An Adaptive Diploid Evolutionary Algorithm for Floating-Point Representations in Dynamic Environments

A. Şima Etaner-Uyar

Istanbul Technical University, Department of Computer Engineering
TR-34469 Maslak Istanbul, Turkey
etaner@cs.itu.edu.tr

Abstract. Many flavors of different EA approaches have been suggested in literature to address various requirements that arise when the environment is dynamic. One such line of research focuses on using memory-based techniques. However, their use is very limited and they are only considered to be advantageous when the environment periodically returns to previously visited states. In addition, most of these approaches require a change detection mechanism to adapt and their performance partly depends on an accurate detection mechanism. In this study, an implicit memory-based approach for floating-point representations is introduced.. A technique similar to estimation of distribution algorithms is used to automatically adapt the population to the change. This approach aims at addressing some of the deficiencies reported for memory-based approaches and it requires no change detection mechanism to trigger the adaptation process. The Moving Peaks Benchmark is used in the experiments to show the desired properties of the proposed approach. Preliminary experimental results are promising and promote further study.

1 Introduction

Evolutionary algorithms (EA) have been applied to a diverse set of problems. Most of the published work on EAs mainly focus on stationary environments. However, recently an increasing interest in applying EAs to dynamic environments can be seen paralleled by the increase in number of papers published on the topic. Many flavors of different EA approaches have been suggested in literature to address various requirements that arise when the environment is dynamic. A detailed survey of the EAs suggested for use in dynamic environments can be found in [1], [4] and more recently in [5].

When designing efficient EAs for dynamic environments, several aspects of the change need to be considered. These have been categorized in [4] as the severity of the change, the frequency of the change, the predictability of the change and the cycle length / cycle accuracy of the change. Each different aspect of the change has distinct properties which may form a basis in the design of a specialized EA approach, but the need to preserve diversity in the population is a key requirement for all. Loss of diversity is one of the main reasons why traditional EA approaches do not perform so well in dynamic environments. The EA approaches in literature that address diversity issues from different viewpoints can roughly be grouped as those using standard diversity preservation techniques such as sharing or crowding, those incorporating some form of a genetic memory and those that use multiple populations or other forms of parallel EAs. There are not many studies that thoroughly compare all of these approaches. One such recently published study [5] focuses on comparing several memory-based techniques with a multi-population approach. The results of the empirical comparisons show that memory-based techniques have many deficiencies when coping with the different aspects and requirements of a dynamic environment.

Much of the work proposed for dynamic environments rely on detecting the change instance to trigger the extra steps to cope with the change and their performance partly depends on an accurate detection mechanism. A trigger mechanism that produces too many "false-positives" will cause the algorithm to perform the extra steps even when there is no actual change in the environment. Since these steps mostly involve some form of a diversity increasing mechanism, applying them when the environment has not changed disrupts the ongoing search process and degrades performance. One such observation can be seen in [13] where the authors report that when the change frequency is low, the occurrence of the "false-positives" increase, possibly leading to a poorer performance. The memory-based techniques used for comparisons in [5] require some form of a detection mechanism to trigger the change, however the detection mechanism has not been explored in the paper. To avoid the performance issues caused as a result of faulty triggers, it would be better to have an approach that adapts automatically when change occurs without requiring an extra effort to detect the change.

In this study, an implicit memory-based approach for floating-point representations is introduced.. A technique similar to estimation of distribution algorithms (EDA) [11] is used to automatically adapt the population to the change. This approach aims at addressing some of the deficiencies reported for memory-based approaches and it requires no change detection mechanism to trigger the adaptation process. A floating-point, diploid representation of individuals is used to provide the necessary diversity. Diploidy is traditionally regarded as being an implicit memory mechanism, however in this study it is mainly used as a source of genotypic diversity. A genotype to phenotype mapping scheme that automatically adapts itself based on the current weighted averages of the gene values of the population is implemented. This genotype to phenotype mapping mechanism can be categorized as being adaptive [8] based on the fact that adaptation is done through feedback obtained directly from the ongoing search process. This approach will be referred to as DECAF (Diploid Evolutionary Computing with Adaptation for Floats) for the rest of the paper. Experimental runs using the Moving Peaks Benchmark are performed. This benchmark for dynamic environments was first introduced in [2] and its source code can be downloaded from [22]. The preliminary results obtained show that in its current form DECAF is comparable in performance to the best memory-based algorithm reported in [2] and in [5], but unlike similar algorithms it does not require any change detection. These promising results promote further study to enhance the automatic adaptation mechanism to reach a better performance comparable to that of the multiple population approach introduced in [3] and used for comparisons in [5].

This paper will be organized as follows: In section 2, a survey of memory-based approaches for dynamic environments will be given and their known deficiencies will be discussed. In section 3, the proposed approach DECAF will be introduced. Experimental results to explore the properties and performance of DECAF in detail will be presented in Section 4. Section 5 will conclude the paper and provide possible directions for future work.

2 Memory-Based Evolutionary Algorithms for Dynamic Environments

Memory-based approaches have been widely used in applying EAs to dynamic environments. Memory has been incorporated into an EA in two ways: as implicit memory through redundant representations or as explicit memory through using external memory to store / retrieve individuals. Introduction of a form of memory into the EA is especially useful when the change in the environment exhibits a periodic nature and returns to old states where previously found good solutions become favorable again.

For introducing implicit memory, redundant representations implemented as multiploid chromosomes with some form of a domination mechanism has been used in literature. The most known approaches that fall into this category are discussed in [7,9,10,12,14,15,16]. In [12], the authors compare two state-of-the-art diploid approaches with a simple hypermutation scheme [6] in which the mutation rate is increased for a predefined number of generations when change occurs. The diploid approaches chosen for comparisons are the Ng-Wong approach [14] and the additive diploidy approach [15]. In the Ng-Wong approach each individual has a diploid chromosome structure where each gene can have four different alleles: dominant and recessive 0 and also dominant and recessive 1. The phenotype of an individual is determined based on the dominance characteristics of the two corresponding genes for each location. In the additive diploidy approach, individuals again have a diploid chromosome structure where each gene can have several allele values each of which is assigned a numeric value. The phenotype is determined using an additive technique. For each locus on the phenotype, the corresponding values of the alleles are added. If the sum exceeds a threshold, the phenotype becomes 1, otherwise it becomes 0. The additive diploidy is easily extended to having more than two chromosomes. In [12], as a result of the preliminary experiments, the authors conclude that a dominance change is required for better performance and they modify both approaches to incorporate the required mechanism. In the Ng-Wong approach, when change occurs, the dominance characteristics of all alleles are inverted, e.g. a dominant 1 becomes a recessive 1. In additive diploidy, when change occurs, the allele values at each locus is either demoted or promoted depending on the value of the corresponding allele on the phenotype. The results obtained using the two modified diploid approaches confirm the fact that implicit memory (through diploid chromosomes) is useful when the environment oscillates between two optima. However when change occurs randomly, both diploid approaches fail and a simple hypermutation scheme is found to be more useful.

For explicit memory, some different approaches have been suggested in literature. Most of these rely upon storing some of the current information for inserting into the population in later generations after the environment changes. A detailed survey of these approaches can be found in [1], [4] and more recently in [5]. In [2], design issues of explicit memory approaches are discussed and a new approach that incorporates the advantages of using a memory together with a mechanism that has a similar effect to a random restart when change occurs is proposed. In the study, the population is divided into two sub-populations: a search population and a memory population. The search population is responsible for keeping the diversity in the population, finding new good solutions and storing them in memory and is randomly re-initialized when a change occurs. The memory population is responsible for remembering previously found good solutions. They experiment with different strategies for deciding which individuals to store in memory and which individuals to replace later when individuals are retrieved from the memory population. The experimental results obtained confirm that memory is useful for periodic dynamics in the environment and show that some method for keeping diversity within the population is required along with memory for better performance. The authors also present some open questions for future work.

3 The Proposed Approach - DECAF

As has been discussed previously, memory-based approaches have a limited use in which they are advantageous over other approaches. When the environment visits previous states, a memory becomes useful in adapting to the change and remembering old solutions, but when the environment changes randomly, the memory property may guide the population toward wrong areas in the search space and prevent discovery of new good solution candidates. This implies that it is also very important to have some form of a diversity preserving technique along with memory to improve performance. Most of the proposed methods in literature for achieving this, require that change is known by the EA to reintroduce diversity. Usually most papers either assume that change is explicitly made known to the EA or they use some form of a mechanism to detect the change. In some cases, it is realistic to assume that change is explicitly made known to the system. For example in a scheduling environment, if a new job arrives or a job leaves or a new machine is added or a machine is removed, it is usually known by the system and no detection is required. However in most problems, this assumption will not be realistic. Change has to be detected by the system. As has been previously mentioned, problems with detecting the change also affect the performance of the EA. Acting on "false-positives" usually degrades performance. Furthermore, in cases where the change occurs in the constraints rather than the problem itself, it is even harder to detect this change correctly, especially when the constraints become less tighter and the current individuals are still feasible but no longer good solution candidates.. So it is clear that an approach that does not need to know when change occurs would not suffer from the disadvantages of a weak detection mechanism.

DECAF has a diploid chromosome structure with real valued genes. Each individual has two chromosomes forming the genotype of the individual. The characteristics that are expressed, i.e. the phenotype, are obtained through a dominance mechanism. To the best of the author's knowledge, no diploidy with a dominance mechanism has been previously suggested in literature for floating point representations. Of course it is possible to use the existing approaches for binary encodings by representing the real valued genes using a binary or a gray coding; however this method will not be very efficient, first because of redundant codes due to the required number of bits to represent the numbers in the allele value range and second due to the limited fixed precision to represent the real valued genes with bits.

In DECAF, the first and second chromosomes of individuals are treated separately. A global array is used for genotype to phenotype mapping. This array keeps the dominance probabilities of the alleles on the first chromosomes of the individuals over the alleles on the second chromosomes To determine these probabilities, first of all weighted averages of the allele values at each locus are calculated for each generation using Eq. 1 where A_i shows the weighted average of the allele values at the ith locus, p_{ij} is the phenotypic value at the ith locus of the jth individual in the population and f_j is the fitness of the jth individual.

$$A_i = \frac{\sum_j (p_{ij} * f_j)}{\sum_j f_j}, \quad j = 1, 2, ..., PopSize \tag{1}$$

An allowed error percentage (err) is defined for the algorithm. A weighted proportion is calculated to determine the dominance probability of the alleles on the first chromosomes. The phenotypic values of those alleles that are within the neighborhood of the weighted average for that locus (defined by the error percentage) are multiplied by the fitnesses and are summed. This sum is divided by the total fitness of the population to determine the dominance probability of the alleles on the first chromosomes for that locus. This procedure is given in the pseudo-code in Fig. 1.

For each locus, when determining the phenotype of an individual, if the allele value of the first chromosome for that locus is within the allowed error limits, then it is expressed in the phenotype with a probability. This probability is given by the dominance vector corresponding to that locus. Otherwise the allele value on the second chromosome is expressed. If both alleles on the two chromosomes are within the allowed error limits or if both are not, then either one is selected randomly with equal probability.

```
begin

  TotalSum = Calculate_Total_Fitness();
  PartialSum = 0;
  for i=0 to ChromosomeLength
  begin
    for j=0 to IndividualCount
    begin
      if (A[i]-A[i]* err) < p[j,i] < (A[i]+A[i]* err)
        PartialSum = PartialSum + (p[j,i]*f[j]);
    end
    Dominance[i] = PartialSum / TotalSum;
  end
end
```

Fig. 1. Algorithm for calculating dominance probabilities

Gaussian mutation is used for the mutation step on the genotype. The alleles on the first chromosomes of all individuals are mutated at a rate of p_{m1} and the alleles on the second chromosomes have a mutation rate of p_{m2}. Both mutation rates are taken as the standard deviations of the normal distribution with mean 0. The first chromosomes will be used to exploit promising areas in the search space and the others will be responsible for exploration. To achieve this p_{m1} is chosen to be smaller than p_{m2}. This causes smaller changes in the alleles on the first chromosomes and bigger changes in the alleles on the second chromosomes as required by the designated roles of each chromosome. During the reproduction phase, cross-over occurs between the corresponding chromosomes of the pairs separately, i.e. the information carried on the first chromosomes and on the second chromosomes are not mixed. The mutation scheme combined with the domination mechanism, causes the algorithm to search for solutions close to the location given by the weighted averages of the current population during the times when the environment is stationary. When the environment changes, due to the difference in the new fitness values of the individuals, the weighted averages of the population change too. As a result, the allele values on the first chromosomes may no longer be within the error limits around the weighted averages. This causes the alleles on the second chromosomes to be expressed. Since these alleles are typically mutated at higher rates, this enables the population to jump in the search space. This adaptation is automatic and does not require any change detection mechanism.

This domination mechanism is an extension of a previous approach proposed by the author in an earlier work [17,18] for binary representations and later extended to non-order based multi-allelic representations in [19]. In the binary case, when determining the phenotype, the genotype elements corresponding to that location may either be equal or different. When the two alleles for the genes on the two chromosomes are the same, the corresponding phenotype equals that allele but in the case where they are different, a method to determine the phenotypic value is needed. In this implementation, alleles 1 and 0 are initially equally dominant. However, through generations the dominance factors for these alleles are modified. To define the probability of an allele being expressed, an array of real numbers in the interval $[0.0, 1.0]$ is used. The length of the array is the same as the chromosome length. Each value on the array shows the domination factor of the allele 1 over the allele 0 for the corresponding location on the chromosomes. The domination factor for an allele is interpreted as the probability of it being expressed in the phenotype. The domination array changes along with the individuals through generations and is calculated using Equation 2.

$$dom_i = \frac{\sum_j p_{ij} * f_j}{\sum_j f_j}, i = 1, 2, .., length \quad j = 1, 2, ...size \tag{2}$$

where p_{ij} is the phenotypic value of the jth individual at the ith location, f_j is the fitness value of the jth individual, $length$ is the chromosome length and $size$ is the population size. Equation 2 is evaluated at the end of each generation using the phenotype and fitness values of the individuals in that population. So the dom_i value will be higher if individuals with the allele 1 in the ith location have higher fitnesses compared to those that have allele 0. Both chromosomes are mutated at very high rates allowing for a high level of diversity on the genotypic level. This approach has been tested against state-of-the-art diploid representations with domination mechanisms in [20] and have been shown to outperform them especially when the change in the environment occurs randomly.

The domination mechanism in both the binary and integer cases as well as the floating point case is based on using the information represented by the current population. In that sense, the mechanism is similar to estimation of distribution algorithms [11] when guiding the search on the phenotypic level. However, since preserving diversity in dynamic environments is very important, the diploid chromosome structure is used to provide this. Historically, diploid representations have been used to remember previous good solutions. In DECAF, the memory property is not emphasized. In this sense, DECAF is similar to the explicit memory approach presented in [2] where there are two populations, one for searching different areas of the search space and the other for keeping a record of found good solutions. The second chromosome in the diploid structure provides the exploration feature while the first chromosome tries to exploit current good solutions. The major difference between both approaches is that in DECAF, adaptation is automatic while for the explicit memory approach, adaptation depends on the detection of the change.

4 Experiments

The same experiments as in [5] are carried out to show the performance of DECAF. The Moving Peaks Benchmark (MPB) first introduced in [2] and used in the experiments in [5] is chosen for the tests in this study. The MPB can be downloaded from [22]. For the tests, the cone function with the default base function is used. Peaks are changed every 5000 evaluations in height, width and location. For all the experiments, unless otherwise stated, the MPB default parameter settings given in Table 1 are used.

Table 1. Default parameter settings for the MPB used in the experiments

Parameter	**Value**
change frequency	5000 evaluations
shift vector length (s)	1.0
lambda	0.5
number of peaks	10
number of dimensions	5
dimension value range	[0, 100]
height severity	7.0
width severity	0.01
height (max,min,std)	70,30,50
width (max,min,std)	7.0,0.8,1.0
random number seed	1

The results are reported based on 50 runs of all the programs. Plots of the offline error and the percentage of peaks covered on the genotypic level through generations, all averaged over 50 runs are given. Graphs are given as error bars where the average and standard deviation of the 50 runs will be plotted.

Offline error is defined in [5] as the average current error over all time steps. Current error is the difference between the fitness of the current best individual (since last change) and the actual optimum. Offline error is calculated as in Eq. 3.

$$e^*(T) = \frac{1}{T}\sum_{t=1}^{T} e_t \qquad where\ e_t = opt(t) - f_{best}(t) \tag{3}$$

The population consists of 100 individuals which are initialized randomly. Each individual has a diploid representation with real valued genes. Generational replacement with an elitism of one individual per generation is used. For each generation, the best individual from the previous generation replaces the worst individual in the current generation if it has a higher fitness in the current environment. Fitness values of the individuals are calculated based on their phenotypes but the variation operators act directly on the chromosomes, i.e. the genotype. Gaussian mutation is used with standard deviations given as $p_{m1} = 0.2$ for the first chromosomes and as $p_{m2} = 10$ for the second chromosomes. The values of p_{m1} and p_{m2} are determined empirically to give good performance. Each dimension represented by each gene takes on values between 0 and 100. In the normal distribution, 95% of the values are within 1.96 standard deviations of the mean. So a choice of 10 for the second mutation rate should provide an acceptable amount of diversity. Two-point cross-over between the first chromosomes and also between the second chromosomes of reproducing pairs is used each with a rate of $p_c = 0.6$ where the two cross-over locations are selected randomly with equal probability for all loci. For mating pool selection, tournament selection with tournament sizes of two is used. All programs are run for a total of 1000 generations each. Since there are 100 individuals in the population, this means that change occurs every 50 generations. For each test case, the EA starts with a different random number seed for the 50 runs whereas the same seed is used for the MPB to obtain the same environment for all runs of the algorithm. For the EA implementation, the GNU Scientific Library [21] implementations of the random number generators and the normal distribution is used with the Mersenne-Twister random number generator chosen to sample from the uniform distribution. For the MPB, to make results comparable with those reported in previous studies, the random number generators included in the benchmark are used.

In [5], the experiments consist of three parts. In the first part, the effect of the change severity is explored. In the second part, the effect of recurrence of optima is tested. And in the final part, the effect of increasing the number of peaks is explored. To be able to make comparisons, the same set of experiments, explained in the following subsections, are performed in this study too. In all experiments, unless otherwise stated, all parameters of the MPB are as given in Table 1.

To test the effect of the change severity, the length of the shift vector in the MPB is varied between 0 and 3. The offline error at the end of 1000 generations, averaged over 50 runs can be seen in the plot in Fig. 2. The percentage of peaks covered at all generations by the genotype space of the populations, averaged over 50 runs, for a shift vector of length $s = 1.0$, is given in Fig. 3. The offline error plot in Fig. 2 is comparable to the one given in [5] for the explicit memory approach and shows similar performance with different change severity levels. So it can be said that performance is not highly dependent on the severity of the change. In Fig. 3, the plots for the percentage of peaks covered on the genotypic level is given. It can be seen from this figure and Fig. 2 that while the percentage of peaks covered on the genotypic level is not very high, the adaptation mechanism is sufficient for keeping a moderately low offline error and this remains consistent over different change severity levels.

To test the effect of recurrence, the value of the lambda parameter in the MPB is varied between 0 and 1. Increasing the value of lambda will make the direction of the move more dependent on the current direction, making recurring optima less likely. In the previous test case, lambda was chosen as 0.5, causing the peaks to move in a random direction at each change instance. The offline error plot can be seen in Fig. 4. The performance of DECAF is again very similar to the explicit memory approach plot given in [5]. It should be noted that the offline performance is not affected too much from the selection of lambda. This supports the fact that the memory feature of diploidy works together with the adaptation and diversity preserving mechanisms to be able to cope with different dynamics as represented by different values of lambda.

To see the effect of changing the number of peaks, the number of peaks parameter in the MPB is varied between 10 and 200. The offline error plot can be seen in Fig. 5 and the percentage of peaks covered on the genotypic level for 10, 50 and 175 peaks can be seen in Fig. 6. It can be seen that even though the percentage of peaks covered by the genotype decreases as the number of peaks are increased, the offline performance is not affected too much.

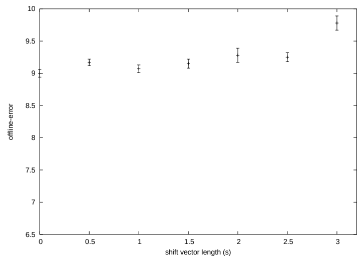

Fig. 2. The effect of increasing the shift vector length (s)

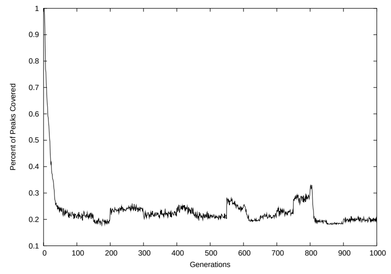

Fig. 3. Percentage of peaks covered by genotype at s=1.0 and 10 peaks

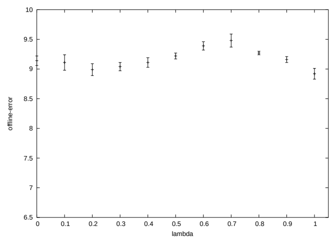

Fig. 4. The effect of increasing lambda

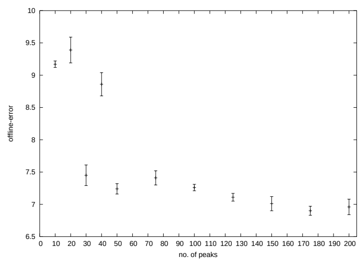

Fig. 5. The effect of increasing the number of peaks

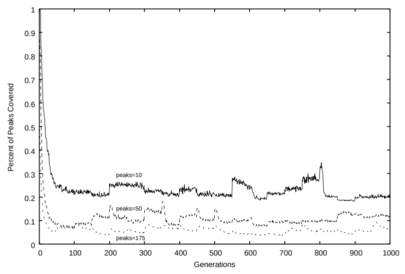

Fig. 6. Percentage of peaks covered by genotype at 10, 50 and 175 peaks

5 Conclusion and Future Work

An adaptive evolutionary algorithm for dynamic environments, using diploid chromosomes with a floating-point encoding and an adaptive genotype to phenotype mapping mechanism is introduced in this study. Through experiments, it is found to perform similarly to an explicit memory approach which was shown to perform better than other memory-based approaches in [5]. As has been reported in [5], memory-based approaches have a very limited use in dynamic environments but the addition of a diversity maintaining mechanism seems to improve performance. However, as the experiments in that study also indicated, currently multiple population approaches perform much better than the best memory-based approach tested. In this study, DECAF is shown to reach the performance of the explicit memory approach reported in that study, but it too is not able perform as well as the multi-population approach. Even though they exhibit similar performance characteristics, DECAF is better than the explicit memory approach of [5] because adaptation in DECAF is automatic and thus performance will not be degraded by possible "false positives" generated by the change detection mechanism.

Even though this study reports the preliminary results of DECAF, these results are quite promising and promote further study. In this study, parameter settings, especially the mutation rates of both chromosomes, have been done empirically to provide good performance. This should be further explored to test the dependency of the performance on the choice of parameters. A better adaptation mechanism will possibly be able to provide a more robust solution.

The results of this study show that an approach similar to estimation of distribution algorithms (EDA) provides promising adaptation properties when used along with a diversity maintenance mechanism. So it seems a worthwhile future work to explore EDAs in greater depth to improve the performance of DECAF. It also may be useful in obtaining better performing EA implementations if EDAs can be incorporated into multi-population approaches to improve their convergence properties during stationary periods.

References

1. Branke J., "Evolutionary Algorithms for Dynamic Optimization Problems - A Survey", Technical report No. 387, AIFB Institute, University of Karlsruhe - Germany, (1999).
2. Branke J,. "Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems", in *Proceedings of the Congress on Evolutionary Computation* 1999, pp. 1875-1882, IEEE, (1999).
3. Branke J., Kaussler T., Schmidt C., Schmeck H., "A Multi-Population Approach to Dynamic Optimization Problems", in *Proceedings of Adaptive Computing in design and Manufacturing 2000*, pp. 299-308, Springer, (2000).
4. Branke J., *Evolutionary Optimization in Dynamic Environments*, Kluwer, (2002).
5. Branke J., Schmeck H., "Designing Evolutionary Algorithms for Dynamic Optimization Problems", in Tsutsui S, Ghosh A. (eds) *Theory and Application of Evolutionary Computation: Recent Trends*, pp. 239-262, Springer, (2003).

6. Cobb H. G., "An Investigation into the use of Hypermutation as an Adaptive Operator in Genetic Algorithms having a Continuous, Time-Dependent Non-Stationary Environments", Technical Report No. AIC-90-001, Naval research Laboratory, (1990).
7. Collingwood E., Corne D., Ross P., "Useful Diversity via Multiploidy", in *Proceedings of AISB Workshop on Evolutionary Computation*, (1996).
8. Eiben A. E., Hinterding R. , Michalewicz Z., "Parameter Control in Evolutionary Algorithms", IEEE Transactions on Evolutionary Computation, Vol. 3, No.2, pp. 124-141, IEEE, (1999).
9. Greene F., "A Method for Utilizing Diploid/Dominance in Genetic Search", in *Proceedings of the First IEEE Conference on Evolutionary Computation*, (1996).
10. Hadad B. S., Eick C. F., "Supporting Polyploidy in Genetic Algorithms using Dominance Vectors", in Proceedings of the Sixth International Conference on Evolutionary Programming, Lecture Notes in Computer Science Vol. 1213, Springer, (1997).
11. Larranaga P., Lozano J. A., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer, (2001).
12. Lewis J., Hart E., Graeme R., "A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems", in Proceedings of Parallel Problem Solving from Nature, Lecture Notes in Computer Science Vol. 1498, Springer, (1998).
13. Morrison R. W., De Jong, K. A., "Triggered Hypermutation Revisited", in *Proceedings of the Congress on Evolutionary Computation 2000*, pp. 1025-1032, IEEE, (2000).
14. Ng K. P., Wong K. C., "A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization", in Proceedings of the Sixth International Conference on Genetic Algorithms, (1995).
15. Ryan C., "The Degree of Oneness", in Proceedings of the 1994 ECAI Workshop on Genetic Algorithms, Springer, (1994).
16. Smith R. E., Goldberg D. E., "Diploidy and Dominance in Artificial Genetic Search", Complex Systems, Vol. 6, pp. 251-285, (1992).
17. Uyar A. S., Harmanci A. E., "Investigation of New Operators for Diploid Genetic Algorithm", in Proceedings of SPIE: Application and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation II, (1999).
18. Uyar A. S., Harmanci A. E., "Preserving Diversity through Diploidy and Meiosis for Improved GA Performance in Dynamic Environments", Lecture Notes in Computer Science Vol. 2457, pp. 314-323, Springer, (2002).
19. Uyar A. S., Harmanci A. E., "An Adaptive Domination Map Approach for Multi-Allelic Diploid Genetic Algorithms", Genetic and Evolutionary Computation Conference - GECCO 2003. Late Breaking Papers, pp. 291-298, (2003).
20. Uyar A. S., Harmanci A. E.,"Comparison of Domination Approaches for Diploid Binary Genetic Algorithms", Application and Science in Soft Computing: Advances in Soft Computing, pp. 75-80, Springer, (2004).
21. http://www.gnu.org/software/gsl/ : GNU Scientific Library
22. http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/ : The Moving Peaks Benchmark.