The Memory Indexing Evolutionary Algorithm for Dynamic Environments

Aydın Karaman, Şima Uyar, and Gülşen Eryiğit

Istanbul Technical University,
Computer Engineering Department, Maslak Istanbul TR34469 Turkey
karamanayd@itu.edu.tr
{etaner, gulsen}@cs.itu.edu.tr

Abstract. There is a growing interest in applying evolutionary algorithms to dynamic environments. Different types of changes in the environment benefit from different types of mechanisms to handle the change. In this study, the mechanisms used in literature are categorized into four groups. A new EA approach (MIA) which benefits from the EDA-like approach it employs for re-initializing populations after a change as well as using different change handling mechanisms together is proposed. Experiments are conducted using the 0/1 single knapsack problem to compare MIA with other algorithms and to explore its performance. Promising results are obtained which promote further study. Current research is being done to extend MIA to other problem domains.

1 Introduction

Evolutionary algorithms (EA) are heuristic search algorithms applied in both stationary and non-stationary (dynamic) problems. For the rest of the paper, EAs designed for dynamic problems will be called dynamic evolutionary algorithms (DynEA). Very successful implementations of EAs in stationary environments exist in literature but there are additional challenges in dynamic environments, such as having to adapt to the new environmental conditions and tracking the optima. In order to accomplish these, new algorithms have been introduced. Detailed surveys and discussions of EAs and dynamic environments can be found in [4, 13, 18]. DynEAs are categorized under three headings in [4] as *Type-1*: DynEAs reacting on change, *Type-2*: DynEAs maintaining diversity throughout the run and *Type-3*: DynEAs based on memory. Each type of DynEA employs a single approach or a set of approaches (models), while adapting to the new environments. These models can be classified as *operator-based*, *memory-based*, *population-based* and *initialization-based*. A similar but more detailed classification can be found in [18]. *Operator-based models* use adaptation or modification of some EA operators, especially those that are responsible for diversity such as mutation and selection. The main aim is to diversify the current genetic material by using genetic operators, when change occurs. Hyper-mutation [5], Variable Local Search [17], Random Immigrants [7] and Thermo-Dynamical Genetic Algorithms [11] are examples of such DynEAs. The first three of these approaches

F. Rothlauf et al. (Eds.): EvoWorkshops 2005, LNCS 3449, pp. 563–573, 2005.

work with the mutation operator, while the last one uses a different selection scheme based on a free energy function. *Memory-based models* use extra memory in order to preserve extra genetic material that may be useful in later stages of the run. Extra memory usage can be implemented either implicitly or explicitly. Algorithms that use redundant representations are among the well known implicit memory implementations. The most common of these representations are ones that use diploid chromosomes with a dominance scheme [10, 14, 16]. Explicit memory is implemented by allocating extra space for preserving currently available genetic material to be used in later runs. A good example of this approach is discussed in [2]. It can be seen that, memory based techniques perform best when the environment oscillates between several states. *Population-based models* use more than one population, each of which may be assigned different responsibilities. The aim is to use the available number of individuals in a more effective way. For example, the memory-enhanced algorithm introduced in [2] uses two populations. One of these populations is responsible for remembering good old solutions and the other is responsible for preserving diversity. A more recently proposed approach [3] uses multiple sub-populations distributed throughout the search space to watch over previously found optima (local or global), thus increasing the diversity in the overall population. *Initialization-based models* use problem specific knowledge in order to initialize the first population of the current environment so that individuals are in the locality of the new optima. The Case-Based Initialization of Genetic Algorithms introduced in [15] uses a similar approach. In that study, a model of the environment is maintained and is updated after environment changes. The good solutions found for previous environments are inserted into the GA population when similar environments are encountered. All of these models have their strengths and weaknesses. Studies show that different types of dynamic problems need different types of models. It can be seen that it is better to have a combined model for better performance under different types of change. In [18], the author develops a formal framework for classifying different types of dynamic environments and tries to perform a mapping between problem classes, DynEAs and performance criteria.

For the purposes of this paper, a DynEA will be defined as an evolutionary algorithm designed for working in dynamic environments, consisting of a set of the models mentioned above. This definition implies that, it might be advantageous to partition an evolutionary algorithm into sub-parts, each of which is designed for different purposes. Most algorithms found in literature have been developed to only include one or two of the models. For instance, the hyper-mutation approach [5] uses only the operator-based model while the memory-enhanced algorithm [2] uses the population-based model together with the memory-based model. When thinking of designing an efficient evolutionary algorithm in terms of these models, it can be seen that the evolutionary algorithm should include an operator-based model in order to supply the needed diversity, a memory-based model to benefit from the previously discovered genetic material, a population-based model to use the limited number of individuals efficiently

and an initialization-based model that uses problem specific knowledge to guide the individuals towards the new optima.

In this paper, a new DynEA approach called Memory Indexing Algorithm (MIA) is proposed. MIA has been designed with the above considerations in mind. This study is a preliminary look into the design and performance of such an algorithm and the results obtained give a preliminary overview as to the applicability and the performance of the approach. Currently, for ease of analysis, a very simple test problem, namely the 0/1 single knapsack problem, has been selected as the benchmark. Experiments are performed to understand the operations of the basic mechanisms of MIA as well as to compare its performance with similar state of the art DynEA approaches in literature. Research continues to extend MIA to other problem domains. This paper is organized as follows: Section 2 introduces MIA and explores its mechanisms and outline. Section 3 gives details of the experimental setup, presents and discusses the test results. Section 4 concludes the paper and proposes possibilities for future work.

2 The Memory Indexing Algorithm

The Memory Indexing Algorithm (MIA) uses problem specific information for working with the appropriate incorporated mechanisms. This information is based on a measure that identifies environments and is used by MIA to index encountered environments. For the 0/1 single knapsack problem, the indexing mechanism used to differentiate environments is termed as the environment quality (EQ) and is explored in greater detail in [8]. As can be seen in the next subsections, MIA uses concepts originating from explicit memory based DynEAs, the hyper-mutation mechanism and also estimation of distribution approaches (EDAs) [9]. MIA can be put in *Type-1* of algorithms because it acts on change and initializes the next generation according to a distribution array (DA) if a similar environment has been encountered before. If there is no previous information regarding the new environment, the algorithm uses a standard technique. In this study, as an example, the hyper-mutation method is applied. MIA can also be said to be of *Type-3*, since it uses a DA that can be seen as a form of an external memory. Due to this, MIA is compared with the hyper-mutation approach of *Type-1*, and Memory Enhanced Algorithm (MEA) of *Type-3*. Experiments and results are given in section 3.

The distribution array (DA) is a list of ratios representing the frequency of each allele at each gene position in the population. Each value in the DA is used by MIA as the probability of initializing the corresponding gene location to the corresponding allele when a similar environment to the one in which the DA was calculated is re-encountered. This idea is similar to the population initialization technique used in the PBIL [1] approach. There is a DA for each different environment group which has been encountered during the run of MIA. To the authors' best knowledge, such a memory and population re-initialization scheme has not been applied to dynamic environments before. The DA values

are calculated as given in Eq. 1, using the individuals in the generation when the change occurs (assuming no changes within a generation).

$$DA_i(j) = \frac{N_{ij}}{PopSize},\ i = 1, 2, ..., L_C \ and\ j = 1, 2, ..., L_A \tag{1}$$

where $DA_i(j)$ is the ratio of allele-j at ith gene location, and N_{ij} is the number of genes having allele-j at ith gene location in the population, L_C is the chromosome length and L_A is the number of different alleles. Grouping of environments requires problem specific criteria to differentiate between different types of environments. For example in [15] the example task is a two-agent game of cat-and-mouse. They use four aspects of the environment (namely distribution of speed values and turn values of the target agent, the radius within which the target may detect the tracker and the size of the target) to categorize the environments. Using fitness distributions may also be a simple but effective measure for this purpose. However, since this paper mainly focuses on the benefits of applying a DA-based re-initialization for similar environments, for simplicity, when applying MIA to the single constraint 0/1 knapsack problem, environments are grouped according to the EQ. Better and more practical groupings are the focus of another study. The grouping used in this study is also controlled by an additional parameter called the *tuner parameter* (TP). TP can be thought of as the number of cuts in the environment space and determines the number of environment groups. In this study, for the chosen example problem, the proportion of feasible points in the search space to the whole search space is used to differentiate between different environments and is assigned as the EQ for each environment. Theoretically, this EQ value and the group id which is based on this, is calculated as in Eq. 2.

$$EQ_i = \frac{m_i}{T} \quad \Rightarrow \quad G_i = \lfloor TP * EQ_i \rfloor \tag{2}$$

where EQ_i is the environment quality of the ith environment, m_i is the number of feasible individuals in the ith environment, T is the total number of all possible individuals in the search space, G_i is the group id of the ith environment and TP is the tuner parameter. Normally the search space of the problems encountered are large and thus a complete enumeration of all points is not possible. An approximation measure is needed. So instead of using all the points in the search space to calculate the EQ value, k points are sampled randomly and the estimated EQ value denoted as EQ' is determined using the chosen samples. As part of MIA, a standard evolutionary algorithm runs until a change is detected. When change occurs, the DA of the population is calculated and is recorded for the corresponding group of the previous environment. The EQ'value for the new environment and its group id is determined. If the DA for the new environment was previously calculated, a percentage of the population is re-initialized using the corresponding DA, otherwise a standard hyper-mutation method is applied. For this current implementation of MIA, for the chosen test problem, a standard EA is run during the stationary periods and a standard hyper-mutation technique is applied if there is no current information to apply the indexing and

re-initialization mechanism of MIA. However, other EA approaches and other change handling techniques can be experimented with.

3 Experiments and Results

The results obtained in this section give a preliminary overview as to the applicability and the performance of the MIA approach. It is shown in [12] that good change detection techniques are effective in the performance of EAs in dynamic environments. In this study, to rule out any effects that may be the result of faulty change detection mechanisms, changes are explicitly made known to all tested algorithms. Since MIA uses the hyper-mutation method (HMT) when the new environment group has not been encountered before, it is compared with HMT [5] and since MIA also employs an external memory, it is also compared with the Memory Enhanced Algorithm (MEA) [2]. For all tested algorithms: there are 50 individuals in each population; the chromosome length is 100; uniform crossover rate is 0.8; parent selection is via binary tournaments; generational replacement is used with elitism where the best individual of the previous population replaces the worst individual of the new population (except for right after a change); binary mutation rate is $0.6/ChromosomeLength$; there are 50 environments for each run and results are averaged over 50 runs. For HMT, hyper-mutation rate is 0.1 and this higher mutation rate is applied for 2 EA generations after a change occurs. For MEA, memory size is 10 individuals and the storing period is 10 EA generations. For MIA, TP is chosen as 10 and for practical purposes, the estimated EQ'values are used to classify environments rather than using the actual EQ values; k (number of sampled points) is chosen as 100 individuals. These randomly sampled 100 points is called the test population. The individuals of the test population are determined randomly at the start of the EA run. The same test population is applied once to each environment that is encountered right after a change to determine the corresponding EQ'value and its group id. Tests were performed in [8] to show that a simple random sampling technique is sufficient to provide the heuristic information needed for MIA. Offline performance [4] measure is used to evaluate the performance of selected algorithms. All results are reported as averaged over 50 runs against the same problem instance. T-tests with a significance level of 0.05 were performed on the offline performances. The results of the tests (omitted here for lack of space) showed that all observed differences, except for the comparison of MIA1 and MIA2 in Test 1 of Experiment 2 and the comparison between MIA2 and MIA3 in Test 2 of Experiment 3, are statistically significant. For each encountered environment, MIA applies the environment to the test population once to determine the group id. For performance comparisons, this means 5000 more fitness evaluations corresponding to 50 generations. Each run consists of 50 randomly generated environments and the change period is determined separately for each test, so the maximum number of generations for each test is $(50 * changePeriod)$. The same single constraint 0/1 knapsack instance which is randomly generated prior to the experimental runs, is used in all experiments.

There are 100 items with profits between 1000 and 5000, and weights between 1 and 100. For simplicity of the analysis, changes are allowed to occur only in the capacity constraint of the knapsack. All capacity constraints are produced randomly if not otherwise stated. A penalty-based method is used for infeasible individuals and the penalty function proposed for multiple knapsack problems in [6] is adapted for a single knapsack problem as given in Eq. 3.

$$penalty = \frac{p_{max} + 1}{w_{min}} * CV \quad , \quad w_{min} > 0 \tag{3}$$

where p_{max} is the maximum profit, w_{min} is the minimum resource consumption (weight), and CV is the constraint violation of the individual. For testing different properties of MIA, in some of the experiments in the next sections, a *training stage* and a *test stage* is defined. In the training stage, controlled environments are introduced to MIA. This stage is defined in such a way that MIA gets a chance to calculate a DA for each of the possible environment groups. The training stage consists of a series of stationary periods. The length of the stationary periods determines the number of generations the EA will run for each environment. The length of these stationary periods in the training stage will be referred to as the *training period* and will be given in EA-generations units. At the end of the training stage, a DA for all possible environment groups is initialized. The number of these environments depends on the selected TP parameter as explained above. In the testing stage, environments are generated and introduced to the EA randomly. A *testing period* is also determined similarly to the one explained above for the training stage.

3.1 Experiment 1

Experiment 1, composed of 3 tests, is for studying the algorithms under different settings of the training period and test period. In the first 2 tests, first 11 environments (TP=10) are selected such that DAs for all groups are initialized once during the training stage. The performances of the algorithms are calculated only for the test stage which begins right after the training stage is completed. The last test in this experiment does not have a training stage. Environments included in the testing stages are generated randomly in all test instances. In the offline performance figures of Test 1 and Test 2, generations are shown to start from zero, however this value corresponds to the beginning of the testing stage. The aim of Tests 1 and 2 in this experiment is to observe the effect of the length of the training period on the performance of MIA. In Test 1, training period and test period are chosen as 100 EA generations and 20 EA generations respectively. The training period is considered to be long, while the test period is not. Under this setting, MIA is expected to perform its best compared to the other settings in the following tests. This is because of the fact that, a long training period means that the EA has more time to converge to a local/global optimum and thus may be able to calculate more accurate distribution arrays. This will allow the EA to start off from a better point in the search space in the similar environments encountered during the test phase. HMT and MEA are expected

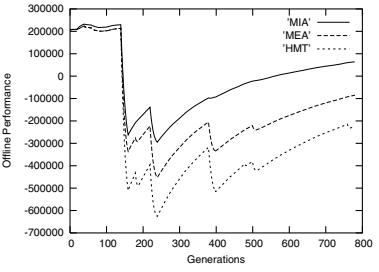

Fig. 1. Offline Performances: Exp.1 Test 1

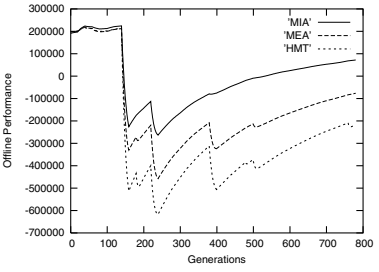

Fig. 2. Offline Performances: Exp.1 Test 2

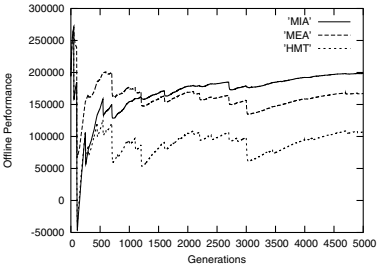

Fig. 3. Offline Performances: Exp.1 Test 3

not to recover as well as MIA, due to the fact that MIA is expected to start from better points in the search space than the others due to the initialization using a DA. In Test 2, both the training period and the test period is chosen as 20 EA generations. The purpose of Test 3 is to evaluate the performance of MIA against the performances of HMT and MEA when there is no training stage. There are a total of 100 environments produced at random where the period of change is every 50 EA generations. Offline performances of algorithms for Test 1, Test 2 and Test 3 are given in Fig. 1, Fig. 2, and Fig. 3 respectively. Fig. 1, Fig. 2, and Fig. 3 show that performance of the MIA is the best whereas the performance of the HMT is the worst. Comparing Fig. 1 and Fig. 2, it can be said that, although the training period is decreased from 100 to 20, this does not much effect the relative performance of MIA. Moreover from Fig. 3, it can be said that although there is no training stage, MIA outperforms the others after some time. The reason for the delay of MIA to be the best performer is that MIA is more likely to run hyper-mutation at the beginning of the evolution. As a result, MIA is expected to perform as well as hyper-mutation in the beginning but better in the later generations after several environments have been encountered and DAs for these environments have been recorded. All figures show that MIA gets better as time passes.

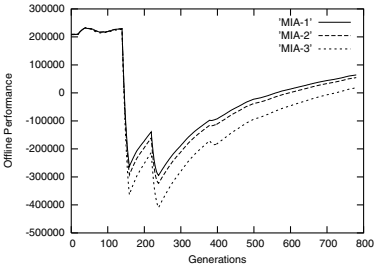

Fig. 4. Offline Performances: Exp.2 Test 1

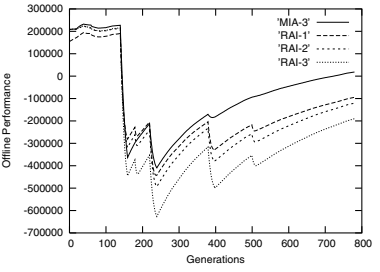

Fig. 5. Offline Performances: Exp.2 Test 2

3.2 Experiment 2

Experiment 2 includes 2 tests with exactly the same parameter settings and the problem instance, including the environments with Test 1 of Experiment 1. The aim of Test 1 is to observe the effect of the initialization ratio on the performance of MIA and the aim of Test 2 is to prove that initialization of the population with a DA is a more powerful technique than a pure random initialization of the population when change occurs. As stated previously, if the new environment is in a group whose DA is initialized previously, the first population in this environment is re-initialized partly using the DA. Initialization ratio is the parameter which determines the ratio of the new individuals to insert into the current population. Parameter settings and the problem instances of the following tests, including the environments, are same as Test 1 of Experiment 1. Test 1 is conducted to test MIA with different initialization ratios. In Fig. 4, MIA1, MIA2, and MIA3 are the MIAs with initialization ratios of 1, 0.5 and 0.2 respectively. The individuals to be replaced by new individuals are determined randomly. In Test 2, performance of MIA is compared with a pure random initialization (RAI). In Fig. 5, RAI1, RAI2, and RAI3 are the random initialization with ratios 1.0, 0.5, and 0.2 respectively. Fig. 4 depicts the MIAs with different initialization ratios. It is seen that performance of MIA3 is the worst, while MIA1 is the best. Moreover, performance of MIA2 is much nearer to MIA1 than MIA3. Therefore, initialization ratio should be greater than 0.5. Fig. 5 shows the performances of RAIs and MIA3. Although MIA3 is the worst of the MIAs compared in Test 1, it outperforms all of the RAIs, which means that MIA's performance is largely due to the use of a DA to initialize the population.

3.3 Experiment 3

This experiment is conducted to observe the effect of the tuner parameter TP on the performance of MIA. In the figures of this experiment, MIA1, MIA2, MIA3 and MIA4 are the MIAs with TP equal to 5, 10, 20 and 60 respectively. Parameter settings and the problem instance of these tests, including the environments are same with the Test 3 in Experiment 1. Results of Test 1 are shown in Fig. 6, which depicts the performances of MIAs with different TPs when there are 100

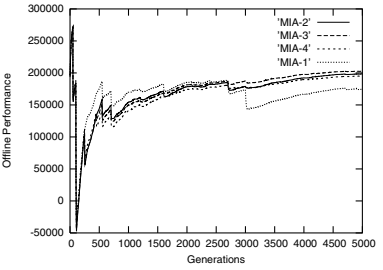

Fig. 6. Offline Performances: Exp.2 Test 1

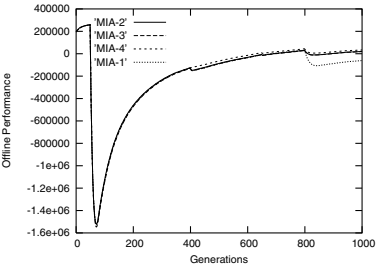

Fig. 7. Offline Performances: Exp.2 Test 2

environment changes. In Test 2, number of environment changes is decreased from 100 to 20. Results are shown in Fig. 7. In both Fig. 6 and Fig. 7, it is apparent that when TP is decreased from 10 to a lower number, performance of MIA decreases drastically. On the other hand, if TP is increased from 10 to a greater value, MIA's performance may decrease or increase. This indefiniteness is due to the fact that, if TP increases, there is a smaller number of environments in each group and thus the groups get more homogeneous. Therefore, DAs of each group will lead to more accurate initializations leading to a better performance. However, since the number of groups is increased by TP, the new environment detected after a change is less likely to fall into a group whose DA was previously initialized. This means that the EA may end up having to use hyper-mutation most of the time. In conclusion, these two contradictory factors determine the performance of MIA when TP increases.

As it is mentioned before, MIA becomes more successful as generations pass due to the initialization-based approach it uses. Each time MIA encounters an environment similar to a previous one, the population is re-initialized using the recorded DA. During the current stationary environment MIA gets a chance to work on the same (or similar) environment for more generations, possibly to find a better solution. At the end of the stationary period, the newly calculated DA is recorded for this environment group. Thus each time MIA works on a representative of a group, it has a chance of finding and recording a better DA for the group.

4 Conclusions and Future Work

Studies in literature introduce many different approaches to dynamic problems. This paper defines dynamic evolutionary algorithms as being a set of approaches working with each other. A DynEA should benefit from operator-based approaches to obtain needed diversity, from memory-based approaches to remember useful genetic material, from population-based approaches to use limited number of individuals effectively, and from initialization-based approaches, if possible, to start from the locality of the optimum of the current environment.

The Memory Indexing Algorithm (MIA) is introduced and compared with hypermutation and Memory Enhanced Algorithms through a set of experiments. All experiments show that MIA outperforms others for the tested cases on the selected test problem and is more robust to environment changes. MIA can be further improved by modification in the calculation of the distribution array. Additionally, in this study a very simple problem was selected as the benchmark, however extending the applicability of MIA to other problem domains is needed. Since, MIA uses problem specific information in order to create an index for each possible environment, it is an interesting study to explore the ways of defining and differentiating environments of other dynamic problems. This study is currently being conducted. Furthermore, MIA only uses three of the identified approaches for dealing with change. In its current implementation, it does not use population-based techniques explicitly, though each indexed environment and its DA may be interpreted similar to the sub-populations in the SOS [3] approach. This aspect of the approach will be explored as a future study.

References

1. Baluja S., Caruana R., "Removing the Genetics from the Standard Genetic Algorithm", in *Proc. of the Intl. Conf. on Machine Learning*, Morgan Kaufmann, pp. 38-46, 1995.
2. Branke J., "Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems", in *Proc. of Cong. On Evolutionary Computation*, pp 1875-1882, IEEE, 1999.
3. Branke J., Kaussler T., Schmidt C., and Schmeck H., "A Multi-Population Approach to Dynamic Optimization Problems", Adaptive Computing in Design and Manufacturing, pp. 299-308, Springer, 2000.
4. Branke J., *Evolutionary Optimization in Dynamic Environments*, Kluwer, 2002.
5. Cobb H. G., "An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms having Continuous, Time-Dependent Non-stationary Environments", NCARAI Report: AIC-90-001, 1990.
6. Gottlieb J., "Evolutionary Algorithms for Constrained Optimization Problems", PhD Thesis, Tech. Univ. of Clausthal, Germany, 1999.
7. Grefenstette J. J., "Genetic Algorithms for changing environments", in *Proc. of Parallel Problem Solving from Nature*, pp. 137-144, Elsevier, 1992.
8. Karaman, Uyar S., "A Novel Change Severity Detection Mechanism for the Dynamic 0/1 Knapsack Problem", in *Proceedings of Mendel 2004: 10th Intl. Conf. on Soft Computing*, 2004.
9. Larranaga P., Lozano J. A., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer, 2001.
10. Lewis J., Hart E., Graeme R., "A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems", in *Proc. of Parallel Problem Solving from Nature*, Springer, 1998.
11. Mori N., Kita H., and Nishikawa Y., "Adaptation to a Changing Environment by Means of the Feedback Thermodynamical Genetic Algorithm", in *Proc. of Parallel Problem Solving from Nature*, pp. 149-158, 1998.

12. Morrison R. W., De Jong K. A., "Triggered Hypermutation Revisited", in *Proceedings of Cong. On Evolutionary Computation*, pp 1025-1032, IEEE, 2000.
13. Morrison R. W., *Designing Evolutionary Algorithms for Dynamic Environments*, Springer, 2004.
14. Ng K. P., Wong K. C., "A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization", in *Proc. of the Sixth Intl. Conf. on Genetic Algorithms*, 1995.
15. Ramsey C. L., Grefenstette J. J., "Case-Based Initialization of Genetic Algorithms", in *Genetic Algorithms: Proceedings of Fifth International Conference*, pp. 84-91, San Mateo: Morgan Kaufmann, 1993.
16. Ryan C., "The Degree of Oneness", in *Proc. of the 1994 ECAI Workshop on Genetic Algorithms*, Springer, 1994.
17. Vavak F., Jukes K. A., Fogarty T. C., "Performance of a Genetic Algorithm with Variable Local Search Range Relative to Frequency of the Environment Change", in *Genetic-Programming 1998: Proc. of the Third Annual Conf.*, pp. 602-608, Morgan Kaufmann, 1998.
18. Weicker K., *Evolutionary Algorithms and Dynamic Optimization Problems*, Der Andere Verlag, 2003.