Addressing Change within a Generation

Jürgen Branke[1], Gülshat Kulzhabayeva[2], and Şima Uyar[2]

[1] Institute AIFB, University of Karlsruhe
76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

[2] Istanbul Technical University, Computer Engineering Department
Maslak 34469 Istanbul, Turkey
etaner@cs.itu.edu.tr

Abstract. Many practical optimization problems change over time, requiring a repeated re-adaptation of the solution. As has been shown in numerous papers, evolutionary algorithms (EAs) can be modified so that they can cope well with dynamic environments. However, basically all papers so far either considered a continuous change, or a change that happens in intervals which coincide with the time to complete a number of generations. In reality, changes may occur at any time and irregular intervals, which raises the question how to deal with a change occurring within a generation of an EA. Different methods for handling such changes are given and experimentally compared in this study.

1 Introduction

Many optimization problems are dynamic and change over time. This means that a good optimization algorithm should be able to follow these changes and adapt the solution each time the environment changes. In nature, species evolve to adapt better to the environments in which they survive. Over time, these environmental conditions may change favoring other individuals over the currently better ones for reproduction and survival. Thus natural evolution can be regarded as adaptation in an inherently dynamic and stochastic environment. Evolutionary algorithms (EAs) are inspired from Darwin's theory of evolution and classical Mendelian genetics. Based on these, EAs seem naturally suited to be used in dynamic optimization problems. Although recent research has shown that the standard EAs are prone to convergence and fail to adapt continuously, a number of ways have been suggested to circumvent this problem and the resulting EA variants show great promise. For detailed overview on dynamic environments, see [1], [3] and [4].

Basically all papers in the area of evolutionary optimization in dynamic environments so far either assume a continuously changing environment, or changes that occur only at the end of a generation. In reality, however, environmental changes are of course independent of the EA and can occur at any time, i.e., also during a generation [2] . This distinction is important especially in quickly changing environments, where only very few generations can be computed between environmental changes.

Technical Report, Institute AIFB, University of Karlsruhe
76128 Karlsruhe, Germany, March 2008

This paper mainly deals with EAs in quickly changing environments, i.e. the time between the changes is relatively small. In nature, changes do not usually occur in a planned way. In order to know how to deal with the problem after a change occurs in a quickly changing environment, we have to examine these changes as they occur in nature, i.e are stochastic. Thus it is equivalent to having changes within generations in EA design. In order to clarify, it is known that EAs are iterative algorithms, so in each *generation*, a number of new solutions are generated, evaluated, and inserted into the population. Previous work on EAs for dynamic environments assume that the environment changes between generations. Although this assumption is convenient, Branke and Wang [2] consider it as an oversimplification, because of the case that generally the environment is independent of the EA, and thus can change at any time, i.e. also within a generation. If the changes are frequent, this aspect should not be neglected.

The main aim of this study is to empirically compare different approaches to handle changes which occur before the current generation ends. The paper is structured as follows: In section 2, an overview of related work is provided. Section 3 introduces the methods proposed to handle changes within a generation while in section 4, a detailed explanation of the experimental setup and the tests performed is provided with the corresponding test results. Section 5 provides a summary and concludes with some ideas for future work.

2 Related Work

Previously all work in dynamic EA literature treat the changes as occurring between generations. However, the environment in the real world is independent of the EA, and thus can change at any time which sometimes may be within a generation. Usually, when applying EAs, a model of the environment is maintained and candidate solutions are evaluated using this model. Treating a change which occurs *within* a generation as occurring *between* generations corresponds to ignoring the change until the current generation is completed and updating the model at the end of the generation.

Initial work on dealing with changes within a generation was done by Branke and Wang in [2]. They analytically compared two very simple approaches to deal with a change of the fitness function within a generation. These approaches are:

- keep the individuals evaluated with the old fitness function and use the new fitness function for the remaining individuals in the current generation
- ignore the change and evaluate all individuals until the end of the current generation using the old fitness function

In the first method, during the selection stage, individuals evaluated with different fitness functions are compared. If the environment change is severe, this may cause wrong individuals to be selected. For the second method, delaying the change would have no unwanted effects if the relative ordering of the individuals in the new environment remains the same as in the old one. For the analysis, they used a simple (1,2) evolutionary strategy on a dynamic bit matching problem. The dynamic bit matching problem (dBMP) may be seen as an extension of the One-Max problem commonly used to test EAs in stationary environments. In dBMP, the bit template to be matched by the EA is changed each time an environmental change occurs. The fitness of a candidate solution is calculated based on the number of bits it matches in the given bit template. The conclusions derived from the results of the analytical study show that depending on the current fitness value and the severity of the environmental change, one method or the other can be preferred.

3 Handling Change within a Generation

The study in this paper extends the work done in [2] by introducing other strategies to deal with changes within a generation and using larger population sizes. We compare the different strategies empirically, again using the dynamic bit matching problem. The actual approaches tested depend on the reproduction scheme of the EA. In EA literature, two such schemes are commonly used [5]: the generational reproduction scheme and the steady-state reproduction scheme. In this study, we will deal with strategies proposed to handle changes within a generation in a generational reproduction scheme framework.

In the generational reproduction scheme, in every generation, n new individuals (with n being the population size) are generated usually through selection, crossover and mutation. These individuals replace the individuals in the current generation and they form the population in the next generation. Often, this mechanism is used in conjunction with *elitism* where the best e individuals are retained from the current population and only $n - e$ new individuals are generated.

If the fitness changes between the generations, it is sufficient to only re-evaluate the elite individuals for consistency. If a change occurs while the offspring in the current generation are being evaluated, several options exist:

- Method 1 (M1): use the changed fitness function for all subsequent individuals, but keep the evaluations of the offspring already evaluated
- Method 2 (M2): temporarily reduce the population size, terminate the generation and use the offspring generated so far as basis to generate the next
- Method 3 (M3): re-evaluate all offspring already generated using the new fitness function
- Method 4 (M4): ignore the change and continue to work with the old fitness function until all offspring of the current generation have been evaluated

In the rest of the paper, we will empirically compare these four approaches under several different settings.

4 Experiments

In the experiments, a simple generational GA (GGA) is applied to the dBMP where an environment is defined by a string of bits of length l. The goal of the

GGA is to evolve a solution string that matches the target string in all locations. Thus the fitness of a candidate solution is determined by the number of bit locations it matches in the target string. An environment change is implemented by flipping $k\%$ bits in the target solution. The magnitude of k determines the change severity.

4.1 Algorithm Details and Parameter Settings

The GGA is used with elitism where the best individual from the current generation is transferred directly to the next generation. Duplicate elimination is applied in all stages of individual generation and insertion into the population. This ensures a sufficient level of diversity. Since the focus of this paper is on comparing the different techniques for handling a change within a generation, we do not employ any extra diversity introducing/maintaining mechanism so that the actual effects of each technique is not blurred by these mechanisms.

The target string length in dBMP is taken as 100 which means that the chromosome length is 100. We use a population size of 100, tournament selection with tournament size of 2, uniform crossover with probability $p_c = 0.8$, and mutation rate $1/L$ where L is the chromosome length, i.e., $p_m = 0.01$.

In our base experiments we use a change severity of 0.1 which means that for each change, approximately 10 bits, which are randomly selected, are flipped in the target string. Three different settings for the change period are used to see the effect of the number of fitness evaluations between changes. If we denote the change period as p, we experimented with periods of $p = 3, 10, 25$. Again in the base experiments, the change occurs in the middle of each generation, i.e. after 50 individuals have been evaluated for that generation. If we interpret this as an offset x (i.e. $x = 50$), the number of fitness evaluations e between changes is calculated as $e = p * 100 + x$ where 100 is the population size. This means that, for the experiments $e = 350, 1050, 2550$ settings are used. For the BMP, these settings correspond to very short, medium length and long change periods respectively.

Initially, the effect of each of the four mechanisms explained in Section 3 is tested in the presence of only two environments, i.e. only one change after e evaluations. For a fair comparison between the methods, an equal number of fitness evaluations after a change is allowed in each case for recovery. In our base experiments, this value is chosen as 300 fitness evaluations.

Results are compared based on the best fitness values. All results, unless stated otherwise, are reported using averages and standard-error values over 1000 runs with random seeds using the same change pattern.

4.2 Basic Experiments

Table 1 shows the average and standard error values of the best-fitness results after 300 fitness evaluations following the change, i.e. after the $650th$, $1350th$ and $2850th$ evaluations for periods 3, 10 and 25 respectively. In the table, *Period* shows the change period; *Change* shows the fitness evaluation at which the change occurs; Avg is the average of the best-fitness values over 1000 runs; $StdErr$ is the standard error of the best-fitness values. It should be noted that for M4, the change takes effect 50 evaluations after the actual change in the environment occurs. This is due to the fact that M4 ignores the change until the end of the current generation. As a results of this, M4 actually has 250 fitness evaluations using the new fitness function to recover after it recognizes the change. Table 2 gives the value intervals of the best-fitness results calculated as $[Avg \pm StdErr]$. In addition to best-fitness tables, the corresponding offline-error plots are also given in Figure 1, Figure 2 and Figure 3 for the different change periods. In these plots, the x-axis show the number of evaluations and the y-axis show the offline error values. It should be noted that for M4, the offline-error is calculated based on the new fitness function representing the new environment while the algorithm still uses the old fitness function for evaluations until the end of the current generation.

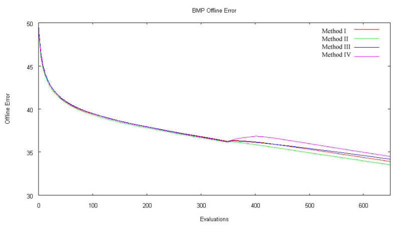

Figure 1. Offline error for period 3

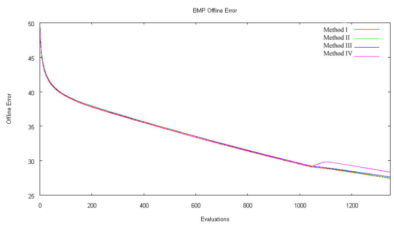

Figure 2. Offline error for period 10

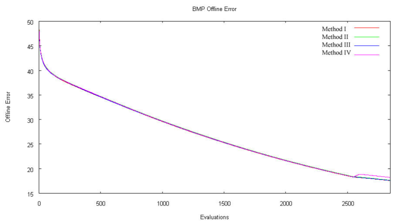

Figure 3. Offline error for period 25

Table 1. Best fitness values for the dBMP in the base experiments

	M1			M2			M3			M4		
Period	3	10	25	3	10	25	3	10	25	3	10	25
Avg	72.24	81.40	95.27	72.60	80.92	89.96	71.40	80.92	95.15	72.04	80.54	89.71
StdErr	0.07	0.06	0.04	0.07	0.06	0.04	0.06	0.06	0.04	0.06	0.06	0.04

From Table 1 and Table 2, we can see the ordering of the methods from best to worst as shown below. The results in the offline error plots also confirm this

Table 2. Best fitness value intervals for the dBMP

	Best Fitness Intervals		
	Period 3	Period 10	Period 25
M1	[72.17, 72.31]	[81.34, 81.46]	[95.23, 95.31]
M2	[72.53, 72.67]	[80.86, 80.98]	[89.92, 90.00]
M3	[71.34, 71.46]	[80.86, 80.98]	[95.11, 95.19]
M4	[71.98, 72.10]	[80.48, 80.60]	[89.67, 89.75]

ordering. As can be seen from Table 2, the best-fitness intervals do not overlap, making the differences significant.

- for period 3: M2, M1, M4, M3
- for period 10: M1, (M2 and M3), M4
- for period 25: M1, M3, M2, M4

From these results, it can be seen that M1 performs unexpectedly well. In M1, when a change occurs, individuals evaluated with the old fitness function compete with individuals evaluated using the new fitness function in the selection stage, however, in the basic experiments this does not seem to be harmful. This could be due to the fact that the change is not very severe and the individuals preserve their relative fitness orderings in the new environment. To see if this is the case, further experiments will be performed in subsection 4.3.

Performance of M2 decreases with increasing change periods. This may be due to the fact that the diversity of the reduced population right after the change is not sufficient for the population to recover if the population has a longer time to converge before the change. The diversity issues will be further tested in subsection 4.3.

Performance of M3 is relatively better in longer periods of change. In M3, when change occurs, individuals which have been evaluated using the old fitness function are re-evaluated. This causes this method to have less fitness evaluations available to search in the new environment than the other three methods. This is the case in all change period cases, however when the population has a longer time to find good solutions before the change, if the change is not too severe, the fewer number of fitness evaluations is not as detrimental as in shorter periods.

M4 performs bad overall. This is due to the fact that the change is ignored until the end of the generation and the population continues to converge around the old peak while the other methods use this time to search for the new optimum in the new environment.

4.3 Supporting Experiments

In this subsection, further experiments to analyze and support the results of the basic experiments are performed.

Test 1 The aim of this test is to determine the similarity of the environments before and after the change. The selection mechanism in the EA uses pairwise *better* relations. If the relative orderings of the individuals in the population remain the same in the two environments, then we can say that from the point of view of the selection operator, the two environments are similar. The test is performed through the following steps on the generation during which the change occurs:

1. sort all individuals in the population in increasing order based on their fitness values, where the first half of the population has been evaluated using the old fitness function while the second half has been evaluated using the new fitness function
2. re-evaluate the first half of the population using the new fitness function and sort all individuals in increasing order of their fitness values
3. pairwise *better* relationships between all possible pairs of individuals in both sorted lists are compared. The number of differences*** shows the number of times the *better* relationship between two individuals has been changed in the second environment.

For the different change period cases, the following averaged results for the differences in the pairwise *better* relationships have been obtained.

- for period 3: 715.7 differences out of 4990
- for period 10: 1067 differences out of 4990
- for period 25: 1560.5 differences out of 4990

As can be seen, the difference in ranking before and after the change increase with increasing change periods. This is probably because the individuals become more and more similar over time, and a change in the environment then has a larger effect on the ranking of the individuals. However the changes never get very high since the change severity is quite low. This confirms the comments made in subsection 4.2 about the relative success of M1 for all change periods.

Test 2 The aim of this test is to explain why the performance of M2 decreases with increasing change periods. To support the possible cause explained in subsection 4.2 about M2, the average hamming distance of the half population before the change is calculated for each change period case and the following results are obtained: 23.3, 18.1 and 8.3 for periods 3, 10 and 25 respectively. Results show that the diversity maintained in the half population decreases with increasing change periods.

4.4 Further Experiments

In the previous subsections, the experiments are based on only one change which is not severe. In this subsection we look at three additional issues to analyze the

*** Worst case is $n * (n-1)/2$ where n is the number of individuals in the population.

behaviour of the four methods, i.e. at the effect of different offset values on M2, at a more severe change and finally at consecutively occurring multiple environmental changes.

Additional Experiments on M2 In the basic experiments, the offset value is taken as 50, which means that the change occurs mid-generation. Now we will use offset values of 10 and 25 in addition to 50 to see how the performance of M2 is affected. The results are given in Table 3 where *diversity* shows the diversity of the population as calculated in Test 2 in subsection 4.3. As can be seen from these results, the offset value does not have a very significant effect on the performance of M2 for short change periods. However, it can be said that as the change period increases, M2 performs better for larger offset values.

Table 3. Results for M2 under different offset values for the dBMP

Offsets	10			25			50		
Periods	3	10	25	3	10	25	3	10	25
Avg	72.48	80.67	89.82	72.51	80.96	89.83	72.60	80.92	89.96
StdErr	0.07	0.06	0.04	0.07	0.06	0.04	0.07	0.06	0.04
Diversity	21.23	16.66	7.62	22.83	17.66	8.00	23.31	18.06	8.28

Additional Experiments on Change Severity In the basic experiments, the change severity was chosen to be low. In these additional experiments, we increase the change severity from 0.1 to 0.4 which means that for each change, approximately 40 bits, which are randomly selected, are flipped in the target string. The best-fitness results can be seen in Table 4. These results are from 100 runs of the methods, hence the larger standard error values.

Table 4. Best fitness values for the dBMP in a more severely changing environment

	M1			M2			M3			M4		
Period	3	10	25	3	10	25	3	10	25	3	10	25
Avg	68.80	79.70	95.14	66.41	67.35	65.76	68.08	79.86	95.17	66.71	66.30	65.74
StdErr	0.16	0.21	0.12	0.21	0.20	0.14	0.17	0.21	0.16	0.19	0.18	0.20

From Table 4, we can see the ordering of the methods from best to worst as:

- for period 3: M1, M3, M4, M2
- for period 10: M3, M1, M2, M4
- for period 25: M3, M1, M2, M4

We can see from these results that the performance of M2 is affected the most from the increase in the change severity. Another observation is that, in a severe change case M3 outperforms M1. This is expected since in a more severe change case, the pairwise *better* relationships cannot be maintained which causes errors in the selection stage of M1. Thus M3, which re-evaluates the individuals using the new fitness function, becomes the best performer though M1 still displays a reasonable performance. M4 is again the worst performer.

Additional Experiments with Multiple Changes In the basic experiments, for simplicity of the analysis, we looked at the effect of only one change in the environment. In these additional experiments, we look at the cumulative effect of multiple changes occurring in the environment. In our experiments here, we will apply 20 changes to the environment (change severity is 0.1). For each tested case, to remove the variance due to the environmental changes, the same set of 20 new environments is used. The best-fitness results obtained after 300 fitness evaluations following the last environment change can be seen in Table 5. These results are again from 100 runs of the methods.

Table 5. Best fitness values for the dBMP for 20 consecutive changes

	M1			M2			M3			M4		
Period	3	10	25	3	10	25	3	10	25	3	10	25
Avg	77.08	88.59	98.26	76.83	85.06	90.67	74.81	89.14	98.59	76.44	84.62	90.30
StdErr	0.18	0.16	0.08	0.23	0.16	0.08	0.21	0.15	0.08	0.21	0.16	0.08

From Table 5, we can see the ordering of the methods from best to worst as shown below.

- for period 3: M1, M2, M4, M3
- for period 10: M3, M1, M2, M4
- for period 25: M3, M1, M2, M4

We can see from these results that the performance of M1 is still acceptable in all cases. M3 performs the worst when the environment changes very quickly. However, M3 is the best performer for slower changing environments. As in the previous experiments M4 is the worst performer and M2's performance decreases with increasing change periods.

5 Conclusion and Future Work

Through the experiments in this study, we empirically compared different approaches to handle changes which occur within a generation, i.e. before the current generation ends. The experiments consist of three parts: In the first part, the strategies are compared on a base test case where the environment change is not severe. In the second part, further tests are performed to support the explanations in the first part. The third part extends the base test cases to include severer environmental changes, as well as consecutive multiple changes and different offset values for change periods.

As a conclusion of all the experiments the following can be said:

- M4 is the worst performer overall. In the current literature, changes are usually assumed to occur between generations. In the cases where the actual change happens while the generation is not ended, this corresponds to M4. However, as we have seen from the results in this study, this is not a recommended option.
- M1 performs best in the cases where the change period is small, i.e. in quickly changing environments. Even though there is a slight performance drop in M1 for slower changes, it is still acceptable. Since the main focus of this paper is on quickly changing environments, M1 is the recommended option.
- When the change periods are longer, M3 becomes the best performer. However, M3 is not a feasible option in quickly changing environments due to the fitness evaluations lost re-evaluating the first part of the population, especially when these evaluations are costly.
- M2 doesn't have a consistent performance. It is greatly affected by change severity and change period lengths. Also in most cases it is among the bad performing strategies. Thus M2 is also not a recommended option.

So overall, it can be said that in a quickly changing environment, with changes within generations and where changes are low to moderate in severity, accepting the change when it occurs, keeping the individuals already evaluated using the old fitness function and continuing the generation with the new fitness function seems to be the best choice.

The dynamic bit matching problem is a very simple test function so to have a better understanding of the performance of the different strategies, further tests should be performed using other problems. Also, the same analysis can be performed for a steady-state reproduction scheme.

References

1. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.
2. J. Branke and W. Wang. Theoretical analysis of simple evolution strategies in quickly changing enviroenments. In E. Cantu-Paz, editor, *Proceedings of Genetic and Evolutionary Computation Conference*, volume 2723 of *Lecture Notes in Computer Science*, pages 537–548. Springer, 2003.
3. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
4. R. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, 2004.

5. F. Vavak and T. C. Fogarty. A comparative study of steady state and generational genetic algorithms for use in nonstationary environments. In T. C. Fogarty, editor, *Proceedings of AISB Workshop on Evolutionary Computing*, volume 1143 of *Lecture Notes in Computer Science*, pages 297–304. Springer, 1996.