discuss a nature of such changes.

The first issue to consider is the visibility of a change. It is necessary to investigate whether the change which occurred in the environment is known to the system or if it must be detected first. When the changes have to be detected first, the information about them can be given to the system from outside or it can deduced on the basis of behavior of system components. In the first case an additional module controlling real-world environment and transmitting update information to the optimization system is necessary. In the second case the system has to manage by itself.

The most intuitive form of changes detection in evolutionary algorithms is the observation of population performance. E.g., the time averaged performance of the whole population was controlled in [32]: a significant decrease of the performance was a signal that a change occurred, so it is time to perform some additional steps to recapture the near-optimum solution.

In further discussion we will assume that all necessary informations about changes are given to the system and are known just after the change appeared. In the following two subsections we consider cases where a change occurs in (1) the objective function or (2) constraints of the problem.

3.1 Changes of a function landscape

Changes of a function landscape represent real-world optimization tasks where values of proposed solutions change in time and thus demand continuous optimization process. An example of a situation of this type is a (real-time) traveling salesperson problem. The level of traffic in the streets change in time respectively to the hours of a day. The salesperson plans the route taking into account the time of the journey (as a longer route can take less time than the shorter one performed on the crowded streets), and the total time depends on the time of the day. In other words, the same solution (in terms of a route) may have different merit at different times.

Changes in the function landscape can be classified in many ways. One of them is based on the regularity of changes:

1. **random changes** — where the next change of the landscape does not depend of the previous change and the time t. If changes are too large we have a situation of optimization of a completely different new problem.
2. **non-random and non-predictable changes** — where the next change do depend of the previous changes, but the dependency is so complex that we consider these changes not predictable.
3. **non-random and predictable changes** — where the change of the landscape is deterministic.

   If the changes are defined by a function which is known we can try to predict the future coordinates of the optimum and improve the search process by some deterministic procedures.

   This class of non-random and predictable changes can be divided into two additional subclasses: (1) cyclical, and (2) non-cyclical changes.

The next criterion of classification is the continuous or discrete nature of changes. We can distinguish two areas of this classification: the time of search and the search space. Changes which are continuous in time make the environment at least a little different every time we measure it. Discrete changes appear in the environment from time to time and there are periods of stagnations between them.

Changes which have a continuous nature in the search space move the optimum to the other point which is close enough to be found again by the local search mechanisms without a risk of becoming trapped in a local optimum. Discrete changes in the space represent situation when optimum skips from one point to another during a single change. The distance between these two coordinates before and after skip is big enough to make the local search inefficient or too expensive.

3.2 Changes of problem constraints

Changes of a set of constraints simulate other real-world situations where some feasible solutions become unacceptable or — in the opposite case — unacceptable solutions become feasible. An example is a factory with a set of machines producing some goods. List of tasks and list of machines are the input data for a system optimizing task management in the factory. A case when some of machines break and need time to be repaired is the situation where a new constraint appears in the optimizing system. Task allocations for the broken machine are not feasible till the machine is repaired.

Changes of problem constraints can be constant or discrete in time and in the search space as well. Additionally they can be classified by the change of capacity of the feasible part of search space.

For a discrete search space $\mathcal{S}$ its capacity $V(\mathcal{S})$ is equal to the number of possible solutions in the domain; similarly, the capacity of the feasible part $V(\mathcal{F})$ is equal to the number of feasible solutions in the domain.

For continuous spaces we can measure the capacity of sets using integrals. An example is a n-dimensional search space $\mathcal{S}$ limited by pairs of boundaries (lower and upper boundary) for each of dimensions. Capacity of $\mathcal{S}$ from the example is measured by the following formula:

$$V(\mathcal{S}) = \int_{q_1}^{r_1} \dots \int_{q_n}^{r_n} dx_1 \dots dx_n$$

where q_i and r_i represent lower and upper boundary of an i-th dimension. Capacity of the feasible search space equals:

$$V(\mathcal{F}) = \int \dots \int_{\mathcal{F}} dx_1 \dots dx_n$$

where: $\mathcal{F}$ is the feasible part of the search space.

As the feasible part $\mathcal{F}$ of the search space is defined by a set of constraints $\mathcal{C}$ (which are time dependent), the capacity of the feasible part of the search space can change over time.

For better control of constraints behavior during the search process two measures of changes can be very useful: (1) a feasible part of the search space capacity ratio, and (2) a gradient of change. Let $\rho(t)$ is a feasible search space ratio for a given time t; for dynamic constraints this ratio is a function of t:

$$\rho(t) = \frac{V(\mathcal{F}(t))}{V(\mathcal{S})}.$$

For dynamic constraints we can also measure a gradient of change $\mathcal{G}(t)$:

$$\mathcal{G}(t) = \frac{V((\mathcal{F}(t) \cup \mathcal{F}(t-1)) - (\mathcal{F}(t) \cap \mathcal{F}(t-1)))}{V(\mathcal{F}(t) \cup \mathcal{F}(t-1))}$$

It's value is in the range $\langle 0, 1 \rangle$ and equal 1 when the feasible search space changed completely and 0 when the feasible space for the time t is the same as the feasible space for the time $t-1$.

4 Existing 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 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 [15, 4],
  - techniques based on the concepts of temperature and entropy [25, 26],
  - techniques based on the concept of the age of individuals [10],
  - a random immigrants mechanism [5, 16],
  - a mechanism of variable range local search around the current locations [32].
- Adaptation and self-adaptation mechanism. Dynamical adjustment of the algorithm to the non-stationary environment is the next feature of the efficient optimization. So adaptive and self-adaptive techniques are the next significant extension of evolutionary algorithm [1, 3, 8]. In adaptation the parameters of the algorithm are updated using statistic or heuristic rules to determine how to update. Update of the parameters in the genetic process in parallel with searching of the optimum is called a self-adaptation. Both these techniques of parameters update were applied to non-stationary optimization tasks.
- Redundancy of genetic material. One of the most important abilities in adaptation to changes is reasoning from previous experiences. If we want to reason from past a place to collect experiences is needed. So the next good idea of enforcement of efficiency in dynamic optimization is adding memory structures to the algorithm.
  One of the earliest forms of memory although not used for non-stationary optimization was the *tabu-search* strategy [12, 13]. Beside TS a considerable number of other ideas using past experience and the forms of memory were proposed. We can classify them into several types [30]:
  - numerical memory — where the modification of algorithm parameters is performed using experience of previous generations [29, 30, 34]. 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) [31]. 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, 14, 17, 20, 23, 25, 28, 35]. 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.

5 Measures for comparisons of algorithms

When the 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 [7].

- **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, 16, 32, 33].

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, 6, 10, 14, 16, 22, 25, 26, 27, 33]. In some papers graphs of average values of all individuals or of the worst individual in the population were also analyzed [5, 14, 6, 25, 26]. 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 [25]. 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 [9] 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 $f(x_0)$ to the value of optimum f_{opt}. For maximization and minimization problems we have following formulas respectively:

$$Op_{max}(x_0) = \frac{f(x_0) - f_{min}}{f_{max} - f_{min}} \qquad Op_{min}(x_0) = \frac{f_{max} - f(x_0)}{f_{max} - f_{min}}$$

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

$$Ac(x_0) = 1 - \frac{|x_{opt} - x_0|}{x_{max} - x_{min}}$$

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 [26, 33] 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 [26] 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 [7]: *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 — Accuracy is a measure dedicated exclusively for dynamic environments. It is a difference between the value of the current best individual in the population of the "just before the change" generation and the optimum value averaged over the entire run.

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

• Adaptability — 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:

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

Clearly, the smaller measured values are (for both Accuracy and Adaptability) the better result. 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.

6 Conclusions

In this paper a model for non-stationary problems was proposed. Two components of the problem can be changed in time: evaluation function and problem constraints. We discussed and classified different types of components changes. A brief review of existing evolutionary approaches to non-stationary problem optimization was presented. They were divided into three main groups representing (1) population diversity maintenance techniques, (2) adaptation and self–adaptation techniques, and (3) redundant genetic material approaches. In the last section different measures for comparisons of obtained results were discussed. They mostly exploited the iterational nature of the search process and observed values of individuals changing in time. Some measures controlled diversity of the population of solutions which is continuously modified and improved by the algorithm. Two new measures of non-stationary optimization results were also proposed.

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. Dasgupta, D., McGregor, D. R., "Nonstationary Function Optimization using the Structured Genetic Algorithm", 2PPSN: Parallel Problem Solving from Nature, Elsevier Science Publishers B. V., 1992, pp 145-154.
7. 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).
8. Eiben, A., E., Hinterding, R., Michalewicz, Z., "Parameter Control in Evolutionary Algorithms", Technical Report TR98-07, Department of Computer Science, Leiden University, Netherlands, 1998.
9. 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.
10. Ghosh, A., Tsutsui, S., Tanaka, H., "Function Optimization in Nonstationary 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.
11. 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.
12. Glover, F., *Tabu Search — Part I*, ORSA Journal on Computing, Vol.1, No.3, pp.190–206, 1989.
13. Glover, F., *Tabu Search — Part II*, ORSA Journal on Computing, Vol.2, No.1, pp.4–32, 1990.
14. Goldberg, D., E., Smith, R., E., "Nonstationary 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.
15. 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.
16. Grefenstette, J., J., "Genetic algorithms for changing environments", Parallel Problem Solving from Nature, Elsevier Science Publishers B. V., 1992, pp 137-144.
17. 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.
18. Homaifar, A., Lai, S., H., Y., Qi, X., "Constrained Optimization via Genetic Algorithms", *Simulation* 62(4), 1994, pp 242-254.
19. 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.
20. 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.

21. 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.
22. 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.
23. 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.
24. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 3-rd edition, Springer-Verlag, New York, 1996.
25. 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.
26. 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.
27. 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.
28. 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.
29. 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.
30. 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.
31. 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.
32. Vavak F., Fogarty T.C., Jukes K., "Learning the Local Search Range for Genetic Optimization in Nonstationary Environments" , Proc. of the 4th IEEE Int. Conf. on Evolutionary Computation - ICEC'97, IEEE Publishing, Inc., pp 355-360.
33. Vavak, F., Fogarty, T., C., "Comparison of Steady State and Generational Genetic Algorithm for Use in Nonstationary Environments" Proc. of the 3rd IEEE Int. Conf. on Evolutionary Computation - ICEC'96, Nagoya University, IEEE Publishing, Inc., pp 192-195.
34. White, T., Oppacher, F., "Adaptive Crossover Using Automata", 3PPSN: Parallel Problem Solving from Nature, vol. 866 in LNCS, Springer, 1994, pp 229-238.
35. 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.