Experimental Analysis of Binary Differential Evolution in Dynamic Environments

Alp Emre Kanlikilicer
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
kanlikilic@itu.edu.tr

Ali Keles
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
kelesal@itu.edu.tr

A. Sima Uyar
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
etaner@itu.edu.tr

ABSTRACT

Many real-world optimization problems are dynamic in nature. The interest in the Evolutionary Algorithms (EAs) community in applying EA variants to dynamic optimization problems has increased greatly. Differential Evolution (DE) belongs to the group of evolutionary algorithms which operate in continuous search spaces. DE has been successfully applied to many stationary problem domains. Recently there has been some research into applying DE to dynamic optimization problems too. Many real-world problems consist of decision variables which require the optimization algorithm to work with binary parameters. This makes it impossible to apply DE in its basic form. For this purpose, binary differential evolution (BDE) approaches have been introduced. The main focus of this paper is to perform a series of experiments to test the behavior of a simple BDE under different change conditions. A simple bit-matching problem is chosen as the test environment. The results of this preliminary study show that further study is needed to make BDEs suitable to work in dynamic environments.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*Heuristic Methods*

General Terms

Algorithms, Performance

Keywords

Differential Evolution, Binary Differential Evolution, Dynamic Optimization Problems, Dynamic Bit Matching

1. INTRODUCTION

Many real-world optimization problems are dynamic in nature. Evolutionary algorithms (EAs) are based on mechanisms found in nature. Adaptation to changes is an ongoing process in nature. This has caused a growing interest in applying various types of evolutionary algorithms (EAs) [2] to such problems. For detailed overviews on EAs in dynamic environments, refer to [6, 7, 9, 8].

The Differential Evolution (DE) [1] algorithm, introduced by Storn and Price in 1995, belongs to the group of evolutionary algorithms which operate in continuous search spaces. DE has been successfully applied to many stationary problem domains. Recently there has been some research into applying DE to dynamic optimization problems too. In [3], DynDE, a multi-population DE approach for dynamic environments is introduced. The test results are promising.

Many real-world problems consist of decision variables which require the optimization algorithm to work with binary parameters. This makes it impossible to apply DE in its basic form to such problems. For this purpose binary differential evolution (BDE) approaches have been introduced. Two such different implementations can be found in [4] and in [5]. The successful results reported in [3] for applying DE to dynamic optimization problems has been the main motivation behind this study.

The main focus of this paper is to perform a series of experiments to test the behavior of BDE under different change conditions. A simple bit-matching problem is chosen as the test environment and several experiments are performed using the BDE implementation reported in [5]. The BDE is used in its pure form with no additional measures to adapt it to perform better in dynamic environments. This is a very preliminary study and the results promote further study.

The rest of the paper is structured as follows: Section 2 gives an overview of issues related to dynamic environments. In Section 3, the BDE algorithm used in this study is explained. Section 4 details the experimental design and provides results and discussions. Section 5 concludes the paper and provides directions for future research.

2. DYNAMIC ENVIRONMENTS

Dynamic optimization problems present a challenging area for EA research. An optimization can be considered dynamic if the change in the environment occurs

- in the objective function,
- in the constraints,
- in the problem instance.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

One possible way to deal with changes is to approach the problem as new and start from scratch. However, the information gathered until the change time could be useful in finding the new optimum especially if it is not too far from the old one.

Different types of change exhibit different characteristics, therefore the best EA approach for a specific environment has to exploit these characteristics. Thus different types of EAs may perform differently in environments with different dynamics. A new approach proposed for use in dynamic environments should be tested for these different change characteristics. The main approaches for coping with different types of changing environments are summarized and discussed in detail in [6]. Changing environments can be categorized based on the following criteria [6]:

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

To analyze and compare the performance of various approaches several metrics have been proposed. One of the commonly used performance comparison approaches is using best-of-generation plots. In these plots, the fitness of the best individual in each generation is plotted. This gives an idea as to how badly the algorithm is affected from the change and how quickly and how well it recovers. Another commonly used metric, which gives an overall idea as to how well the algorithm copes with the changes is the offline performance [6]. In this metric, a cumulative average of the best fitness values found so far is calculated. Detailed analysis of different performance measures can be found in [10].

There are many different EA approaches proposed for improved performance in dynamic environments. A current overview can be found in [7].

3. BINARY DIFFERENTIAL EVOLUTION

The Differential Evolution (DE) [1] algorithm was introduced by Storn and Price in 1995. Basically, DE is a population based algorithm which operates in continuous search spaces. DE is based on four main steps:

- initialization
- mutation
- recombination
- selection

Each individual in the population passes through these operations. While the initialization step is only done in the first iteration, the other three steps take place in each iteration of DE. The chromosomes of an individual are made up of real valued genes, shown as in Eq. 1, each of which correspond to the parameters of the problem to be optimized.

$$X_{j,i,g} = \left\{ \begin{array}{ll} j & \text{index of parameter} \\ i & \text{index of individual} \\ g & \text{index of generation} \end{array} \right\} \quad (1)$$

The algorithmic flow of a DE can be seen in Fig. 1. In this study, BDE which works within a binary search space

```
randomly initialize population
while not finished do
  for all individuals in population do
    choose target and base vectors
    randomly choose two other vectors
    computed weighted difference vector of the two
    add difference to base vector
    select between trial vector and base vector
  end for
end while
```

Figure 1: The Differential Evolution Algorithm

is used. As can be seen from the definition of the mutation operator of the DE algorithm given in Eq. 3, it is not possible to use it for binary valued problems without a modification. There are some approaches in literature for modifying DE so that it works with binary problems. One of these methods uses an *angle modulation technique* to transform the binary space into a continuous space [4]. In the initial testing stage of this study, the experimental results obtained using this technique for a dynamic environment turned out to be very insufficient. So in the rest of the study and in this paper, the approach explained in [5] by Gong et. al. is used as the BDE implementation. The details of this algorithm is given below.

The initialization step sets the initial values of the parameters in the population. The value can be either 1 or 0 as seen in Eq 2.

$$X_{j,i,0} = rand\{0, 1\} \quad (2)$$

Each individual in the population, called the *target vector*, goes through the mutation and recombination steps. There are several mutation operators. One of the most commonly used forms of these operators, the DE/rand/1 method, chooses three different vectors from the population and creates a mutant vector from these, which will be called the *donor vector*, through the equation given in Eq. 3.

$$Vi, g = X_{r0,g} + F(X_{r1,g} - X_{r2,g}) \quad (3)$$

F takes values in the range (0,1+) and it is recommended to set F less than 1 [1]. Also, the constraint over deciding the r_0, r_1, r_2 vectors can be changed [1]. For example, in some functions when $r_0 = r_1$ is allowed, better optimized results can be obtained.

The modification for DE to make it run within binary space, is done to the mutation operator. According to the approach defined by Gong et al. [5], the multiplication, addition and substitution operators are changed as explained below:

1. The value of any parameter in any of the vectors can be either 0 or 1. To keep all values within the set of {0,1}, the result of the subtraction ($X_{r1,g,i} - X_{r2,g,i}$) is obtained using the hamming distance [5] between the two vectors. The result of the addition operator is also calculated using hamming distances.

2. After the substitution step, each parameter in the vector is multiplied with the F parameter. This operation forces the values of the parameters to change from binary space to continuous space. To be able to use this

newly created vector, in the next step of the mutation operator, which is the addition operator, the values should be transformed into either 0 or 1 [5]. To implement this, a rounding mechanism is used.

A sample application of the substitution operator is given in Table 3.

Table 1: Sample application of the substitution operator

$X_{r2,g}$	1	1	0	1	0
$X_{r3,g}$	0	1	0	0	0
After substitution	1	0	0	1	0

The aim of the recombination operation is to create a different vector based on the donor and the target vectors. The parameters of this vector are taken from the target vector when a uniformly distributed random number is greater than a predefined Cr value; otherwise, it is taken from the donor vector [1] as shown in Eq. 4. There are two proposed ways [5] to implement this step: binomial and exponential. The binomial crossover operation considers each parameter in a vector separately, however in the exponential crossover operation after $rand_j(0,1) \leq Cr$ is true for the first time, the remaining parameters are taken from the *donor vector* as a block [1]. According to the experimental results obtained by Gong et. al. [5], binomial crossover operator shows better performance for the *OneMax* problem. Because the problem used for the tests in this paper is similar to the *OneMax* problem, the binomial crossover operator is chosen in this study. Cr takes values in the range [0,1]. The vector that is created through the recombination step is called the *trial vector*.

$$U_{j,i,g} = \begin{cases} V_{j,i,g} & \text{if}(rand_j(0,1) \leq Cr \text{ or } j = j_{rand}) \\ X_{j,i,g} & \text{otherwise} \end{cases} \quad (4)$$

Selection is the step to choose the vector between the target vector and the trial vector with the aim of creating an individual for the next generation as shown in Eq. 5.

$$X_{i,g+1} = \left\{ \begin{array}{ll} X_{i,g} & \text{if } f(U_{i,g}) \leq f(X_{i,g}) \\ U_{i,g} & \text{otherwise} \end{array} \right\} \quad (5)$$

These steps continue until an acceptable solution is found or a predefined number of maximum iterations has been reached.

4. EXPERIMENTS

The main aim of the experiments is to analyze the performance of BDE in dynamic environments under different change characteristics. The experiments consist of four tests.

- Test 1 compares the performance of a simple genetic algorithm (SGA) to that of BDE in a stationary environment
- Test 2 compares the performance of a simple genetic algorithm (SGA) to that of BDE in a base case dynamic environment
- Test 3 analyzes the performance of BDE under changes of different severities
- Test 4 analyzes the performance of BDE under changes occurring with different frequencies

4.1 Test Problem

A simple, unimodal test problem is chosen for the tests in this preliminary study. In the bit-matching problem, the aim of the EA is to find the string which fully matches a given string called the *target string*. The fitness of a solution is defined as the number of bits in the solution candidate which match the corresponding bits in the target string. A higher match means a higher fitness. In the binary case, the fitness of an individual i can be calculated using the Hamming distance between the target string and the solution candidate as given in Eq. 6.

$$f_i = Length - Hd(target, string_i) \quad (6)$$

where $Hd(target, string_i)$ gives the Hamming distance between the $target$ string and the solution candidate $string_i$ given by the chromosome of individual i and $Length$ is the length of the target string.

The optimum solution matches the target string in all positions, and thus the hamming distance is 0. Therefore there is only one global optimum for this problem and it is also unimodal. In the dynamic version of the bit-matching problem (DBM), the target string is changed everytime the environment changes. The number of bits that are changed in the target string is determined by the severity of the change. A higher number of bits are changed when the change in the environment is severe.

4.2 Parameter Settings

For the BDE and the DBM in all experiments, a base-case for the parameter settings is defined as given below. Base-case parameter settings for the BDE:

- F and CR are taken as 0.6 and 0.3 respectively. These values are determined experimentally.
- A chromosome consists of 100 bits.
- The population consists of 100 individuals
- DE/rand/1 mutation is used.
- Binomial crossover method is used.

Base-case parameter settings for the DBM problem

- Target string length is 100.
- At every change approximately 20% are changed, i.e. for every bit position on the target string, the probability of being changed is 0.2.
- Changes occur every 3000 fitness evaluations.
- A total of 20 changes are applied for every run.

For Test 1 and Test 2, the BDE will be compared to a SGA. As a SGA, a very standard generational GA with default parameter settings [2] is used as given below. The algorithmic flow of the SGA can be seen in Fig. 2.

- Population consists of 100 individuals.
- Chromosome length is 100.

1: randomly initialize population
2: **while** *not finished* **do**
3: select mating pool
4: perform crossover on parents in pool
5: perform mutation on the offspring
6: offspring replace parents
7: **end while**

Figure 2: The Simple Genetic Algorithm

- Binary tournament selection is used.
- Recombination is performed through two-point crossover with probability $p_c = 0.8$. This type of crossover is chosen to make it similar to the crossover mechanism in BDE.
- Bit-flip mutation with probability $p_m = 1/Length$, i.e.$p_m = 0.01$, is used.
- The next generation of individuals is determined through generational replacement where the offspring fully replace the parent population with no elitism.

In Test 3, BDE will be analyzed under changes of different severities. The following severity settings are used:

- $lowSeverity = 0.05$
- $mediumSeverity = 0.2$
- $highSeverity = 0.4$
- $veryHighSeverity = 0.75$

In Test 4, BDE will be analyzed under changes occurring with different frequencies. The following frequency settings are used:

- $highFrequency$: changes occur every 1000 fitness evaluations
- $mediumFrequency$: changes occur every 3000 fitness evaluations
- $lowFrequency$: changes occur every 5000 fitness evaluations

The performance comparisons are done based on offline performance values calculated as given below.

$$\begin{aligned} &x' = \tfrac{1}{T}\sum_{t=1}^{T} e'_t \\ &where \\ &e'_t = max[e_\tau, e_{\tau+1}, ..., e_t] \end{aligned} \tag{7}$$

where τ is the last time step ($\tau < t$) at which a change occurred.

All results are averaged over 20 runs with different random seeds and different random initial target strings. The average offline performance values over 20 runs and the corresponding standard error values will be reported for each test.

4.3 Test Results

All results will be given as offline performace plots and also as average and standard error value tables for each test separately. In all the plots, the x-axes show the number of fitness evaluations and the y-axes show the corresponding offline performance values averaged over 20 runs.

In Test 1, the main aim is to compare the performace of BDE to a SGA in a stationary environment as a base case. BDE uses the base settings and both algorithms are allowed to run a total of 63000 fitness evaluations which corresponds to the total number of fitness evaluations for 21 environments with 3000 fitness evaluations in each. This value is chosen because in the base-case settings for DBM, 20 changes occur with a period of every 3000 fitness evaluations so each algorithm has a chance to work in 21 environments. The offline performance plot is given in Fig. 3. The average offline performance and the standard errors for SGA and BDE at the end of 63000 fitness evaluations is given in Table 2.

Both SGA and BDE show a very similar performance in the stationary case. Since the bit matching problem is unimodal and easy, the algorithms display a good performance. BDE and SGA perform equally well under the same conditions.

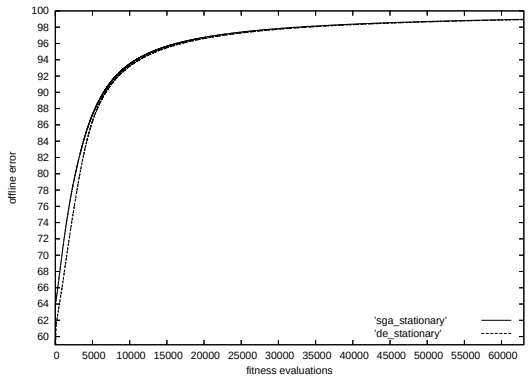

Figure 3: Offline performance comparison of a SGA and a BDE in a stationary environment

Table 2: Results for Test 1

	Average	Standard Error
SGA	98.97	0.02
BDE	98.92	0.008

In Test 2, the main aim is to compare the performace of BDE to a SGA in a base-case setting of a dynamic environment. BDE uses the base settings and both algorithms are subject to the base-change parameter settings of DBM. In the base-case settings for DBM, 20 changes occur with a period of every 3000 fitness evaluations so each algorithm has a chance to work in 21 environments. The offline performance plot is given in Fig. 4. The average offline performance and the standard errors for SGA and BDE at the end of 63000 fitness evaluations is given in Table 3.

In this test, the change severity and frequency is chosen as medium. SGA drops a little in performance. However the drop in BDE is drastic. Even a SGA with no special

mechanisms for dynamic environments performs better than the BDE.

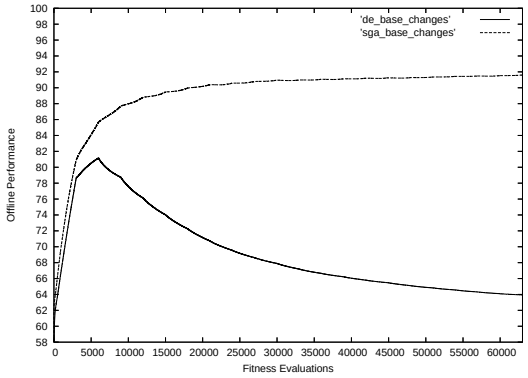

Figure 4: Offline performance comparison of a SGA and a BDE in base-case change scenario

Table 3: Results for Test 2

	Average	Standard Error
SGA	91.6	0.07
BDE	63.95	0.32

In Test 3, the main aim is to analyze the performace of BDE under changes of different severity. BDE uses the base settings and is again subject to the base-change parameter settings of DBM for a total of 21 environments except for the change severity parameter. The tested change severity settings are given in the previous subsection. The offline performance plot is given in Fig. 5. The average offline performance and the standard errors at the end of 63000 fitness evaluations for different severity settings is given in Table 4.

As is expected, performance drops as the severity of changes increases. But the performance drop between a medium severity change and a high severity change is not very large. A very high severity change causes approximately 75% of the bits to change, i.e. change probability for each gene is 0.75. A change probability of 0.5 for a gene means it is a new problem to be solved. In this case, the best approach would be to use *random-restart*. The population which has converged around the current optimum, mainly hinders the progress of the EA after such a severe change. Thus, to study the performance of any EA for any change severities higher than 50% is not very meaningful. For a BDE, this threshold seems to be lower. The offline performance difference between a high and very high severity change is quite small.

Table 4: Results for Test 3

	Average	Standard Error
Low severity	73.97	0.46
Medium severity	63.95	0.32
High severity	61.70	0.19
Very high severity	60.57	0.15

In Test 4, the main aim is to analyze the performance of BDE under changes occurring with different frequencies.

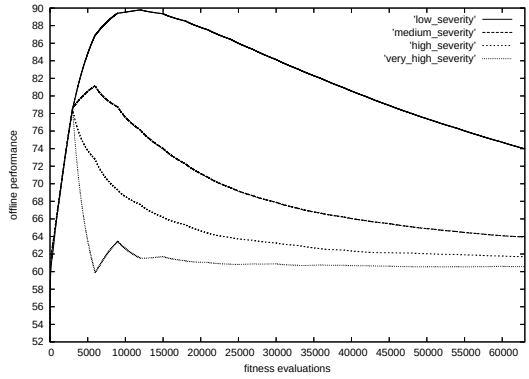

Figure 5: Offline performance in a dynamic environment with different change severities

BDE uses the base settings and is again subject to the base-change parameter settings of DBM for a total of 21 environments except for the change frequency parameter. The tested change frequency settings are given in the previous subsection. The average offline performance and the standard errors for the different frequency settings is given in Table 5. Note that in this test case, since changes occur with different number of fitness evaluations in between, at the end of 21 environments, the total number of fitness evaluations are different for each case. Due to this, the offline performance plots for each change frequency is given separately as in Fig. 6, Fig. 7 and Fig. 8.

The worst performance occurs under low frequency changes. This means that the BDE has time to converge between the changes and once it is converged, it looses its diversity and is not able to adapt to the new environment. Similarly, the best performance is achieved in a high frequency change environment. The BDE doesn't have sufficient time to converge and thus is able to track the optimum better.

Table 5: Results for Test 4

	Average	Standard Error
Low frequency	58.52	0.35
Medium frequency	63.95	0.32
High frequency	65.30	0.19

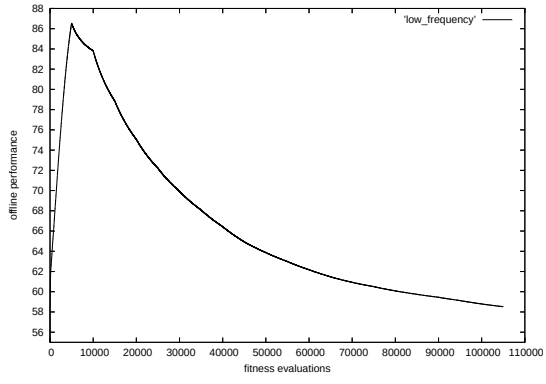

Figure 6: Offline performance in a dynamic environment with low frequency changes

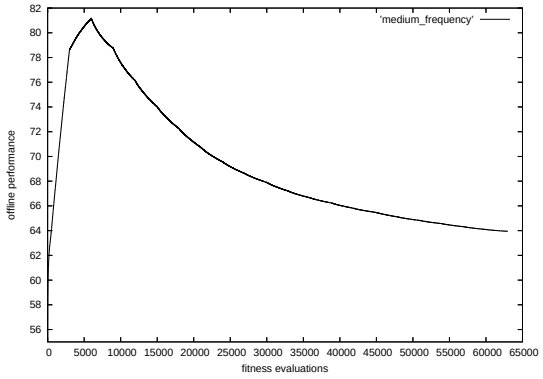

Figure 7: Offline performance in a dynamic environment with medium frequency changes

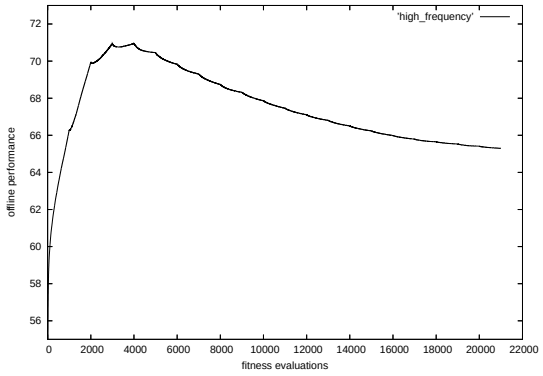

Figure 8: Offline performance in a dynamic environment with high frequency changes

5. CONCLUSION AND FUTURE WORK

In this study, a BDE implementation recently proposed in literature has been tested in a dynamic environment setting with different change characteristics. It has been shown that in the stationary case, a SGA and the BDE perform similarly. However, even in the base case change, BDE performs poorly when compared to the SGA. In the further tests, BDE performs even worse. The results indicate that loss of diversity is a major issue when applying BDE to dynamic optimization problems. Once the population looses diversity it is very hard to adapt to the new environment. It can especially be seen in the results of the tests with different frequencies.

It can be concluded from the results that BDE in its pure form is not suitable to be used for solving dynamic optimization problems. However, since it performs well in stationary environments, it promotes further study. Incorporating diversity maintaining techniques into BDE, would help improve performance. Also, multi-population approaches similar to DynDE [3] should be experimented with. In [3], a good performance is reported for a DE in a dynamic environment with real valued parameters. Similar enhancements can make BDE perform better too.

In conclusion, the results of this preliminary study show that further study is needed to make BDEs suitable to work well in dynamic environments.

6. REFERENCES

[1] Keneth V. Price and Rainer M.Storn and Jouni A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization. Springer,2005.

[2] A. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. Springer,2003.

[3] R. Mendes and A.S. Mohais. DynDE: A Differential Evolution for Dynamic Optimization Problems. IEEE Congress on Evolutionary Computation (CEC 2005), pp. 2808-2815, 2005.

[4] G. Pampara and A.P. Engelbrecht and N. Franken. Binary Differential Evolution IEEE Congress on Evolutionary Computation (CEC 2006), pp. 1873-1879, 2006.

[5] Tao Gong and Andrew Tuson. Differential Evolution for Binary Encoding. 11th Online World Conference on Soft Computing in Industrial Applications (WSC 11), 2006.

[6] J. Branke. Evolutionary Optimization in Dynamic Environments. Kluwer, 2003.

[7] Y. Jin and J. Branke. Evolutionary Optimization in Uncertain Environments – A Survey. IEEE Transactions on Evolutionary Computation, Vol. 9, No. 3, pp. 303-317, 2005.

[8] K. Weicker. Evolutionary Algorithms and Dynamic Optimization Problems. Der Andere Verlag, 2003.

[9] R. Morrison. Designing Evolutionary Algorithms for Dynamic Environments. Springer,2004.

[10] K. Weicker. Performance Measures for Dynamic Environments. Parallel Problem Solving from Nature (PPSN VII), 2002.