Preserving Diversity through Diploidy and Meiosis for Improved Genetic Algorithm Performance in Dynamic Environments

A. Sima Etaner-Uyar[1] and A. Emre Harmanci[2]

[1] Istanbul Technical University, Istanbul 80626, Turkey
etaner@cs.itu.edu.tr
[2] Istanbul Technical University, Istanbul 80626, Turkey

Abstract. Genetic algorithms have been applied to a diverse field of problems with promising results. Using genetic algorithms modified to various degrees for tackling dynamic problems has attracted much attention in recent years. The main reason classical genetic algorithms do not perform well in such problems is that they converge and lose their genetic diversity. However, to be able to adapt to a change in the environment, diversity must be maintained in the gene pool of the population. One approach to the problem involves a diploid representation of individuals. Using this representation with a dynamic dominance map mechanism and meiotic cell division helps preserve diversity. In this paper, the effects of using diploidy and meiosis with such a dominance mechanism are explored. Experiments are carried out using a variation of the 0-1 knapsack problem as a testbed to determine the effects of the different aspects of the approach on population diversity and performance. The results obtained show promising enhancements.

1 Introduction

Genetic algorithms have been applied to a diverse field of problems with promising results. While most of these mainly address stationary problems, there's another group where the problem is dynamic and is represented by a changing fitness function. This class of problems are characterized by a need for a mechanism to adapt to the change. Different characteristics of changing fitness functions can be exploited in different ways to obtain a near optimal solution. Criteria for categorizing dynamic environments have been given as follows [2]:

- frequency of change
- severity of change
- predictability of change
- cycle length / cycle accuracy

The development of a modified algorithm which would perform better than the others in specific types of problems may be accomplished by first using the above given criteria for categorizing the type of change in the environment and then

T. Yakhno (Ed.): ADVIS 2002, LNCS 2457, pp. 314–323, 2002.

deciding on an appropriate approach. However, no matter into which category the problem falls, diversity plays an important role in performance. The main reason why traditional algorithms do not perform well in such cases is that they converge to a solution and the population loses its genetic diversity. Consequently when a change occurs it is hard for the converged population to adapt. There are different ways to achieve the diversity needed in the population. The main approaches in coping with changing environments are summarized in [2] and discussed in [6].

In this study, a diploid representation is used to maintain the desired diversity. However, as shown in [9] diploidy alone is not sufficient and other mechanisms combined with an adaptive dominance map is needed. This paper focuses on the importance of preserving diversity in the population and how different parts of the chosen diploid algorithm contributes to this end.

2 The Diploid Algorithm and Population Diversity

The diversity in a population is lost when the gene pool of the population loses its diversity. This is closely related to the concept of convergence as defined in [1]. According to the definition, a locus is said to converge if at least 95% of the individuals have the same allele at their corresponding loci. And a population is said to converge if all loci of the chromosomes have converged. At this point, diversity in the gene pool of the population is lost.

Since a standard genetic algorithm converges to a solution after several generations (depending on the genetic algorithm parameters chosen and population size), it may not be able to follow a change that occurs after this stage. The only mechanism a standard genetic algorithm has for introducing diversity into the gene pool is mutation and the probability of mutation is typically chosen to be very small. Other mechanisms or variations are needed to maintain genotypic diversity in a population.

2.1 Representation and the Domination Mechanism

In the algorithm chosen for this study, each individual is represented with three chromosome strings and a fitness value. Chromosome 1 and chromosome 2 are homologues and form the diploid genotype of the individual. The third string which is the phenotype, shows the characteristics that are expressed. In this implementation, the chromosome and the phenotype strings each are made up of either a 1 or a 0 at each location.

The phenotype of the individual is the set of characteristics that are expressed. The fitness is determined using the phenotype. Therefore a mechanism to map the genotype onto the phenotype is needed. This is a very important part of diploid genetic algorithms and there has been some research done most of which are explained in detail in, [3], [4], [5], [6], [7], [8], [9], [10], [11], [12].

When determining the phenotype, the genotype elements corresponding to that location may either be equal or different. In the cases where the two alleles for the genes on homologue 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 natural organisms the dominant allele is seen in the phenotype, so a mechanism to simulate this in artificial systems will be used. In this implementation, a domination array composed of real numbers in $[0.0, 1.0]$ is used. The length of the array is the same as the chromosome length with each value showing the dominance factor of the allele 1 over the allele 0 corresponding to the same location on the chromosomes. For example, if the alleles on the two chromosomes are different for the ith location and if the ith entry in the domination array is $dom_i = 0.8$, the phenotypic value for that location will be 1 with probability 0.8 and 0 with probability 0.2.

The domination array evolves along with the individuals in each population and is calculated using Equation 1.

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

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 (the total number of individuals in the population). Equation 1 is evaluated at the end of each generation using the phenotype and fitness values of the individuals in that population.

2.2 The Main Steps of the Algorithm

The diploid algorithm chosen as the basis for this study is explored in greater detail and compared against other approaches in [13], where overlapping populations with the introduction of an aging mechanism and a possible random replacement of aged individuals is used. However, for the purposes of this study, the offspring replace their parents causing no overlapping between populations in consecutive generations and thus there's no aging and random replacement of older individuals. The simplified algorithm used here is given below.

```
begin
initialization;
do
  reproduction;
  mutation;
  calculation of new domination vector;
until stop;
end.
```

The *initialization* step is similar to the one in the simple genetic algorithm. Each of the genes on the two chromosomes is initialized randomly to be a 0 or a 1. All locations on the domination array are initialized to 0.5. After this step, the phenotypes of the individuals are determined using the initial domination array and the fitnesses are calculated based on the phenotypic values.

The *reproduction* phase consists of selection of the mating pairs, gamete formation through meiosis, pairing off and the actual mating phase to form the offspring. A roulette wheel selection mechanism is used to determine the individuals which will go into reproduction. Gametes in natural, diploid organisms are the haploid reproductive cells. One haploid gamete from each mating pair comes together to make up the diploid cell of the offspring. In most cases in nature, gamete formation is the result of a cell division process called meiosis. In this artificial implementation, each parent goes through a meiotic cell division separately. In the first step of meiosis, a copy of each chromosome string is made during which errors may occur. The chromosome and its copy are called sister chromatids. At the end of this step, the individual has four haploid chromatids. In the second step, crossing over may occur between non-sister chromatids. In this implementation, a two point cross-over approach is used. In the final step, after each mating parent completes its meiosis-like process, there are four gametes from each parent, ready to go into mating. Since each mating produces two offspring, two gametes from each parent are selected at random and each gamete from each parent goes to each one of the offspring. As a result of each mating, two offspring are produced which replace their parents in the population.

The *mutation* operator is as defined in the simple genetic algorithm. Mutation acts directly on the genotype, i.e. for the diploid case, on each gene of the two strands of chromosomes.

At the end of each generation, the new domination array is calculated using Equation 1. This new array will be used in the next generation for determining the phenotypes of the individuals in that population from their genotypes.

3 Tests for Effects on Diversity

The complete algorithm used in these tests is explained in Section 2. To see the effects of the diploid representation and the meiotic cell division on performance and diversity separately, this algorithm is modified. The modified algorithm consists basically of the same steps as given in Section 2. However, in the reproduction phase, gametes are not obtained using meiotic cell division but chosen directly as the chromosomes of the parents. Tests are performed to compare the full diploid algorithm, the modified version (without meiosis) and the simple (haploid) genetic algorithm with regard to how they preserve diversity in the genotypic and phenotypic levels and how they perform in following the change.

3.1 The Test Function

A modified version of the 0-1 knapsack problem defined in [12] is used. Mathematically the problem can be represented as finding

$$max \sum_{i=1}^{n} v_i x_i \qquad \text{subject to the weight constraint} \qquad max \sum_{i=1}^{n} w_i x_i,$$

where x_i are variables that can either be 1 or 0, v_i and w_i are given problem specific parameters, n is the problem size and W is the weight constraint.

It is important to note that the chosen knapsack has only one solution for each possible weight constraint. In the modified knapsack problem used in this study, change in the environment is implemented by changing the weight constraint at random intervals. It must be noted that this change is not periodic and is not predictable. In order to have controlled experiments, the change instances and the values for the weight constraint for that instance is kept the same for each test case. These are given in Table 1 where each row gives the generation number when the change occurs, the value for the new weight constraint, the Hamming distance between the previous and the new solutions which is calculated as the number of positions which have different values and the optimal solution values and strings at that change instance.

Table 1. Change Instances and Values

i	Genr. No.	Weight Cons.	Hamming Dist.	Opt. Sol. Val.	Opt. Sol. String
0	0	115	0	131070	01111111111111111
1	50	40	9	128512	00000000011011111
2	525	11	4	110600	00010000000011011
3	1997	22	1	112648	00010000000111011
4	2027	23	5	114688	00000000000000111
5	2357	100	12	131066	01011111111111111
6	2845	36	8	128032	00000100001011111

The same set of genetic algorithm parameters, where applicable, are used in each test case. The population size is 250, the maximum number of generations is 3000, the crossing over probability is $p_{cross} = 0.9$, the mutation probability is $p_{mut} = 0.001$ and the probability of copy error in meiosis is $p_{err} = 0.001$.

4 Experimental Results

The results will be given as tables and plots of performance for the simple genetic algorithm (SGA), the algorithm with only diploidy and the adaptive dominance map (BareGA) and the full algorithm with diploidy, dominance and the meiotic cell division (withMeiosGA). The tables list the optimal solutions, the best solution values found by the algorithm, the number of steps after the change it took the algorithm to obtain the values and the percentage error of the found solutions to the optimals for each generation interval. In the performance plots, the x-axis shows the number of generations and the y-axis shows the solution values found. In changing environments, finding better and acceptable solutions quickly is as important as finding the optimal solutions. As can be seen in the plots and tables, even though it may take longer for a specific algorithm to find its best result, it may have found acceptable results much earlier.

Table 2. SGA results

Generations	Opt. Sol.	Best Val.	Steps	Err.%
[0,50)	131070	130059	48	0.77%
[50,525)	128512	127240	16	0.99%
[525,1997)	110600	0	0	100%
[1997,2027)	112648	102530	26	8.98%
[2027,2357)	114688	111744	213	2.57%
[2357,2845)	131066	127624	487	2.63%
[2845,3000)	128032	127488	74	0.42%

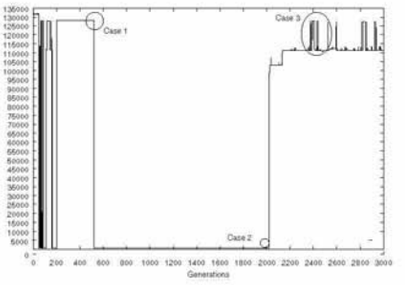

Fig. 1. Plot for SGA

Table 3. BareGA results

Generations	Opt. Sol.	Best Val.	Steps	Err.%
[0,50)	131070	131070	1	0%
[50,525)	128512	128512	4	0%
[525,1997)	110600	110600	1	0%
[1997,2027)	112648	112640	1	0.007%
[2027,2357)	114688	112648	55	1.78%
[2357,2845)	131066	130034	481	0.02%
[2845,3000)	128032	128032	66	0%

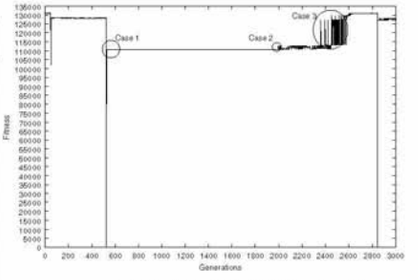

Fig. 2. Plot for BareGA

Table 4. withMeiosGA results

Generations	Opt. Sol.	Best Val.	Steps	Err.%
[0,50)	131070	131070	0	0%
[50,525)	128512	128512	111	0%
[525,1997)	110600	110600	112	0%
[1997,2027)	112648	112648	20	0%
[2027,2357)	114688	112648	23	1.78%
[2357,2845)	131066	130066	160	0%
[2845,3000)	128032	128032	2	0%

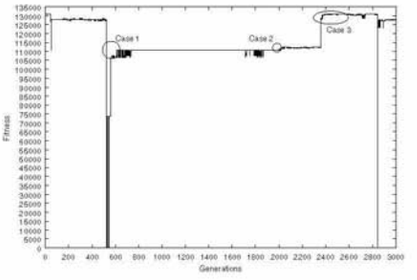

Fig. 3. Plot for withMeiosGA

Next, three change instances chosen according to the nature of the change will be examined more closely to understand the way each algorithm acts. These three change instances are marked on the general plots.

4.1 Case 1

The first instance is at generation 525. This change occurs 475 generations after the previous one and the hamming distance is 4. At the time of the change, for the SGA, 15 loci have converged in the population; for the BareGA, 7 have

converged in the genotype and 15 in the phenotype; for the withMeiosGA, 2 have converged in the genotype and 10 in the phenotype.

The SGA finds the new solution in 57 generations (Fig. 4), the BareGA in 17 generations (Fig. 5) and the withMeiosGA in 7 generations (Fig. 6). This result is directly related to the amount of loci converged in each population. For the two cases of the diploid algorithm, even though the population is almost at the same level of phenotypic convergence with the SGA, on the genotypic level, both have preserved their population diversity to a greater extent. The diversity on the genotypic level helps the population to adapt to a change much quicker.

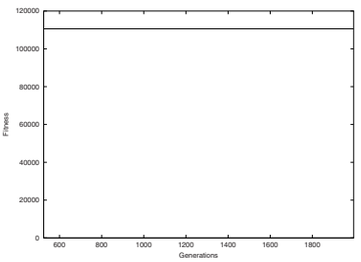

Fig. 4. SGA after genr. 525

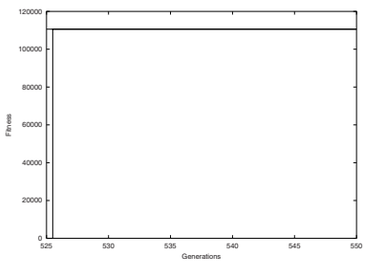

Fig. 5. BareGA after genr. 525

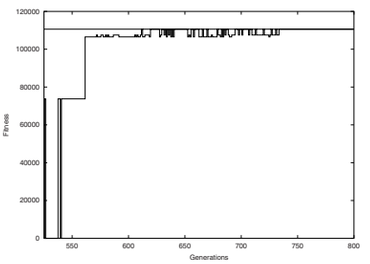

Fig. 6. withMeiosGA after genr. 525

4.2 Case 2

The second instance is at generation 2027. This change occurs 30 generations after the previous one and the hamming distance is 5. At the time of the change, for the SGA, 15 loci have converged in the population; for the BareGA, 6 have converged in the genotype and 12 in the phenotype; for the withMeiosGA, 1 has converged in the genotype and 9 in the phenotype.

The SGA and BareGA can not find a solution (Fig. 7 and Fig. 8); they get stuck at the solution found for the previous change instance. The withMeiosGA cannot find the exact solution but it is able to find a better solution (Fig. 9).

This result is directly related to the amount of loci converged at the time of the change. The fact that the change occurs very shortly after the previous one also plays a role in the results obtained. The converged loci and the severity of the change, represented by the hamming distance, are similar to the previous case.

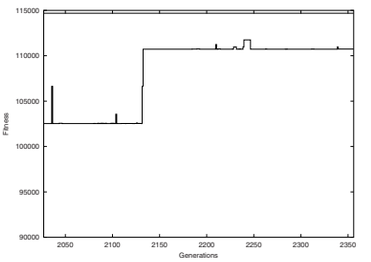

Fig. 7. SGA after genr. 2027

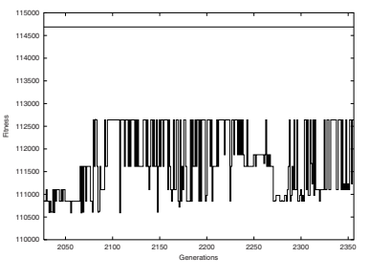

Fig. 8. BareGA after genr. 2027

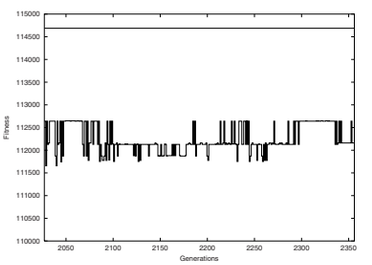

Fig. 9. withMeiosGA after genr. 2027

4.3 Case 3

The third instance is at generation 2357. This change occurs 330 generations after the previous one and the hamming distance is 12. At the time of the change, for the SGA, 16 loci have converged in the population; for the BareGA, 3 have converged in the genotype and 14 in the phenotype; for the withMeiosGA, 2 have converged in the genotype and 11 in the phenotype.

Again in this case none of the three algorithms can find the exact solution (Fig. 10, 11 and 12). However, the withMeiosGA finds a solution better than the others and finds it much quicker than the other two. The SGA finds its best solution around generation 2800. The BareGA finds its best solution around generation 2620. The withMeiosGA finds its best solution around generation 2420. The amount of loci converged in each populations still plays an important role but the main difference from the previous two cases is the severity of the change characterized by the high hamming distance.

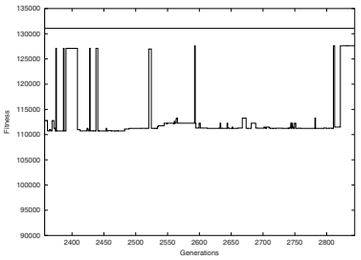

Fig. 10. SGA after genr. 2357

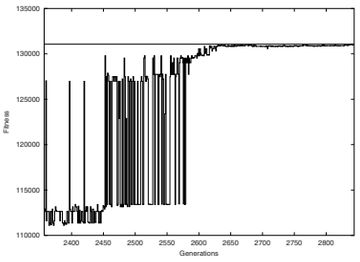

Fig. 11. BareGA after genr. 2357

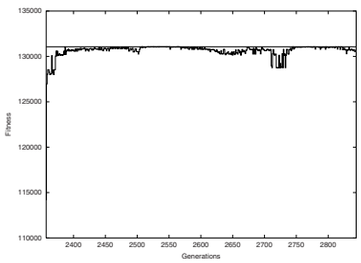

Fig. 12. withMeiosGA after genr. 2357

5 Conclusion

The algorithms chosen in this study have different features which address different aspects of a changing environment. Each feature contributes in in preserving the diversity in the population and tracking the change The graphs for each case given in the previous sections are in keeping with those that are to be expected for each case. The change instances that were explored can be grouped as follows:

- Change occurs after a moderate amount of generations and the severity of the change is moderate.
- The number of generations between two change instances is very low and the change is again moderately severe.
- Change occurs again after a moderate amount of generations but the severity of the change is quite high.

All three cases require a high amount of diversity to be present in the population. The first and third cases show similar characteristics: the change occurs after a moderate amount of generations have elapsed, giving the populations time to converge to a solution. The convergence rates in the SGA are similar to the rates for both diploid cases on the phenotypic level. Since fitness is based

on the phenotype of an individual, the fact that the SGA and the diploid algorithms perform as well in cases when there's no change is important. The domination array directs the convergence rate on the phenotypic level. However on the genotypic level, the diploid cases show greater diversity; so when a moderately severe change occurs, the diversity preserved in the genotype helps the population to adapt to the change much quicker than the haploid SGA which has almost totally converged. The difference between the first and third cases arises from the severity of the change. In the third case the hamming distance between the solutions is quite high, making it harder for the population to adapt to. In this case diversity plays an even more important role. The SGA is again almost converged but the both diploid cases have quite a low number of converged loci in the gene pool and they have preserved their diversity to a greater extent. This makes them more robust and allows them to adapt to the change much quicker.

References

1. Beasley, D., Bull, D., Martin, R.: An Overview of Genetic Algorithms, Part 1: Fundamentals. University Computing. **15(2)** (1993) 58–69
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers. (2002)
3. Cobb, H. G.: An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent, Nonstationary Environments. Naval Research Laboratory Technical Report. **AIC-90-001** (1990)
4. Collingwood, E., Corne, D., Ross, P.: Useful Diversity via Multiploidy. AISB Workshop on Evolutionary Computation. (1996)
5. Greene, F.: A Method for Utilizing Diploid/Dominance in Genetic Search. Proceedings of the First IEEE Conference on Evolutionary Computation. (1996)
6. Grefenstette, J. J.: Genetic Algorithms for Changing Environments. Parallel Problem Solving from Nature 2. North Holland. (1992) 137–144
7. Hadad, B. S., Eick, C. F. Supporting Polyploidy in Genetic Algorithms Using Dominance Vectors. Proceedings of the Sixth International Conference on Evolutionary Programming of LNCS Springer Verlag. **1213** (1997)
8. Kim, Y., Kim, J. K., Lee, S., Cho, C., Hyung, L.: Winner Take All Strategy for a Diploid Genetic Algorithm. Proceedings of the First Asia-Pacific Conference on Simulated Evolution and Learning. (1996)
9. Lewis, J., Hart, E., Ritchie, G.: A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems. PPSN'98. **1498** (1998) 139–148
10. Ng, K. P., Wong, K. C.: A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization. Proceedings of the Sixth International Conference on Genetic Algorithms. (1995)
11. Ryan, C.: The Degree of Oneness. Proceedings of the 1994 ECAI Workshop on Genetic Algorithms, Springer Verlag. (1994)
12. Smith, R. E., Goldberg, D. E.: Diploidy and Dominance in Artificial Genetic Search. Complex Systems. **6** (1992) 251–285
13. Uyar, A. S., Harmanci, A. E.: Investigation of New Operators for a Diploid Genetic Algorithm. Proceedings of SPIE: Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation II. (1999)