İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

EVOLUTIONARY ALGORITHMS IN DYNAMIC ENVIRONMENTS: MANAGING CHANGES WITHIN GENERATIONS

M.Sc. Thesis by

Gulshat KULZHABAYEVA

Department : COMPUTER ENGINEERING

Programme: COMPUTER ENGINEERING

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

EVOLUTIONARY ALGORITHMS IN DYNAMIC ENVIRONMENTS: MANAGING CHANGES WITHIN GENERATIONS

M.Sc. Thesis by

Gulshat KULZHABAYEVA

Department : Computer Engineering

Programme: Computer Engineering

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

EVOLUTIONARY ALGORITHMS IN DYNAMIC ENVIRONMENTS: MANAGING CHANGES WITHIN GENERATIONS

M.Sc. Thesis by

Gulshat KULZHABAYEVA

(504021548)

Date of submission : 07 May 2007

Date of defence examination: 13 June 2007

Supervisor (Chairman): Asst. Prof. Dr. Şima UYAR

Members of the Examining Committee Prof.Dr. Emre HARMANCI (İTU.)

Assoc.Prof.Dr. Haluk TOPÇUOĞLU (MU.)

JUNE 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DİNAMİK ORTAMLARDA EVRİMSEL ALGORİTMALAR: NESİLİÇİ DEĞİŞİMİN YÖNETİMİ

YÜKSEK LİSANS TEZİ

Gulshat KULZHABAYEVA
(504021548)

Tezin Enstitüye Verildiği Tarih : 07 Mayıs 2007
Tezin Savunulduğu Tarih : 13 Haziran 2007

Tez Danışmanı : Yrd.Doç.Dr. Şima UYAR

Diğer Jüri Üyeleri Prof.Dr. Emre HARMANCI (İTÜ.)

Doç.Dr. Haluk TOPÇUOĞLU (MÜ.)

HAZİRAN 2007

ACKNOWLEDGEMENTS

I would like to deeply thank Asst. Prof. Dr. Şima Uyar, for refereeing this work. I am grateful to her providing me with useful and helpful advises by sparing a lot of her time and patience. Without her care and consideration, this thesis would likely not have matured.

My thanks to Dr. Juergen Branke, for his guidance during my work with his great experience and vision in this area.

Also, I would like to thank my friends, Murat Mehmet and Zeliha Görmez for their help and efforts to give motivation to accomplish the thesis.

My sincere thanks to my husband, Askhat and my son, AmirAli. This work is the result of their great support, encouragement, help and love. My great thanks to my family for their prayers and encouragement, especially my parents, Nurbala and Baltabay, who got me to and through all my life.

MAY 2007 Gulshat KULZHABAYEVA

CONTENTS

ABBREVIATIONS

CPU	: Central Process Unit
BMP	: Bit-Matching Problem
EA	: Evolutionary Algorithm
GA	: Genetic Algorithm
GR	: Generational Reproduction Method
MKP	: Multidimensional Knapsack Problem
SKP	: Single Knapsack Problem
SSR	: Steady State Reproduction Method
TDGA	: Thermo Dynamic Genetic Algorithm

TABLE LIST

Page

FIGURE LIST

Page

EVOLUTIONARY ALGORITHMS IN DYNAMIC ENVIRONMENTS: MANAGING CHANGES WITHIN GENERATIONS

SUMMARY

Evolutionary algorithms (EAs), based on natural evolutionary theory, are computational methods for static and dynamic problems. Since EAs are naturally inspired, they seem to be more suitable to be applied to dynamic optimization problems. In order to apply EAs to the continuously changing real world problems, changes have to be analyzed in a detailed way. According to the type of change and problem instance, if method of managing changes can be predicted it will improve ability of problem solving of EAs.

In nature changes are not happening in an organized way, although almost all researchers' approach to dynamic changing environment was in that way, thus assuming that changes are happening between generations

The main goal of the thesis is to examine and show the ability of methods on changing environments within generations by empirical way. The methods are:

- Use the changed fitness function for all subsequent individuals, but keep the evaluations of the offspring already evaluated

- Temporarily reduce the population size. The generation is terminated, and the offspring generated so far serve as basis to generate the next.

- Re-evaluate all offspring already generated; ignore the change and continue to work with the old fitness function to the end of that generation

- Ignore the change and continue to work with the old fitness function until all offspring of that generation have been evaluated.

As a result, in real world problems where changes happen at any time, as in design of EAs within generation, knowing the ability of above methods will help us in organizing more effective EAs which are not time consuming as well as resources.

DİNAMİK ORTAMLARDA EVRİMSEL ALGORİTMALAR:
NESİLİÇİ DEĞİŞİMİN YÖNETİMİ

ÖZET

Evrimsel Algoritmalar, evrim teorisinin kavramına dayalı olarak statik ve dinamik problemleri çözmek adına geliştirilmiş hesaplama yöntemleridir. Evrimsel Algoritmaların doğadan esinlenen yapısı itibarı ile günümüzdeki değişken problemlere en çok uyum sağlayan algoritmalardır.

Dinamik problemlere Evrimsel algoritmalar ile çözüm bulabilmek için değişen ortamın ve değişimin çok detaylı analiz edilmesi gerekmektedir. Yani ortam değiştikten sonra değişim çeşidine ve probleme bağlı olarak en uygun yol seçildiğinde Evrimsel Algoritmaların daha etkin bir şekilde çözüm bulunması sağlanabilir. Günümüzdeki gerçek problemlerde var olan değişimin rastgele olmasına rağmen, bu alandaki çalışmaların hemen hepsi değişimlerin sistematik bir şekilde, yani nesiller arası gerçekleştiğini varsayılarak yapılmıştır. Bu varsayım yeterli olmuş olsa da geliştirilmesi ve araştırılması gereken bir konu olarak karşımıza çıkmaktadır.

Bu tezin asıl amacı ortamın nesil içi değiştiğini varsayarak, aşağıdaki yöntemlerden hangisinin daha etkin bir yol olduğunu deneysel sonuçlarla kanıtlamak. Bunlar:

- Ortam değiştikten sonra neslin geri kalanını yeni ortama göre hesaplamak ve yeni bireyleri eski bireylerle değerlendirmek
- Ortam değiştikten sonra neslin yaşamını durdurmak ve bir sonraki nesli eski nesilden türeterek devam etmek
- Ortam değiştikten sonra neslin şimdiye kadar hesaplanan bireylerini tekrar yeni ortamda değerlendirmek
- Ortam değiştikten sonra neslin geri kalan bireylerini eski ortama göre hesaplamaya devam etmek ve değişen ortamı bir sonraki nesle uygulamak

Yukarıda sıralanan yöntemlerin hangisinin daha etkin bir yol olduğunun bilinmesi gerçek problemler üzerinde, yani değişimin neslin ortasında olduğu durumlarda daha etkin çalışan, zamandan ve kaynaktan tasarruf eden uygun Evrimsel Algoritma geliştirmemize yardımcı olacaktır.

1 INTRODUCTION

Since many optimization problems are dynamic and change over time, a suitable optimization algorithm has to be ready to act on these changes by repeatedly adapting the solution to the changed environment. Since Evolutionary Algorithms (EAs) are naturally inspired, they are suitable to be applied to dynamic optimization problems.

In nature, changes do not occur in an organized way, although almost all researchers' approach to dynamic changing environment has been in that 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 EAs' 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. So far previous works on all publications on EAs for dynamic optimization problems assume that the environment changes between generations. Although this assumption is convenient, Branke and Wang 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 [3].

When change happens within generation, the question is whether to reevaluate the individuals generated before the change in the same generation or continue to calculate the fitness of the rest of the individuals according to the new environment. Therefore to have an idea where to direct the search after change has happened we have to examine the different changes also with different approaches within a generation. In order to perform an empirical work on above stated idea, the following proposal was made by Branke in [9]:

- Use the changed fitness function for all subsequent individuals, but keep the evaluations of the offspring already evaluated

- Temporarily reduce the population size. The generation is terminated, and the offspring generated so far serve as basis to generate the next.
- Re-evaluate all offspring already generated; ignore the change and continue to work with the old fitness function to the end of that generation
- Ignore the change and continue to work with the old fitness function until all offspring of that generation have been evaluated.

The main goal of the thesis is to compare the above four methods in changing environments. It is believed to be of help to us in organizing our actions when designing a suitable EA for a problem in changing environments.

The structure of the thesis is as follows:

Chapter 2 gives introductory information about EAs and their main considerations, application areas as well as their mechanisms. Chapter 3 is mainly about dynamic EAs and criteria on their design. Also some approaches used in EAs for dynamic environments are explained briefly. Chapter 4 is about experiments and tools used in the thesis. Chapter 5 gives the conclusions obtained after the experiments in the thesis.

2 BACKGROUND INFORMATION

2.1 History

The idea of Evolutionary Algorithms is borrowed from nature by imitating the natural process of evolution. An ultimate goal of the algorithm is to reach the optimum decision in a complex problem. As is known, during evolution the most adapted individuals survive. This leads to the fact that the fitness of a population increases, allowing it to survive under changing condition.

Since the mid 1960s, a few tools are invented which are inspired by the Darwinian evolution theory such as: evolutionary programming, genetic programming, evolution strategies, and genetic algorithms. All of them can be combined under the term of "Evolutionary Algorithms". On the first look they can be perceived the same algorithm because of their similar names, but indeed these names carry quite distinct meanings to the scientists deeply involved in this area of research. In short, all tools' basic principle parts are the same such as:

- Working on a population of individuals, each of which is a solution to the problem.
- Having an iterative, stochastic search process which is based on the goodness or badness of individuals.
- Undergoing selection, reproduction and replacement, in one generation [8]

2.2 Major Considerations of Using EAs

Successful encoding solutions of a given problem can make Evolutionary Algorithms useful to nearly everyone. Thus an effective EA representation and meaningful fitness evaluation are the keys of the success in EA applications. EA applications are used when traditional ways fail. Failure can be connected with one of the following reasons: rough dependence of optimized criterion on selected parameters; too big number of parameters; impossibility to calculate derivatives on parameters. Thus difference of Eas from traditional methods can be considered as searching from one

population of solutions to another, rather than from individual to individual; using only objective function information, not derivatives; using probabilistic, not deterministic transition rules [13].

Moreover, the approach of the EA can be beneficial in that it can handle arbitrary kinds of constraints and objectives. All constraints and objectives can be handled as weighted components of the fitness evaluation [see Section 2.4.1.3.], making it easy to adapt the EA to the particular requirements of possible problems.

2.3 Application Areas of EAs

EAs have been used for problem-solving and for modelling. Moreover, EAs are applied to many scientific, engineering problems, in business and entertainment, including:

Optimization

Automatic Programming

Machine and robot learning

Economic models

Immune system models

Ecological models

Population genetics models

Interactions between evolution and learning

Models of social systems [10]

2.4 A Brief Overview of EAs' Mechanisms

From biology we know, that any organism can be presented by the phenotype which actually defines the object in the real world, and a genotype which contains the information about an object. Thus each gene, that is an element of the information of a genotype, has the reflection in a phenotype. Thus, for the decision of problems it is necessary to present each attribute of object in the suitable form of genetic algorithm in order to be able to apply evolutionary operators on them and solve problem in a desired way [See Section 2.4.1.1].

All further functioning of mechanisms of EAs is made at a level of a genotype and this causes its wide application in the most different tasks. Each chromosome which is usually called individual presents a solution to a problem. There should be satisfactorily many solutions in a search space in order to find the optimum. For that reason there is a population formed by individuals as in nature. Each individual is weighted with its fitness value which is definitely its adapting measurement according environment. Moreover, some better individuals, selected with respect to their fitness values, after evolutionary operators such as selection, recombination and mutation are applied, can pass into the next generation as an offspring generation. Evolutionary operators and fitness values are explained further.

The scheme of an Evolutionary Algorithm which is used in the thesis is given in Figure 2.1 in a pseudo-code fashion; Figure 2.2 shows a diagram.

```
BEGIN
 INITIALISE population with random candidate solutions;
 REPEAT UNTIL (   TERMINATION CONDITION is satisfied)
        EVALUATE each candidate;
        SELECT parents;
        RECOMBINE pairs  of parents;
        MUTATE the resulting offspring;
        CHECK FOR DUPLICATE the offspring
END.
```

Figure 2.1 : The scheme of an Evolutionary Algorithm in a Pseudo-code Fashion

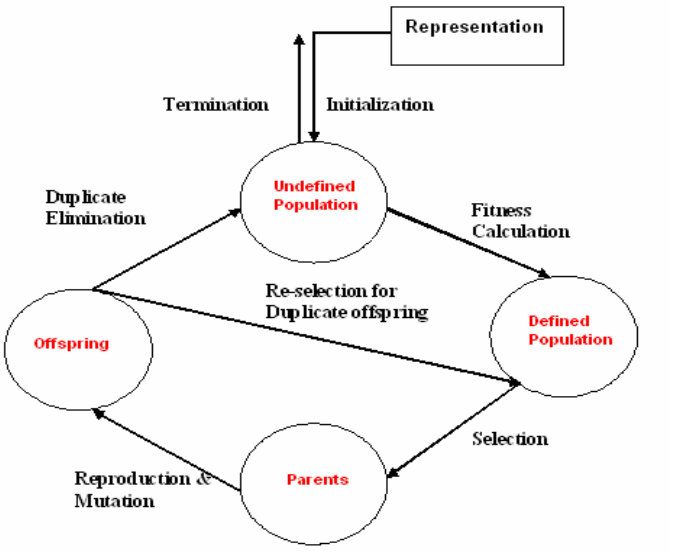

Figure 2.2 : The scheme of an Evolutionary Algorithm in a Diagram

2.4.1 Main Components of EAs

In order to design a particular EA some of the main components should be considered and the most important components are:

- Representation (definition of individuals)
- Evaluation function (or fitness function)
- Reproduction (selection mechanism)
- Variation operators, recombination and mutation
- Reinsertion operators
- Duplicate-elimination operators
- Termination

2.4.1.1 Representation

As mentioned above in order to apply EAs to a problem in an effective way, representation of solutions is important. Chromosomes are basic concepts of EAs and they consist of genes, as in nature. Therefore in order to computerize EAs we have to represent each gene usually identified as an allele.

Although binary representation form is widely used, representation for each problem can vary depending on its requirements when solving real world problems on using EAs.

The most common representations are:

- Binary representation, where allele Є{0, 1}

- Real-valued representation, where allele Є R

- Integer representations, where allele Є Z

2.4.1.2 Initialization

The first population is commonly formed by randomly generated individuals. Here each of the genes in each chromosome is generated randomly according to the representation. For example, assuming representation is binary, an unbiased coin is tossed for each gene. If it turns up heads the gene's value is 0 and if it is tails the value of gene is 1. In this manner, all chromosomes in the first population are generated. However in some cases a Case-Based initialization is used [1].

2.4.1.3 Fitness Evaluation

Evolution is a process of adaptation and as mentioned above the chromosome is a coded decision and there is a value of function of suitability which corresponds to each chromosome thus to each alternative decision. The main goal is to reach the best chromosomes according to their suitability.

Therefore EAs work not with one chromosome, but instead they work with a population of chromosomes. It makes search for an effective decision start at once from several points of a search space. At each iteration of EAs, there is a switching of an old population to a new generation. Thus some chromosomes pass from an old population to the new when others die by leaving the population. Thus, it is provided that, according to the principle of Darwin, the chromosome having a better value of suitability has more chances "to survive", i.e. to pass to next generation.

For example, let us assume that our problem is maximization of adaptation to the environment. So there has to be one chromosome representing the environment and let's have 3 individuals with length of 6 and our representation is also binary representation such as:

Environment chromosome →010101

1st Individual →100100 Fitness Value of 1st Individual is 3

2nd Individual →101000 Fitness Value of 1st Individual is 1

3rd Individual →101001 Fitness Value of 1st Individual is 2

As expected here we have used a bit matching fitness calculation, thus fitness value is the number of genes of individual that matches with the genes of environment chromosome.

Each problem has its own fitness landscape defined by the fitness function over the search space. So the structure of fitness landscape varies from problem to problem.

2.4.1.4 Reproduction

EA is an iterative process in which individuals all over are selected for crossing and then crossed. After crossing, a new generation is formed from the offspring and all begins all over again. Strategy of selection is a main and one of the most important components of EAs and it defines "worthy" individuals for crossing according to their fitness value. Below the most widespread strategies are considered such as [21]:

Rank selection: Each individual in the population is assigned a numerical rank based on fitness, and selection is based on this ranking rather than absolute difference in fitness.

Roulette-wheel selection: Individuals according to their fitness are replaced on a circular diagram and roulette is rotated. The individual from the sector where roulette stops is chosen out for selection. Mathematically, selection probabilities of individuals are proportional equivalent to their fitness value compared to the fitness values of their competitors.

Scaling selection: As the average fitness of the population increases, the strength of the selective pressure also increases and the fitness function becomes more discriminating.

Tournament selection[11]: Selected t individuals from a population containing N individuals, and the best one among t individuals enters the group called mating group in which individuals are used for reproduction. This operation repeats N times. The size of the group of the individuals selected for tournament is often equal to 2. In this case tournament size is defined according to t, selected individuals for tournament. Permutation Selection, used in this thesis, is a kind of a tournament selection where all individuals in the population are paired according to a randomly generated permutation, so it is a pairing of set of individuals where each pair appears exactly once.

2.4.1.5 Variation Operators

Variation operators are necessary to apply principles of heredity and variability to a population used in EAs. Thus described operators are not necessarily applied to all crossed individuals which brings an additional element of uncertainty to the search process for the optimum. In this case, uncertainty does not mean a negative factor, and can be defined as “a degree of freedom" of EAs [20]. There are two types of variation methods such as: recombination and mutation.

2.4.1.5.1 Recombination

The recombination, also named as crossover, is the basic genetic operator making the exchange of genetic material between individuals, called parents in order to reproduce offspring. Recombination is a stochastic operator, thus the choice of what parts of each parent are combined depends on random drawings. Thus the random

number defines a point inside of a chromosome in which both chromosomes (parents) exchange that part of chromosomes. This point is called a crossover point or cut-point. There are many variants of crossover which vary in the number of cut-points such as: One-point, two points and uniform crossover [19]. The situation mentioned above is illustrated in Figure 2.3.

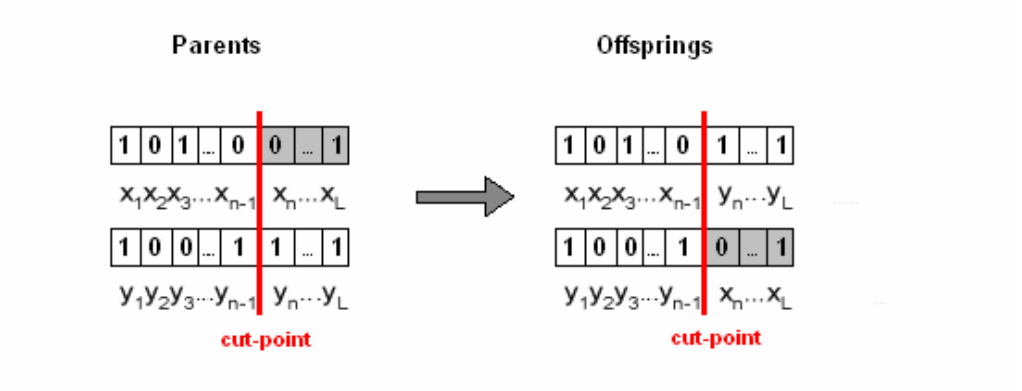

Figure 2.3 : One Point Crossover

2.4.1.5.2 Mutation

The mutation operator is necessary to drive a population away from a local extremum. Moreover, it promotes protection against premature convergence and loss of important notions. Most genetic algorithm research has used mutation as a tool for recovering desirable genes that have been accidentally deleted from population [13]. These are obtained by inverting casually chosen bits in a chromosome, as shown on Figure 1.4. Note that although for simple string encoded EAs, low mutation rates are sufficient, it is known that an efficient way of coping with low coverage is to use higher mutation rates [12].

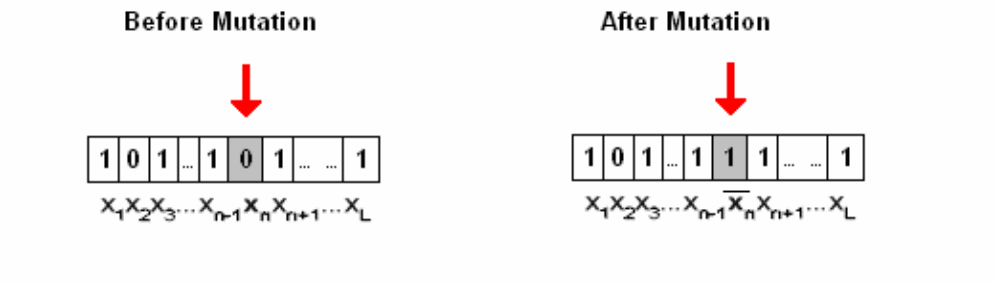

Figure 2.4 : Binary mutation

2.4.1.6 Reinsertion

There are some reinsertion approaches applied by researchers for different purposes. Thus purposes can be compensating size of population between generations. Sometimes algorithms produce more or less offspring from the parents than the population size. Therefore elimination of extra individuals or addition of new ones is needed in order to keep population size constant. In addition to these, some algorithms such as Random Immigrants [16] use this operator for "preserving the diversity" by reinserting new randomly generated individuals into the population. Moreover, elitist principle can be applied by reinsertion method. The essence of this principle is that the best parental individuals join the new generation. Their count can be 1 and more. However carefully designed application of this principle is needed because of the following reason: while it prevents losing the good intermediate solution, the algorithm can get stuck in a local optimum.

2.4.1.7 Duplicate-Elimination

The main goal of this operator is to prevent duplication of individuals. Some algorithms use this operator in order to preserve diversity but it has an additional cost, therefore should be considered well before using. Moreover, continual usage will slow down the algorithm and can be time consuming for real world problems.

2.4.1.8 Termination Condition

The loop of a genetic algorithm proceeds until the pre determined condition of its termination is reached. For example, if the problem has a known optimal fitness level, if one or several individuals' fitness values reach this desired optimum, this can be used as a termination condition. Stochastic iterative characteristic of EAs usually make algorithms work forever because it can not guarantee to reach the optimum level of fitness value. Therefore there are commonly used methods to stop the algorithm such as: the maximum allowed CPU time is elapsed; the total number of fitness evaluations reaches a given limit; for a given period of time the fitness improvement remains under pre-determined value; the population diversity drops under a given threshold [2].

3 DYNAMIC ENVIRONMENTS

Since the EAs are inspired from nature where continuous changes are indispensable, dynamic real world problems are attempted to solve by using suitable dynamic EAs. However, the decision of solving dynamic real world problems requires us to consider many uncertainties. So the sources of uncertainty are:

- Noise
- Robustness
- Fitness Approximation
- Time Varying fitness function

3.1 Noise

Due to the sensory measurement errors or randomized simulations, fitness evaluations are subjected to noise. When conditions change within the environment, fitness of a solution can change abruptly or gradually. If each fitness is evaluated in a finite period of time, the fitness of other solutions can change while one solution is being evaluated. This unpredictable change causes to uncertainty in the current fitness of the solutions [14].

3.2 Robustness

The design variables can change after the optimal solution has been determined. Therefore, despite of slightly change in design variables a common requirement is that a solution should still work satisfactorily. Such solutions are called robust solutions [3]. Robustness and Noise looks like as if the approach of EAs to them is the same, but they have a difference, since noise acts on the fitness function while robustness is due to perturbances in the design variables. To clarify, when noise affects appear it cannot guarantee the same value for the same individual in consecutive evaluations. However, in robustness even though the fitness function is the same, solution can change after optimization.

3.3 Fitness Approximation

Fitness approximations are used usually when the fitness function is too expensive to evaluate or when an analytical fitness function is not available. It is the case when fitness functions generated from collected data or from simulations are used.

3.4 Time Varying Fitness Function

In the general case of an open system, fitness function is not a time invariant function, since it is controlled by an evolutionary mechanism. The fitness function is in this case a measure of the goodness of the response of the system to environment events. As in the real life examples, which is a quiet dynamic environment where everything is in flux. Interest rates change, the weather conditions vary everyday, exchange rates are different each day, etc. So as in EAs given individual may change its fitness as time goes on and the environment changes. Optimal solutions at a given time can become bad solution.

Change in the environment occurs through:

- *change in the objective function*
- *change in the constraints*
- *change in the problem instance*

Above stated matters usually causes optimum to change and forces adaptation of the old solution. There are possible approaches such as

- Treat as a new solution after change and the problem with it is that change may not be detected immediately or new solution may not be too different from the old one so starting as if a new solution may be too time consuming.
- The optimization continuously adapts to the change

Although there are a lot of meta-heuristic search methods which are used with dynamic environments, EAs seem to be a suitable candidate because they have been inspired from natural evolution where there is a continuous adaptation process. However, then the main problem with standard EAs while handling dynamic Environments as an optimizer, are that EAs eventually converge to an optimum and thereby loose their diversity. Keeping diversity in a population is necessary for efficiently exploring the search space and their ability to adapt to a change in the environment when change occurs[7].

As mentioned about changes, ignoring the noise that effects the fitness function, there are different dynamic environments which require different optimization approaches. Therefore all criteria of changes in dynamic environments should be known in order to characterize that one algorithm is better than the other to apply for a problem. In works [4] and [5], dynamic environments are grouped with respect to some criteria given below.

3.5 Criteria for Dynamic Environments

Frequency of change defines the average number of generations (EA time) passed for one environment. Frequent changes make it harder to find the optimum than infrequent changes, because fast adaptation to the environment after a change is more difficult in case of frequent changes. Some algorithms' performance can get better than others as the EA time passes. Therefore, frequency of change will be one of the most important criteria of which algorithm to choose.

Severity of change accounts for the magnitude of changes by comparing the landscape before and after the change. This is also an important criterion in choosing or designing the algorithms. If change severity is low, EAs' first population after change is not so different from the last population before change.

Predictability of change defines if the next change can be predicted. In some dynamic problems, it is quite possible that environment changes follow a recognizable pattern. If this is the case, EA evolves accordingly and will be ready for the next change. Predictability divided roughly into three classes in [6]: 1) highly or completely predictable changes such as translatory and cyclic movements induced by analytic coordinate transformations, 2) completely unpredictable changes such as those depending on realizations of random variables and 3) chaotic changes. Also in above stated research interdependencies between severity, change frequency and predictability of the changes are analyzed and their experiments carried out that the main influence is severity. For more information refer to [6], [7].

Cycle length / accuracy defines the average EA time to encounter a previously seen environment or close to that environment and the similarity between these environments respectively.

3.6 Approaches in EAs

As stated above EAs are more suitable as an optimizer in dynamic environments, therefore it is time to have a look at its approaches in it. These approaches' aim is have better solution. These approaches are

- Restart
- Generate diversity after a change
- Maintain diversity through the run
- Memory based approaches
- Multipopulation approaches

3.6.1 Restart

Population is re-initialized randomly after a change and no information is transferred from the previous instance. This method is not recommended in most cases because it is useless if the new solution is close to the old. Some individuals may be transferred to the new population. The amount of information transferred is important, thus if it is too much may lead to convergence, also too little may slow down the search. So knowledge base of individuals that perform well are kept, indexed with a measure of their environment. When change occurs, population is initialized using individuals that have performed well under similar conditions. In order to perform this kind of task it must be possible to measure environment similarities.

3.6.2 Generate Diversity After a Change

As stated above, one of the problems with standard EAs was losing the diversity while searching for an optimum in the environment. Also we know that in EAs the mutation operator is for generating new different individuals throughout the run. Usually mutation rate is small in order to not spread away from the optimum. Therefore adapting mutation rate explicitly after change can help as to spread out the individuals to find the new optimum. As a result of experiments it is seen that higher mutation rate helps the converged population to spread out and search. However method of adaptation of mutation rate can be grouped in two groups, depending on its application on individuals throughout the run, such as:

- (triggered) Hypermutation as proposed by Cobb [29] whenever change occurs in the environment, mutation rate is increased drastically for some number of generations.
- Variable local search is variant of Hypermutation method, has been suggested by Vavak et al. [30] after change occurs, range of mutation is increased slowly. If population fitness does not improve, the range of the local search is extended by increasing the mutation rate more till the population fitness improves. Experiments done show that this method performs best with very small changes.

3.6.3 Maintain Diversity Throughout The Run

There are several works done on maintaining diversity through the run as stated in [5]:

Random immigrants was introduced by Grefenstette [16] where in every generation population is partly replaced by random new individuals. Thus this preserves diversity in population through the process.

Sharing/crowding an effect of genotypic and phenotypic sharing on the EA's ability to track moving optima was examined by Andersen [18]. This method tries to spread out the population over multiple peaks, it should keep the diversity. Experiments done related to this method concluded that the sharing method remarkably enhances the EA's ability to find optima in slowly changing environments.

Thermodynamical genetic algorithm (TDGA) which was proposed by Mori et al. [31] is to control diversity in the population explicitly through a measure named "free energy". For a minimization problem, this term is calculated as F=<E>-TH where <E> is the average population fitness TH is the measure of diversity in the population. New population selected from the parents and offspring one by one based on trying to minimize F <t. T is a temperature parameter set to change the importance of diversity over time.

As a result of examining overall the studies on maintaining diversity through the run the optimization process results of tests performed show: If change has low severity triggered hypermutation performs better, however in cases of higher severity changes, random immigrants perform better.

3.6.4 Memory-Based approaches

EA is supplied with memory to remember useful information from past generations. It is quite useful when the optimum returns to previous locations. There are two main groups of providing memory: implicit memory and explicit memory approaches. For more information refer to [5].

3.6.5 Multipopulation approaches

In the multipopulation approach, the main idea is dividing the population into sub-populations which are searching for peaks in their own space at the same time. The goal of different subpopulations is maintaining information about promising regions of the search space. There are some example approaches such as:

- self-organizing scouts
- a multinational GA
- shifting balance GA
- sentinels

3.6.5.1 Self Organizing Scouts

The main idea is when x peaks are found, the population is split into x small fractions called the *"scout population"* which watches over the peak and the rest of the population called *the "base population"* spreads out and continues search for new peaks overtime. When a watched peak moves, scout population follow peak by demanding reinforcement from base population. In order to supply the request of reinforcement for scout population when population size is limited, individuals are redistributed to sub-populations where they are most needed. Thus unpromising regions may be abandoned by reporting successful results. For more information refer to [5].

3.6.5.2 Shifting Balance GA

The main aim is to increase exploratory power. Population is divided into a core and a number of small *colony* populations. The core population exploits the best optimum found, and the colony populations are forced to search in other parts of landscape. If a colony gets close to the core population, it is driven away using a distance measure at intervals. It shows good performance only with small changes in the environment.

3.6.5.3 Multinational GA

The main idea is grouping of individuals based on *hill-valley detection* procedure for two points in the search space. Defining borders of the subpopulation requires many extra fitness evaluations to detect valleys. Results reported on two peak environments are shown to be better than sharing method.

3.6.5.4 Sentinels

Sentinels are population members distributed uniformly on the search space where they are treated as regular members used for selection and crossover. They are never replaced when the population converges around a peak and the environment changes, other sentinels are selected for reproduction. Main aim is to have a uniform distribution of individuals on the search space. There are many successful methods existing in literature and successful results are reported.

3.7 Suitable Benchmark Problems

Branke stated in his work [5] that optimization in dynamic environments seems to require two fundamental capabilities:

- Tracking of a solution that changes slightly
- Jumping from an old solution to a quite distant new optimum that appeared elsewhere.

Thus it should be possible to vary many of the environmental variables such as peak heights, peak shapes, peak locations. It should also provide benchmarking for binary and real valued encodings while it should be possible to vary change dynamics, change frequency and change severity, it should be simple to implement, it should be simple to analyze and it should allow conjectures to real world problems. The Moving Peaks Benchmark introduced by Branke in [27,28] tries to provide the above aspects. There are several kinds of commonly used Benchmark problems such as:

- Dynamic Multiple Knapsack Problem
- Dynamic Bit-Matching Problem (will be introduced in [Section 4.1.1])
- Dynamic Simple Knapsack Problem (will be introduced in [Section 4.1.2])

3.8 Measuring Performance

There are some criteria taken for account by many researchers while evaluating the results of algorithms in changing environments, such as intuitive meaning, straightforward methods for statistical significance testing of comparative results, and measurement of performance over a sufficiently large exposure to landscape dynamics

Some of the mostly used performance measures for dynamic environments are online performance, offline performance, offline error and best fitness values.

3.8.1 Online Performance

Online Performance at EA time T is defined as the average fitness of all evaluations over entire run and every evaluation requires testing the real world.

3.8.2 Offline Performance

Offline performance at EA time is defined as the average fitness of all best individuals found so far and optimization is done in a simulated environment and only best solutions are transferred to real world

For non-stationary environments, offline performance should only consider individuals evaluated since the last change. Also offline performance requires that the changes are detected or known.

3.8.3 Offline Error

Offline Error at EA time T is defined as the average of current errors, i.e. the difference between the current optimum and the current best fitness, over the entire run. This performance is applicable only if the researchers know the optima of all environments encountered.

$e_t' = \max \{e_\tau, e_{\tau+1}, ..., e_t\}$ is the last step at which change occurred **(3.1)**

$$\varepsilon^* = \frac{1}{T}\sum_{t=1}^{T} e_t' \quad \textbf{(3.2)}$$

Where ε^* is the offline error performance and T is the number of evaluations considered.

3.8.4 Best Fitness Performance

Best Fitness Performance at EA time is defined as the set of best fitness values found in each environment encountered up to that time.

3.9 Theories on EAs

Most of the work done on dynamic environments was of practical nature, however in the recent time, researchers try to look at the problem from a theoretical point of view. To summarize there are some approaches done overall:

A first approach can be found in [23], where equations for the transition probabilities of a (1+1) EA on the dynamic bit matching problem was stated.

Droste [24] looks at the first hitting time (the expected time to hit the optimum for the first time) for a (1+1) evolution strategy on the dynamic bit matching problem. The polynomial $p \in O(\frac{\log n}{n})$, where exactly 1 bit is changed with a given probability defined by p.

A formula predicting the tracking distance of the population from the target is derived by Arnold and Beyer [26]

Finally Branke and Wang [3] also consider the dynamic bit matching problem, and analytically compare different strategies to deal with an environmental change within generation based on similar methods, as in [23]. In their work firstly, two reproduction methods of (1,2) and (1+1) on an Environment Changing between Generations are compared and results are derived which are supported by some empirical tests. As a conclusion derived from the results, it is seen that it would be beneficial to use (1,2) at the beginning then switch to (1+1) at the end of run. As for work on environmental change within generations, they have compared two statements such as:

- *Evaluating two individuals with the respective current fitness function*
- *Delaying the change and use the old fitness function for the second individual*

As an illustration of above statements refer to the Figure below:

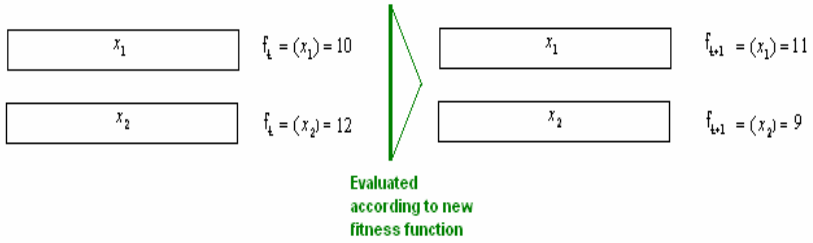

Figure 3.1 : Illustrative example for situation in EAs before and after change

If algorithm will behave according to the first statement individual x_1 will be selected:

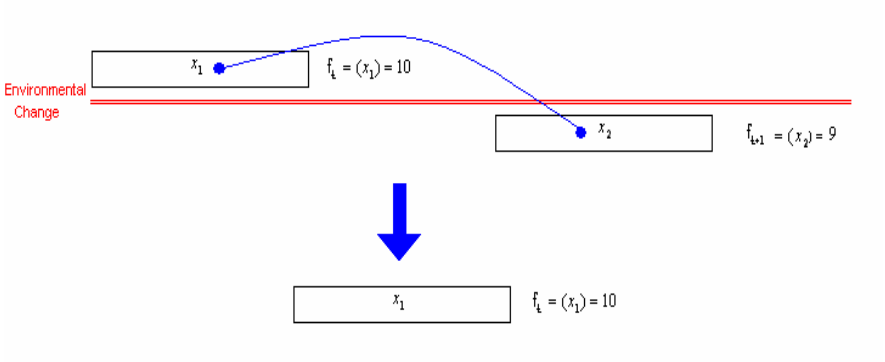

Figure 3.2 : Illustrative Example for Selection According to 1st Statement

On the other hand, according to second statement x_2 will be selected, where it has a less fitness value in new environment:

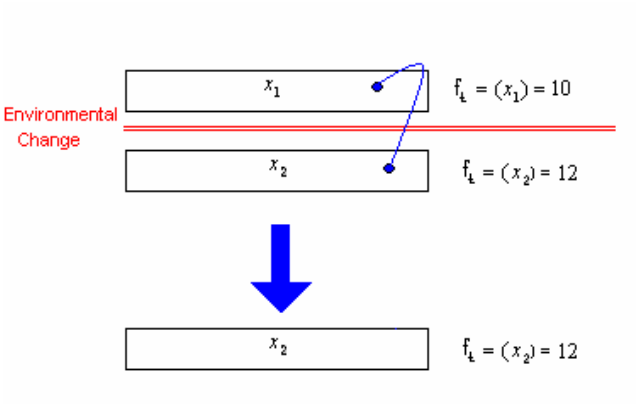

Figure 3.3 : Illustrative Example for Selection According to 2nd Statement

An empirical test with (1,2) reproduction scheme on bit-matching problem was done. The conclusion derived from the result was that according to the severity of the problem, one or the other can be preferred. Thus, for instance if the severity of the environment change is stated as d, ordering from best to worst approaches according to the research results can be summarized in the following way:

d=1 with new individual is best, d=1 with old fitness is next, d=2 with old fitness is in third place, d=2 with new fitness following them, d=3 with old fitness is next before last and d=3 with new fitness is the last in performance.

Above stated theory served an inspiration and basis to our thesis. In order to have any idea where to direct the search after change has happened within generation we have to examine the following approaches proposed by Branke in [9]:

1. Use the changed fitness function for all subsequent individuals, but keep the evaluations of the offspring already evaluated
2. Temporarily reduce the population size. The generation is terminated, and the offspring generated so far serve as basis to generate the next.
3. Re-evaluate all offspring already generated; ignore the change and continue to work with the old fitness function to the end of that generation
4. Ignore the change and continue to work with the old fitness function until all offspring of that generation have been evaluated.

Thus the consecutive section is about the experiments performed in order to compare the four methods stated above.

4 EXPERIMENTS

4.1 Used Tools

Experiments on two benchmarks are performed. While one of them compares four methods on the Bit-Matching Problem (BMP), a unimodal problem, the other compares them on the Single Knapsack Problem (SKP). For both of the experiments the Generational Reproduction Method of EAs is used.

4.1.1 Bit-Matching Problem

In the BMP, the environment is defined as a string and the fitness of each individual is calculated as the sum of the bits matching the environment string. Environment change is applied on the string with a predefined rate by complementing some of the bits. For example if severity of change is 0.05, at least randomly selected 5 bits in string the are complemented.

4.1.2 Single Knapsack Problem

The Single Knapsack Problem (SKP) is defined as:

$$\text{Maximize} \sum_{j \in J} P_j * x_j \qquad \text{where } j = 1, \ldots J \tag{4.1}$$

subject to

$$\sum_{j \in J} w_j * x_j \le C \qquad \text{where } j = 1, \ldots J, \tag{4.2}$$

C : capacity,

w_j : resource consumption of j^{th} item,

P_j : profit of j^{th} item,

x_j : Decision Variable (equal to 0 if item is not included, 1 if it is)

In the SKP, the environment consists of profits, weights and capacity. Also its usage as a penalty based Single Knapsack Problem, where each individual's fitness is defined as

$$F_i = f_i - \text{Penalty}(x) \tag{4.3}$$

$penalty(x) = 0$ if x is feasible,

$penalty(x) > 0$ if x is unfeasible **(4.4)**

where f and Penalty is defined as follows [32]:

$$f_i = \sum_{j \in J} (P_j * x_{ij}) \; where \quad i, j \in J, J = 1,....n$$

$$Penalty = \left({}^{(P_{\max}+1)}\!\!\diagup\!\!{}_{w_{\min}} \right) * \max\{CV(x,i) | i \in I\}$$

$$P_{\max} = \max\{P_j | j \in J\}, J = 1,...,n$$

$$w_{\min} = \min\{w_j | j \in J\}, J = 1,...,n$$

$$CV(x,i) = \max\left(0, \sum_{j \in J} w_j * x_j - c\right) \quad \textbf{(4.5)}$$

Where $w_j > 0$, $p_{\max}$ is the maximum profit, $w_{\min}$ is the minimum resource consumption, $CV(x,i)$ is the maximum constraint violation, n is the number of items and x_{ij} is the j^{th} gene of i^{th} individual in the EA population.

The initial knapsack instances is generated by using David Pisinger's knapsack generator codes in [33]. The generated sample knapsack has 100 items and profits and weights take on values between 0 and 1000. The profit and weight values are highly correlated. The original knapsack generated by the generator had a very low tightness ratio so the capacity value was manually changed in order to have a tightness ratio of 0.75, which makes the initial problem fairly easy.

Following values are used in running the knapsack instance generator given in "generator.c" [33]:

c=generator.c, n=100, r=1000, type=3, i=1, S=1000, where

C : name of the generator's code

N : number of items

R : range of coefficients

Type : 1=uncorrelated, 2=weakly correlated, 3=strongly correlated, 4=inverse strongly correlated, 5=almost strongly correlated, 6=subset-sum, 7=even-odd subset-sum, 8=even-odd knapsack, 9=uncorrelated similar weights, 11=Avis subset-sum, 12=Avis knapsack, 13=collapsing KP, 14=bounded strongly corr, 15=No small weights

I : instance no

S : number of tests in series (typically 1000)

Change of the environment in the SKP, is achieved through changing the profits, weights and capacity according to predefined Upper and Lower bound. The dynamic multi-dimensional knapsack problem generation method given in [22] is modified to be applied to the SKP, as explained below. The initial values for the environment are defined at the beginning of the run according to the formula:

Lower bound of $p_j=p_j*(0.8)$ for each profit

Upper bound of $p_j=p_j*(1.2)$

Lower bound of $w_j=w_j*(0.8)$ for each weight

Upper bound of $w_j=w_j*(1.2)$

Lower bound of $c=c*(0.8)$ for capacity

Upper bound of $c=c*(1.2)$ **(4.6)**

At each change instance, each of the profits, weights and capacities are changed according to the following statement:

$p_j=p_j*(1+N(0,0.05))$

$w_j=w_j*(1+N(0,0.05))$

$c=c*(1+N(0,0.05))$ **(4.7)**

where N(0,0.05) is the random number from the Gaussian distribution with mean=0, and standard deviation=0.05

For more information related to penalty based fitness calculation and towards the analysis of Multiple Knapsack Problem refer to [22]

4.1.3 Generational Reproduction Method

The offspring of the individuals selected from each generation become the entire next generation. No individuals are retained between generations.

4.1.4 Time for Change of Environment

In our experiment, three kinds of periods are tested for change over the entire run, i.e. the beginning, in the middle, and at the early end of the run. To clarify, if time of

change is defined by x, the value of x for the change at the beginning of run is 3, x=10 for medium change, and x=25 for the change at the end of run.

Moreover, different stages in the generation are defined as an offset, which shows at which individual's evaluation the environment has been changed. In our experiments this value is defined as 50, in order to have the environment changed at the middle of generation where 100 individuals exist in the population.

To sum up, for example if x=3, and offset=50 with a population which has size of 100, change happens at (x * population_size + offset) = 350th evaluation.

In our experiments, basically, four methods in previously stated proposal are compared. The main goal of the experiment is to see how to manage when change happens within a generation. First we have compared four methods with x=3,x=10 and x=25. The main aim is to see how the methods manage change which happens at early stage, at the medium stage and at the end. The results are shown in plots as well as in tables.

In order to compare four methods in equal time period giving them an equal chance for recovering, values of offline error at Z evaluations after change are compared, where Z is defined as Z=z*population_size, (in our work z=3).

In each case, 1000 runs are performed and the average of these runs are plotted. Also standard error, which is equal to [(Standard Deviation)/(sqrt (number of runs))] is calculated in order to look at intervals of calculated standard error and see differences in a numerical way.

4.2 Experiment on Bit-Matching Problem

4.2.1 Details of Experiment

Representation: Binary representation

Fitness Evaluation: Fitness evaluated according to Bit-Matching Problem

Selection: Permutational Selection

Crossover: Uniform crossover of rate 0.8

Mutation: Mutation rate of 0.01 is applied

Duplicate-Elimination: For each randomly generated individual at the beginning of population, same individuals are not allowed, as well as for offspring.

Elitizm: Elitist individual of previous generation which are different from currently generated offspring are allowed to join them.

Change Severity: Environment change of 0.1 severity means at most randomly selected 10 bits are flipped when change happens.

Performance Measurement: Offline Error, Best Fitness

4.2.2 Results of Experiment w.r.t. Offline Error Performance

Table 4.1 shows offline errors at Z evaluations after change as well as the offline errors at 4000^{th} evaluations. Moreover Table 4.1 shows the Standard Error at Z evaluations after change and calculated intervals according to Standard Error is given in Table 4.2. As mentioned above, in order to compare four methods by giving to them equal time to recover after change, we have to look at their offline errors at Z evaluations after change. The important point here is that the offline error is calculated in two different ways such as: 1) it is calculated only after the change has happened in order to see the difference of the methods significantly 2) it is calculated from the beginning in order to see the overall performance of the methods.

Thus, in 1^{st} type of calculation ordering of the methods from best to worst according to the values of each method at Z evaluations after change is as follows:

for period 3: 2, 1, 4, 3
for period 10: 2, 1, 4, 3
for period 25: 2, 4, 3, 1

In 2^{nd} type of calculation ordering of the methods from best to worst according to the values of each method Z evaluations after change is as follows:

for period 3 : 2, 1, 4, 3
for period 10 : 1, 2, 3, 4
for period 25 : 1, 3, 2, 4

1st Method: Unexpected good performance of 1^{st} Method where fitness values according to old environment compete with the fitness values according to new environment could be because the changes are not very severe and the landscape is unimodal. Possibly even though the fitness values of the individuals change, ordering of individuals could be staying more or less the same. In order to support this idea pair wise ordering of individuals according to their fitness before and after the change can be examined. Refer to Test 1 performed in Section 4.2.4.

2nd Method: Although performance of method 2 is best in first calculation type, it decreases in overall period with larger period of change. It can be explained possibly as there is more time for the EA, it gets more converged around a local optimum. Then not having enough time with decreased size of the population, there is not sufficient diversity to find optimum. In order to support this idea Test 2, where average hamming distance of reduced population is calculated at different periods of change time in order to have a look at diversity in periods, has been performed in Section 4.2.4.

3rd Method: The bad performance of method 3 in smaller periods or in quick change could be because of the fact that it spends some of its time for reevaluating and has less time for trying to find the new optimum. This idea is supported by the increasing performance with respect to increasing period. For example, for the period of 25, method 3 also gets better because at this moment it also gets some time to look for the new optimum. Moreover from Table 4.2 and illustrated graphs shown on Figure 4.4, Figure 4.5 and Figure 4.6 it can be said that 1st and 3rd Methods' behaviors are similar.

4th Method: Performance of the 4th Method is better than 3rd Method in the 1st type of evaluation. Indeed it seems to be better than the 3rd Method because of the method of calculation of offline error. In the 4th Method after change has happens, offline error is calculated according to the new environment while algorithm itself continues with the old environment by ignoring the change. This idea can be shown by comparing methods according to their Best Fitness Performance [Section 4.2.3]. In the 2nd type of calculation where overall offline error is calculated 4th Method is the worst. Possible reasons can be that it ignores the change and in its time given to converge, it continues to converge around the wrong peak. Thus since even more diversity is lost, it takes longer to move to the new peak. In order to support this idea Test 3 is performed in Section 4.2.4.

Figure 4.1, Figure 4.2 and Figure 4.3 illustrate the performance w.r.t. Offline Errors after change.

Table 4.1 : BMP with Offline Error

Bit Matching Problem												
Offline Error Performance												
Methods	I			II			III			IV		
Periods For Change	3	10	25	3	10	25	3	10	25	3	10	25
Change Happened at Given Fitness Evaluation	350	1050	2550	350	1050	2550	350	1050	2550	400	1100	2600
Change Time + Z	650	1350	2850	650	1350	2850	650	1350	2850	650	1350	2850
Value at Z^{th} Evaluation after change	31.202	22.008	11.614	30.308	21.202	11.216	31.967	22.220	11.604	31.374	22.097	11.553
Value at 4000^{th} Evaluation	10.703	8.639	7.838	10.557	8.477	7.547	11.078	8.720	7.721	10.809	8.702	7.757
Std. Error of 1000 Runs at 4000^{th} evaluations	0.0243	0.0294	0.0264	0.0589	0.0291	0.0286	0.0257	0.0289	0.0279	0.0252	0.0266	0.0269
Std. Error of 1000 Runs at Z^{th} evaluations after change	0.0507	0.0496	0.0361	0.0283	0.0583	0.0363	0.0581	0.0544	0.0358	0.0492	0.0490	0.0347
Ordering according to value at Z^{th} evaluations after change	**2**	**2**	**4**	**1**	**1**	**1**	**4**	**4**	**3**	**3**	**3**	**2**

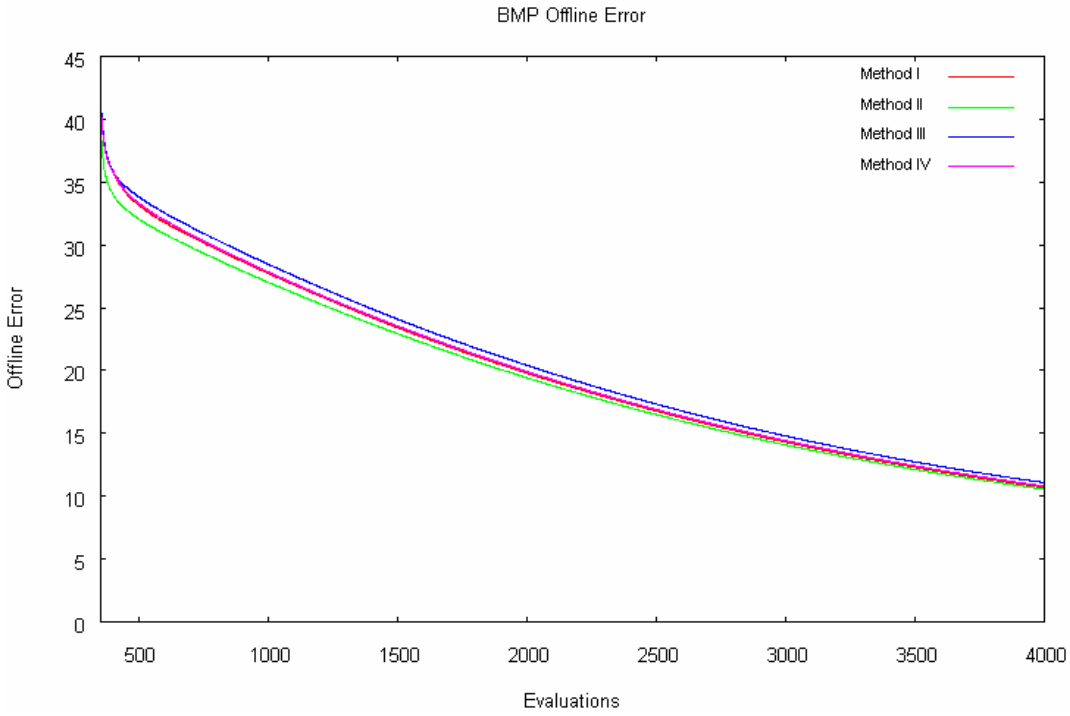

Figure 4.1 : Methods for Period 3 on BMP w.r.t. Offline Error Performance

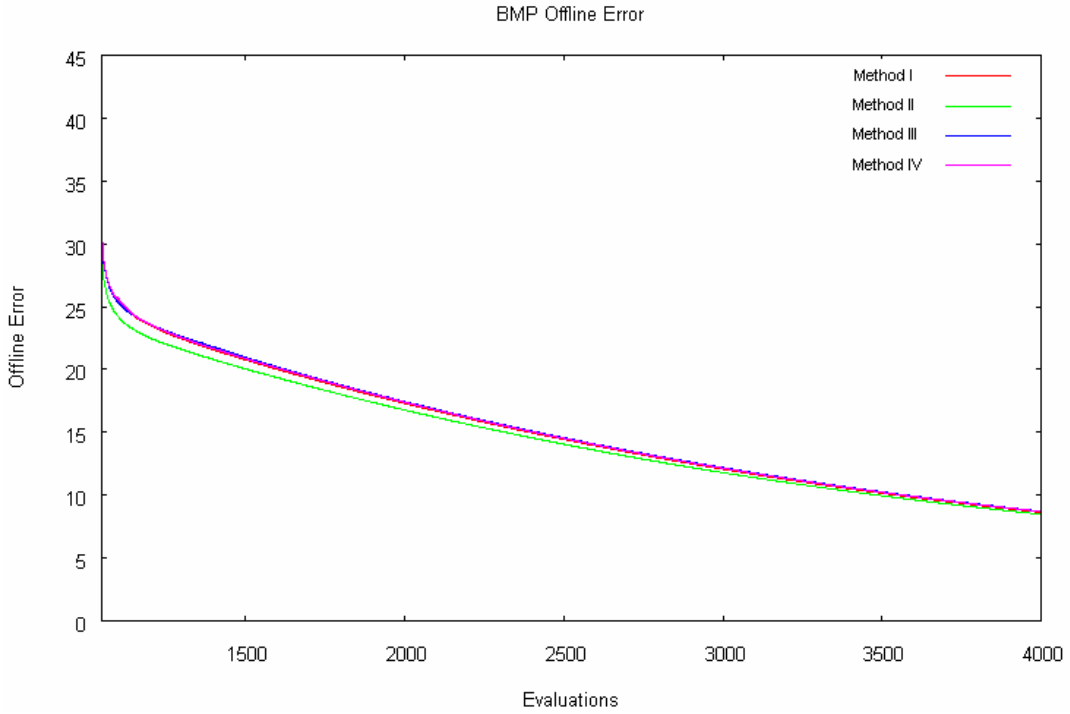

Figure 4.2 : Methods for Period 10 on BMP w.r.t. Offline Error Performance

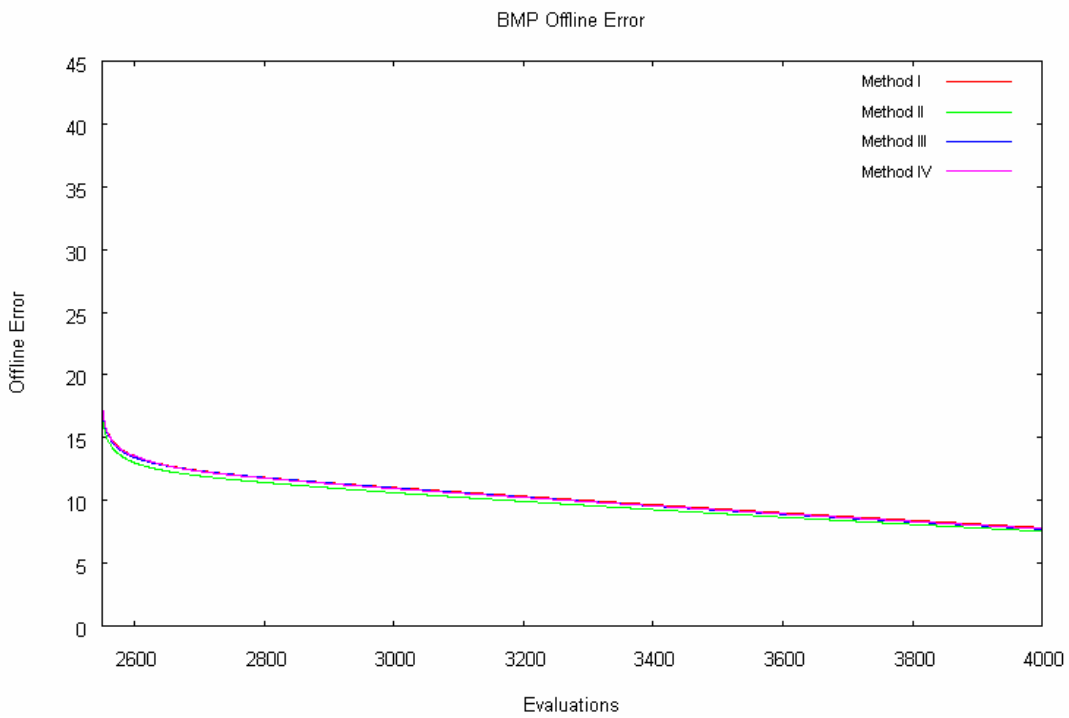

Figure 4.3 : Methods for Period 25 on BMP w.r.t. Offline Error Performance

Table 4.2 : Intervals of Methods at z Evaluations w.r.t. Offline Error Performance

	For period 3	For period 10	For period 25
Method 2	[30.280;30.336]	[21.144;21.260]	[11.180;11.252]
Method 1	[31.151;31.252]	[21.958;22.048]	[11.578;11.650]
Method 4	[31.325;31.423]	[22.048;22.146]	[11.518;11.587]
Method 3	[31.990;32.025]	[22.166;22.274]	[11.568;11.640]

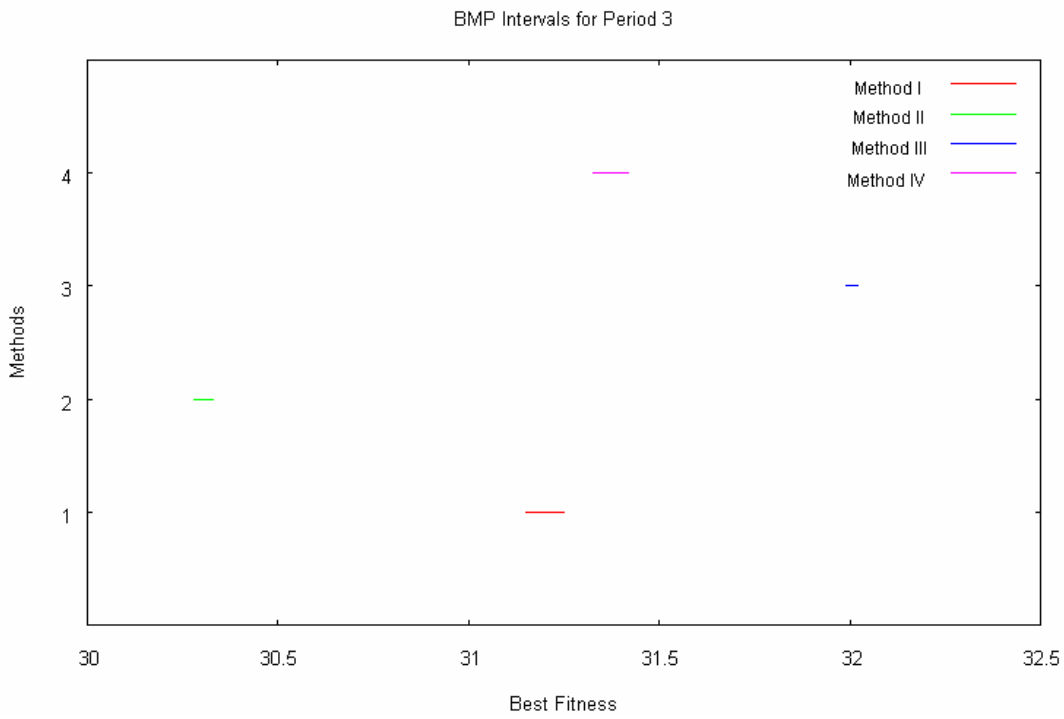

Figure 4.4 : Intervals for Period 3 on BMP w.r.t. Offline Error Performance

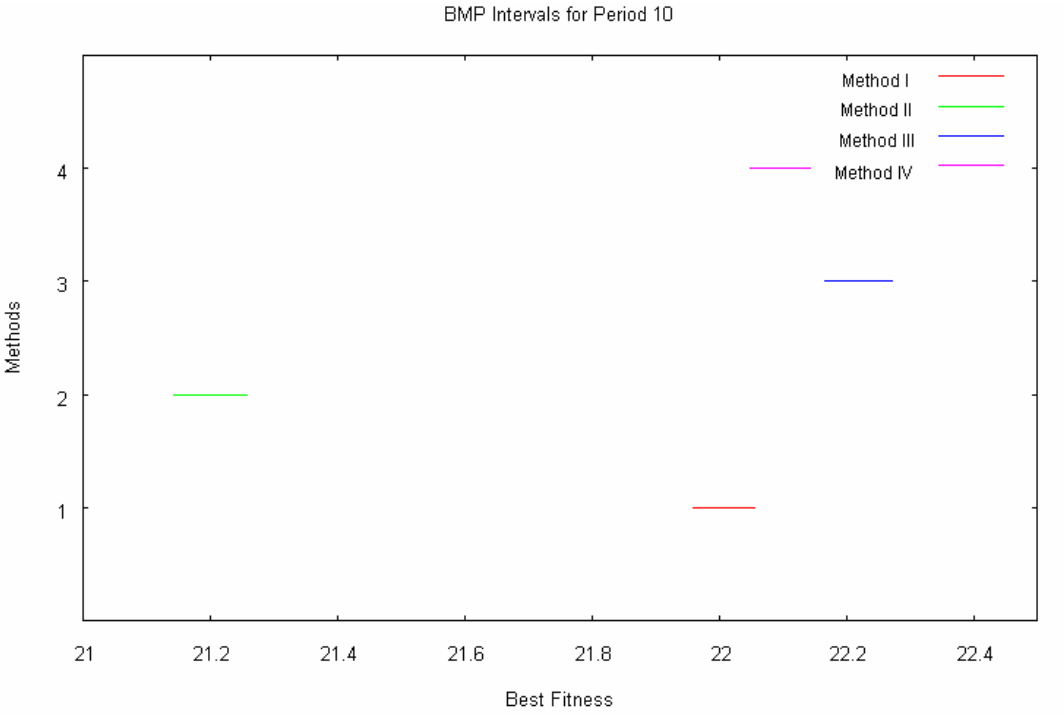

Figure 4.5 : Intervals for Period 10 on BMP w.r.t. Offline Error Performance

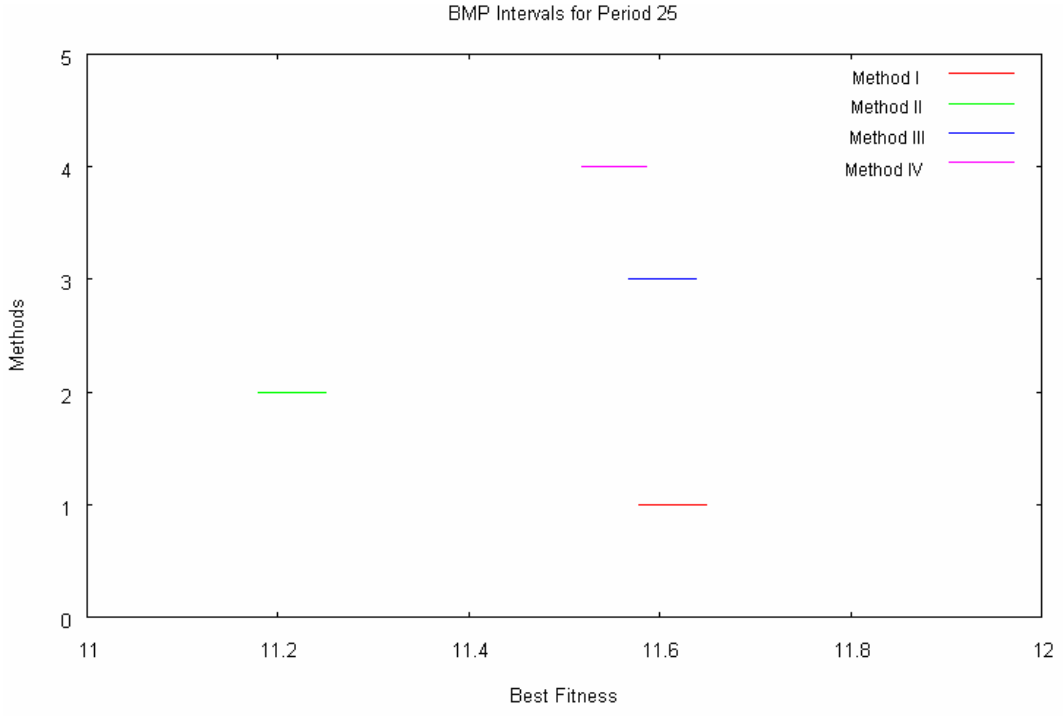

Figure 4.6 : Intervals for Period 25 on BMP w.r.t. Offline Error Performance

4.2.3 Results of Experiment w.r.t. Best Fitness

Table 4.3 shows Best Fitness overall at Z evaluations after change as well as the Best Fitness at 4000th evaluations. As in above experiment, the Standard Error at Z evaluations after change is given in Table 4.3 and calculated intervals according to Standard Error is given in Table 4.4.

Ordering of the methods from best to worst according to the values of each method on Z evaluations after change is as follows:

for period 3 : 2, 1, 4, 3

for period 10 : 1, 2, 3, 4

for period 25 : 1, 3, 2, 4

Here the ordering is same as the offline error calculated overall. It means that performance of methods overall is same with the first experiment where offline error is calculated.

1st Method: As in the previous experiment, we have seen that after change although 1st Method's performance is slightly decreasing it is still best according to other methods.

2nd Method: In this experiment, it is significantly seen that 2nd Method's performance is decreasing with larger periods of change time. We can also say that 2nd Method's decreasing performance is much bigger than the 1st Method's.

3rd Method: The result clearly shows that 3rd Method is having better performance with increasing period of change time. Moreover this experiment have been performed again with BMP but this time w.r.t. Best Fitness Performance in order to show the related performance of 3rd and 4th Methods. As a result we can see that 4th Method is in fact worse than 3rd Method, actually it is the worst one.

Figure 4.1, Figure 4.2 and Figure 4.3 illustrate the performance w.r.t. Best Fitness overall. In graphs it is important that 4th Method taking the change of environment into account after a half generation passed.

Table 4.3 : BMP with Best Fitness

Bit Matching Problem												
Best Fitness Performance												
Method	I			II			III			IV		
Periods For Change	3	10	25	3	10	25	3	10	25	3	10	25
Change Happened at Given Fitness Evaluation	350	1050	2550	350	1050	2550	350	1050	2550	400	1100	2600
Change Time + Z	650	1350	2850	650	1350	2850	650	1350	2850	650	1350	2850
Value at Z^{th} Evaluation after change	72.238	81.401	95.268	72.603	80.923	89.963	71.404	80.921	95.154	72.038	80.539	89.714
Value at 4000^{th} Evaluation	99.707	99.286	96.477	99.700	99.293	96.334	99.637	99.300	96.483	99.676	99.291	96.198
Std. Error of 1000 Runs at 4000^{th} evaluations	0.0151	0.0185	0.0298	0.0148	0.0192	0.0319	0.0154	0.0189	0.0302	0.0154	0.0187	0.0312
Std. Error of 1000 Runs at Z^{th} evaluations after change	0.0672	0.0559	0.0384	0.0659	0.0626	0.0381	0.0619	0.0556	0.0391	0.0635	0.0573	0.0383
Ordering according to value at Z^{th} evaluations after change	**2**	**1**	**1**	**1**	**2**	**3**	**4**	**3**	**2**	**3**	**4**	**4**

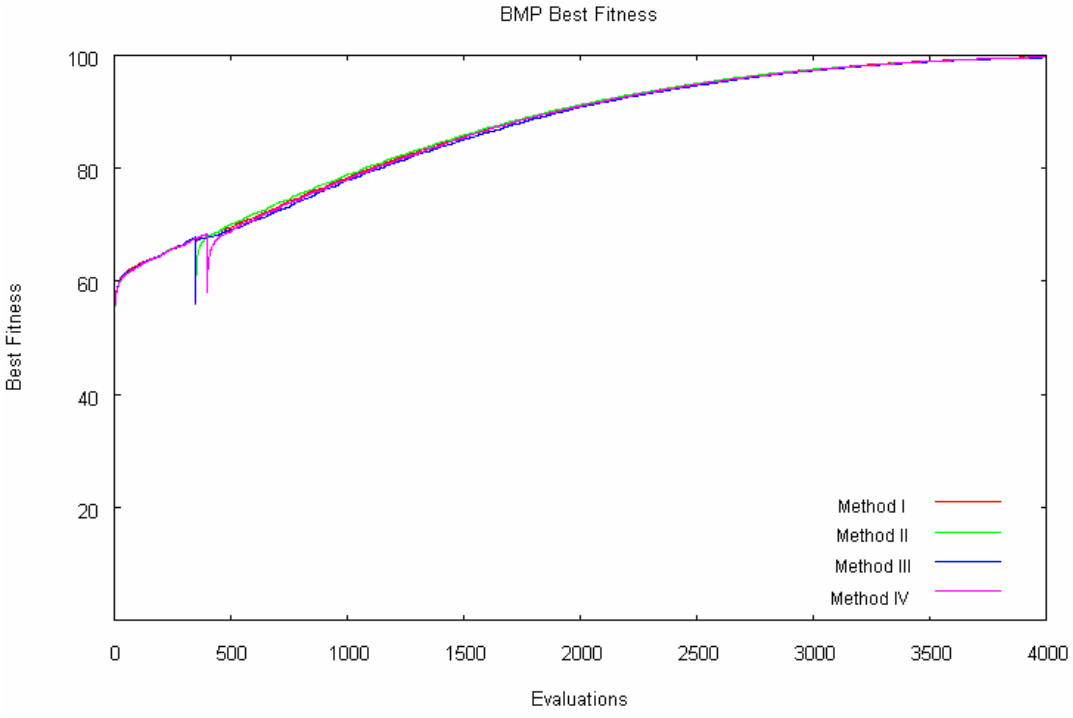

Figure 4-7 : Methods for Period 3 on BMP w.r.t. Best Fitness Performance

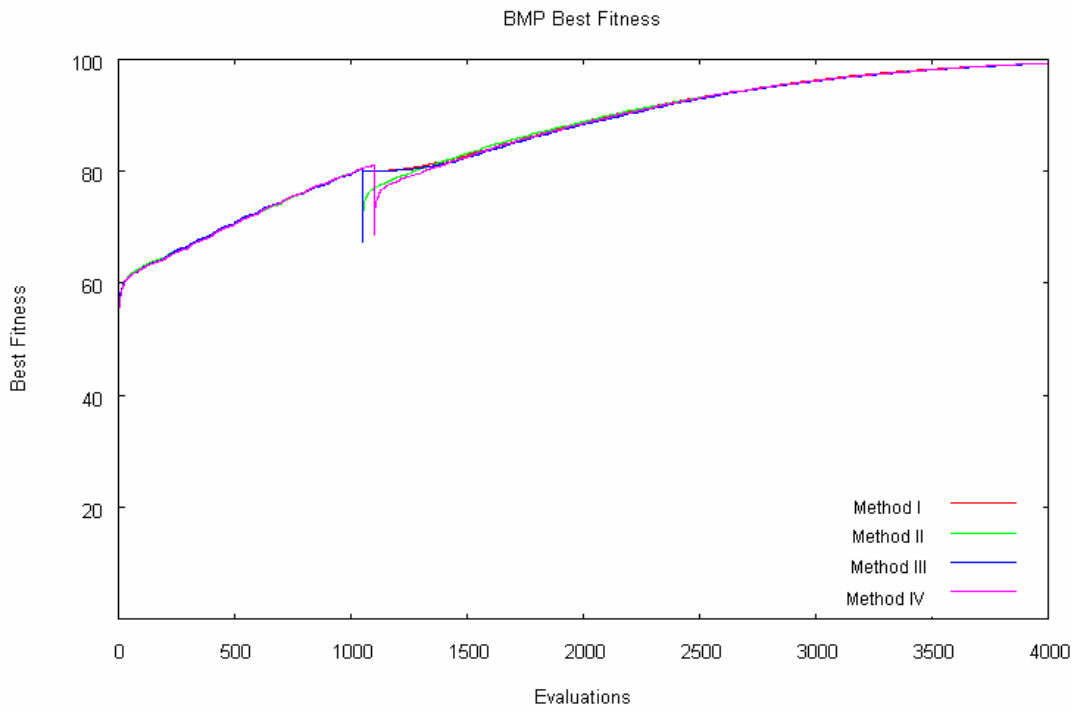

Figure 4.8 : Methods for Period 10 on BMP w.r.t. Best Fitness Performance

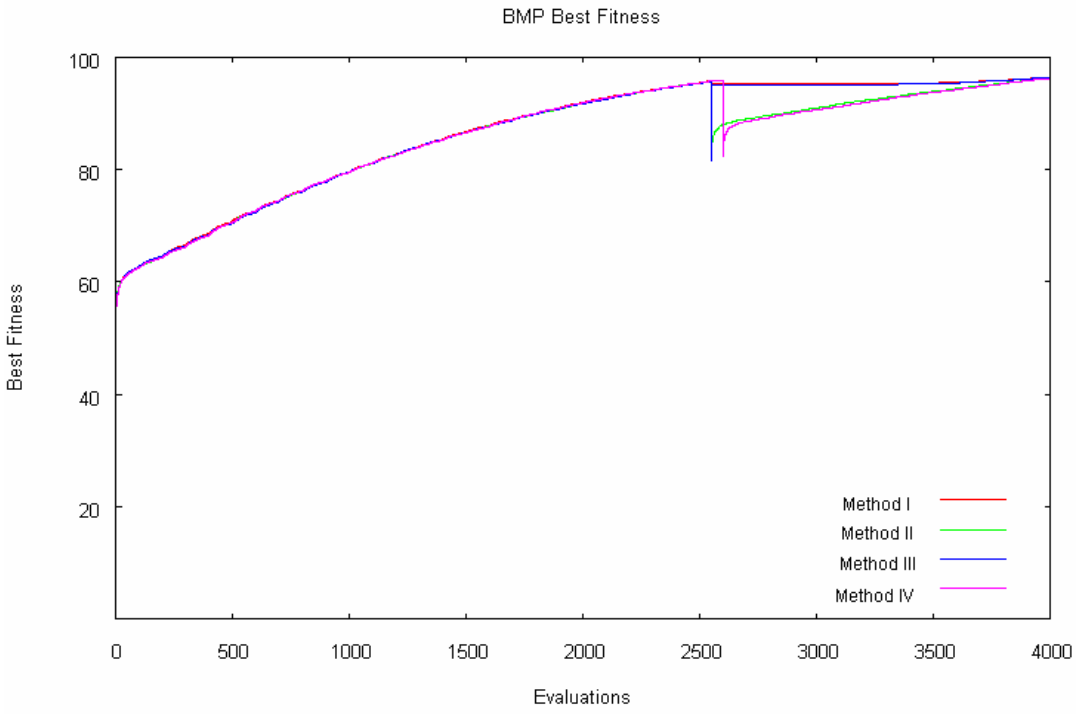

Figure 4.9 : Methods for Period 25 on BMP w.r.t. Best Fitness Performance

Table 4.4 : Intervals of Methods at z Evaluations

	For period 3	For period 10	For period 25
Method 2	[72.564;72.696]	[80.860;80.986]	[89.925;90.001]
Method 1	[72.171;72.305]	[81.345;81.457]	[95.230;95.306]
Method 4	[71.975;72.102]	[80.482;80.596]	[89.676;89.752]
Method 3	[71.342;71.466]	[80.865;80.977]	[95.115;95.193]

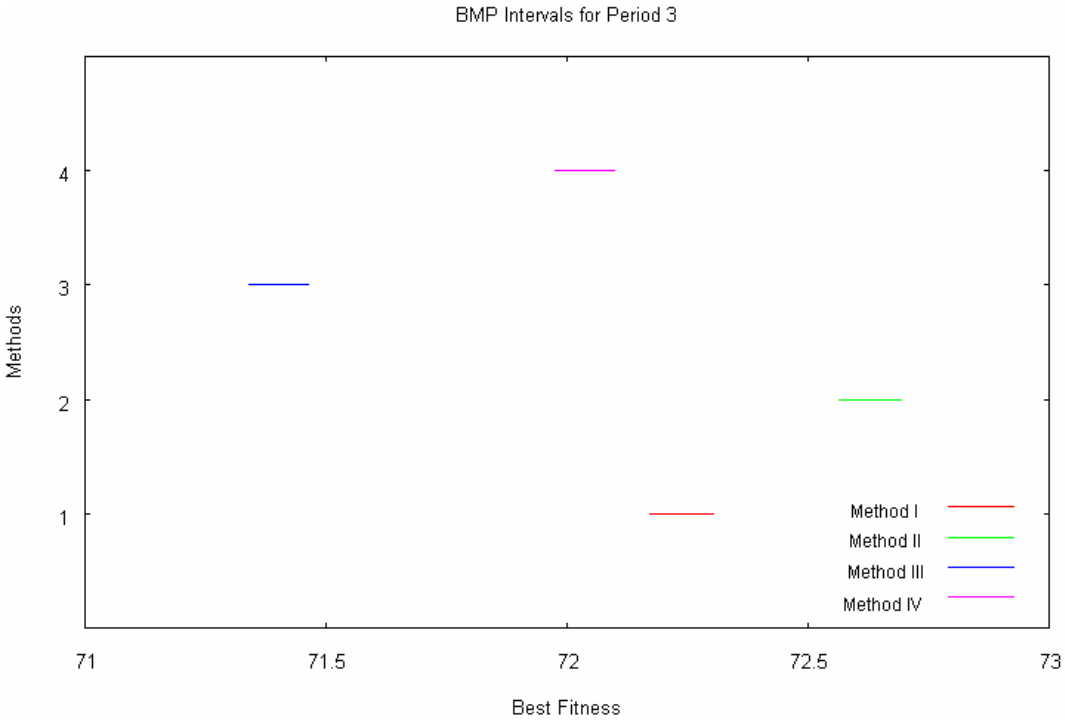

Figure 4.10 : Intervals for Period 3 on BMP w.r.t. Best Fitness Performance

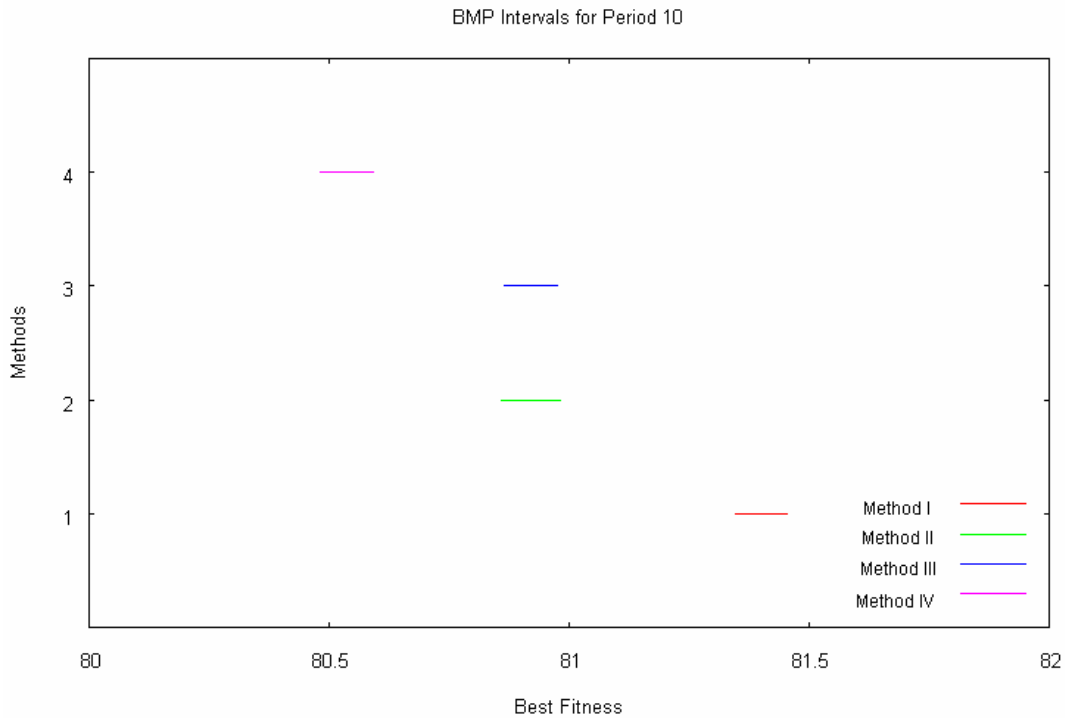

Figure 4.11 : Intervals for Period 10 on BMP w.r.t. Best Fitness Performance

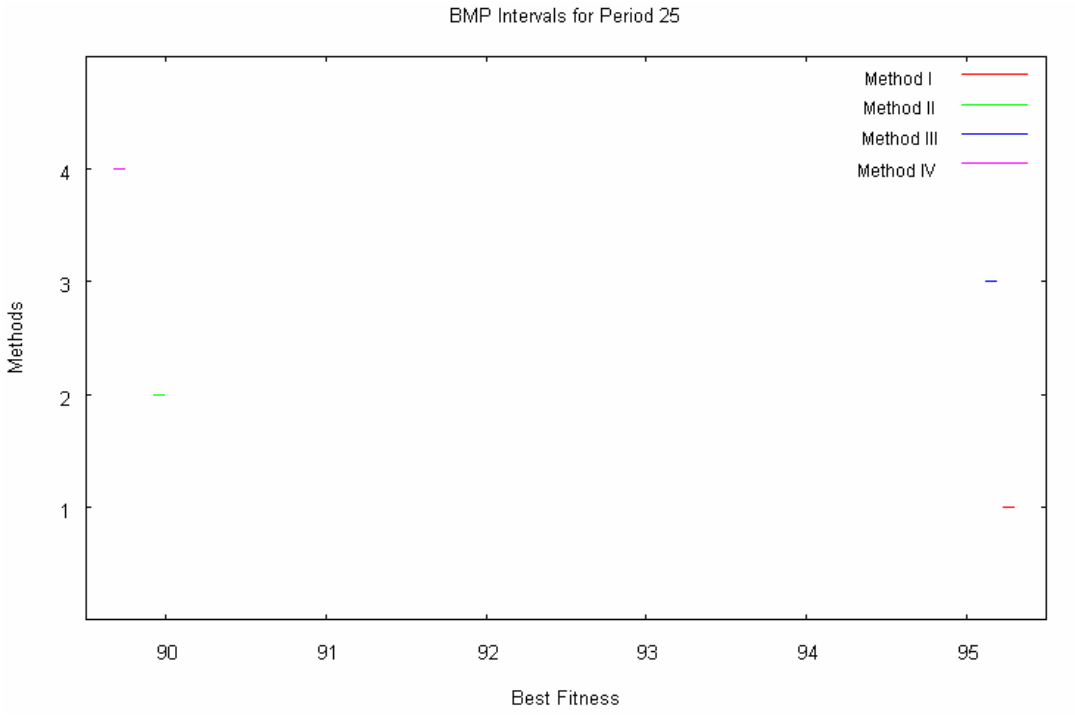

Figure 4.12 : Intervals for Period 25 on BMP w.r.t. Best Fitness Performance

4.2.4 Tests Performed According to Results of Experiment on Bit-Matching Problem

Test 1: The aim of this test is to detect the severity of change or in other words to have a look at the similarity of environments before and after the change.

- All individuals in the population (half with old fitness, half with new fitness) at the generation where change has happened are ordered - it is the first string
- First half of population was re-evaluated according to the new environment and all individuals in the population are ordered-it is the second string. As an ordering algorithm Bubble Sort Algorithm has been used in order to not change the related order of individuals with the same fitness value.
- Pair wise "better" relationships between all individual pairs in both strings are compared. The number of differences (worst case is n*(n-1)/2 where n is the number of individuals. It is equal to 4990 in our experiment) is calculated.

Illustration of Test 1is as follows, where x_i, i $\epsilon\{1\ldots6\}$ is for individuals and current time is defined as t so that fitness function at the current time is defined as f_t, the time after change has happened is defined as t+1, also fitness function at that time is defined as f_{t+1}.

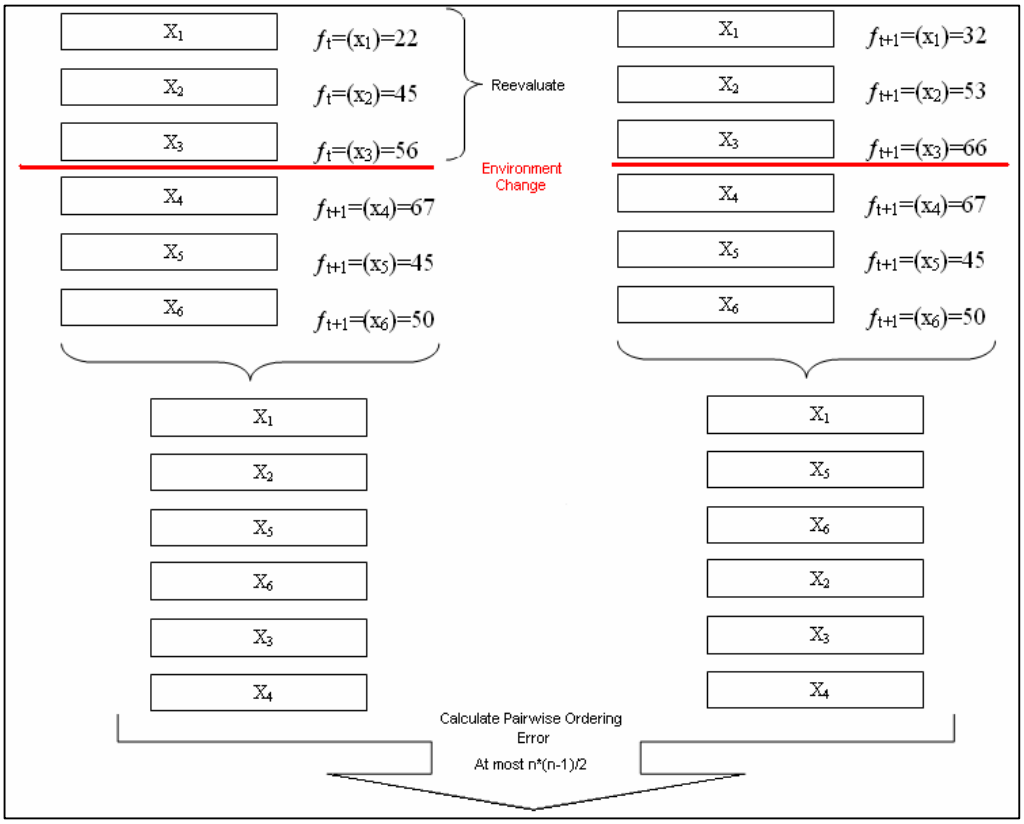

Figure 4.13 : Illustration of Test 1

Results:

for period 3: 715.7 different out of 4990

for period 10: 1067.0 different out of 4990

for period 25: 1560.5 different out of 4990

As it can be seen that similarity of environments before and after the change is decreasing as the period of change time is increasing. Therefore this result has proved our assumption about 1st Method. Thus, as the period of change time is decreasing, the poopulation has time to converge and for that reason severity of change is larger than in small periods. Similarly in small periods of change time, the population is distributed and is not huddled around the optimum, change severity is small and orderings of individuals do not change roughly.

Test 2: The aim of this test is to explain the claim of the experiments as to why the 2nd Method looses its performance with the increasing period of change time by calculating the average hamming distances of the half population before change.

Illustration of Test 2 is as follows, where where x_i, i $\epsilon\{1\ldots6\}$ is for individuals:

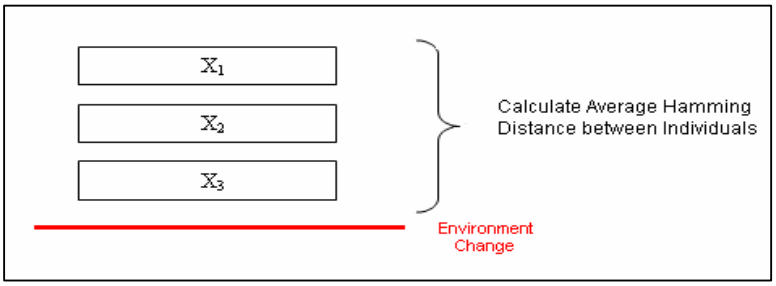

Figure 4.14 : Illustration of Test 2

Results:

for period 3: 23.309

for period 10: 18.060

for period 25: 8.282

As seen in the results above, the average hamming distance of the half population before change is decreasing as the period for change time is increasing. Therefore, the result supports the assumption about 2nd Method. Thus 2nd Method has worse performance the increasing period of change time because of the decreasing diversity of the population.

Test 3: The aim of the test is to show that 4th Method is losing its diversity by continuing to converge around the wrong peak. In order to show this, average hamming distance of the half population, which is AVR1, before change is calculated and compared with the average hamming distance of all the population in that generation which is AVR2.

Illustration of Test 3 is as follows, where x_i, i $\epsilon\{1\ldots6\}$ is for individuals and current time is defined as t so that fitness function at the current time is defined as f_t. Since Test 3 is applied for 4th Method, fitness function is same after change until the end of current generation:

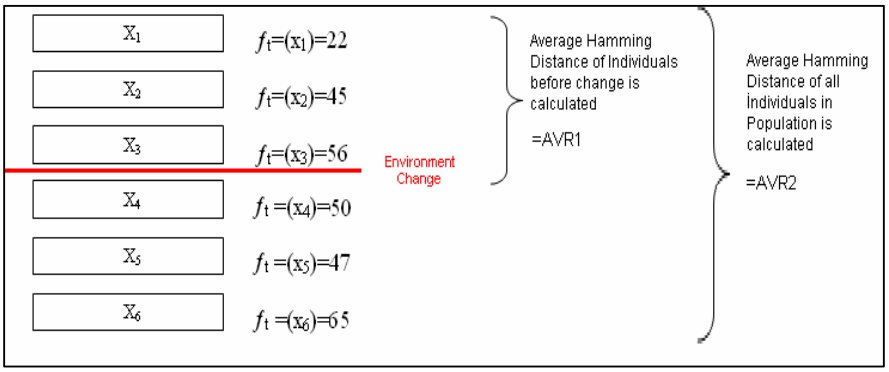

Figure 4.15 : Illustration of Test 3

Results:

for period 3: AVR1=22.829; AVR2= 23.308

for period 10: AVR1=17.713; AVR2= 18.101

for period 25: AVR1=8.165; AVR2= 8.342

It can be seen from the results that the diversity of population before change and the diversity of population after change are nearly same. To say in a different way their difference is decreasing as the period of change is increasing. Thus it supports the idea about 4th Method that since it ignores the change, in its time given to converge it continues to converge around the wrong peak. Thus even more diversity is lost, it takes longer to move to the new peak.

Test 4: The aim of the test is to show that 1st Method have larger diversity than in 4th Method. To do this:

a) keep the old fitness values in the current generation (Method 4), look at the diversity in next generation

b) use new fitness values after the change (Method 1), look at diversity in next generation. It is expected that the diversity in 4th Method is getting smaller as period of change is increasing.

Illustration of Test 4 is as follows, where where x_i, i $\epsilon\{1\ldots6\}$ is for individuals and current time is defined as t so that fitness function at the current time is defined as f_t, the time after change has happened is defined as t+1, also fitness function at that time is defined as f_{t+1}. M4 and M1 are stated for 4th and 1st Methods.

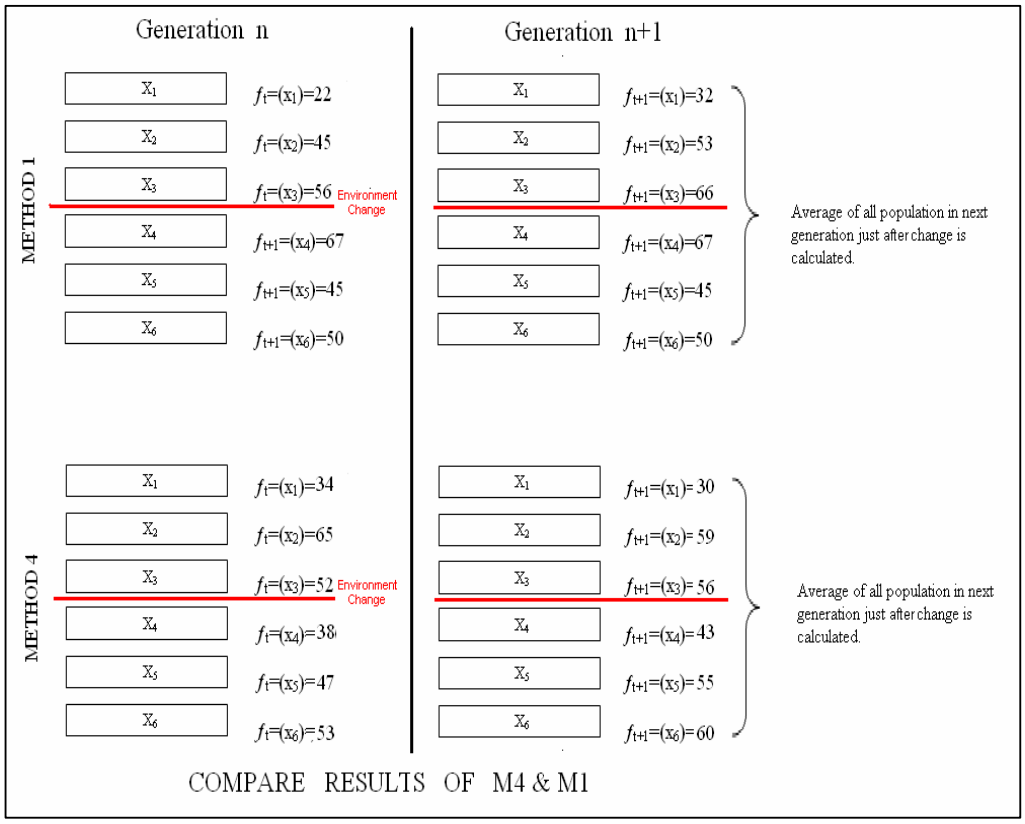

Figure 4.16 : Illustration of Test 4

Results:

a) Avr. Diversity of M1: period 3 =22.157; period 10 =16.911; period 25 = 8.066

b) Avr. Diversity of M4: period 3 =22.607 ; period 10 =17.294; period 25 = 7.863

It is shown that in smaller periods, there are ignorable small differences between diversity of populations in 4th Method and 1st Method, and the diversity of population in 4th Method is getting smaller as time passes.

Test 5: The aim of the test is to compare diversity of population in 2nd Method (M2) and 4th Method (M4) in next generation just after change.

a) In M4 look at the diversity in next generation just after change

b) In M2 look at the diversity in next generation just after change

Illustration of Test 5 is as follows, where x_i, i $\epsilon\{1\ldots6\}$ is for individuals and current time is defined as t so that fitness function at the current time is defined as f_t, the time

after change has happened is defined as t+1, also fitness function at that time is defined as f_{t+1}. M4 and M2 are stated for 4th and 2nd Methods.:

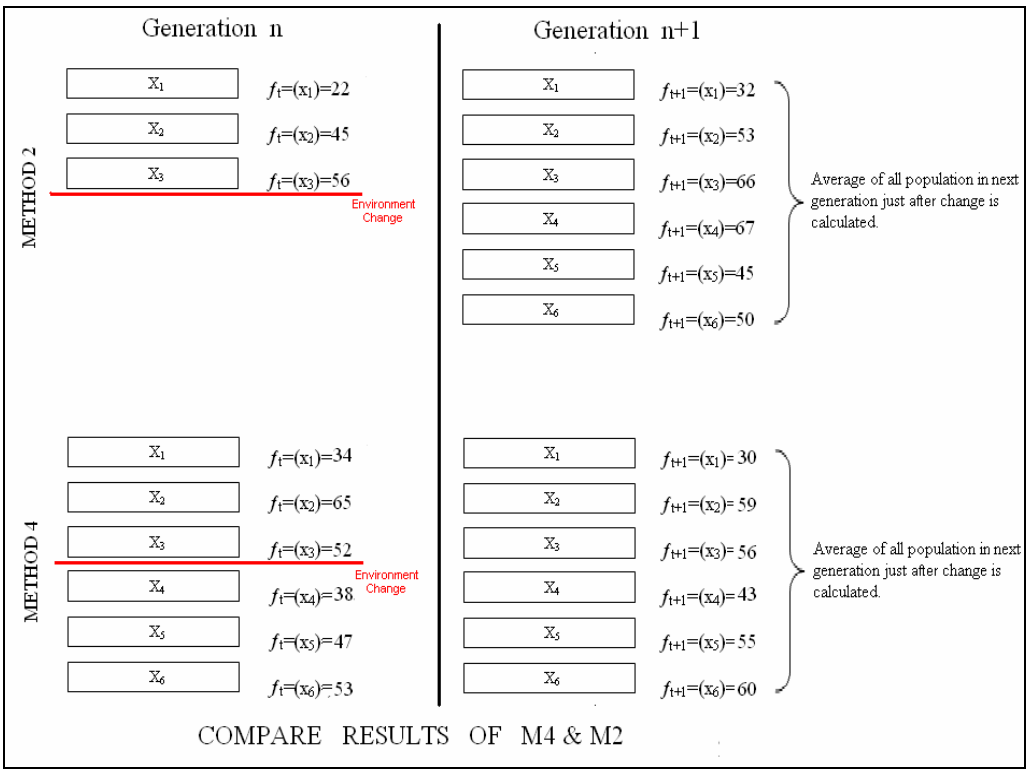

Figure 4.17 : Illustration of Test 5

Results:

a) Avr. Diversity of M2: period 3 = 23.309; period 10 = 18.060; period 25 = 8.282

b) Avr. Diversity of M4: period 3 =22.607 ; period 10 =17.294; period 25 = 7.863

4.3 Experiment on Single Knapsack Problem

4.3.1 Details of Experiment

Representation: Binary representation

Fitness Evaluation: Fitness evaluated according to Single Knapsack problem

Selection: Permutational Selection

Crossover: Uniform crossover of rate 0.8

Mutation: Mutation rate of 0.01 is applied

Duplicate-Elimination: For each randomly generated individual at the beginning of population, same individuals are not allowed, as well as for offspring.

Elitizm: Elitist individual of previous generation which are different from currently generated offspring are aloowed to join them.

Change Severity: Severity of Environment is changed according to randomly generated numbers from Gaussian distribution with mean=0, and standard deviation=0.05.

Performance Measurement: Best Fitness Performance

4.3.2 Results of Experiment

Table 4.5 shows Best Fitness overall at Z evaluations after change as well as the Best Fitness at 4000^{th} evaluations. As in above experiments, the Standard Error at Z evaluations after change is given in Table 4.3 and the calculated intervals according to Standard Error is given in Table 4.4. As it is can be read from intervals' illustrations, in Figure 4.16, 1^{st} and 2^{nd} Method has same performance at period of 3. In figure 4.17 it can be seen that differences between Methods' performances are increasing. Also in Figure 4.17 and Figure 4.18, it can be seen that 2^{nd} Method and 4^{th} Method are same as period of change time is increasing. Here the performance of 2^{nd} Method became worse than in BMP.

Ordering of the methods from best to worst according to the values of each method on Z evaluations after change is as following:

for period 3 : 2, 1, 4, 3

for period 10 : 1, 3, 2, 4

for period 25 : 1, 3, 2, 4

One important issue to note is that if change has happened in an early stage, independently from the problem, ordering according to their performance is same in all experiments. Thus it is in a way that 2^{nd} Method is best, than 1^{st} Method is next, after that 4^{th} Method is at third place and the 3^{rd} Method is the last.

1st Method: Performance of 1^{st} Method is better than in BMP. It can be seen in Figure 4.10 and in Figure 4.16. To clarify, the intervals of 1^{st} Method and 2^{nd} Method do not intersect in BMP, while in SKP they fall down in same interval. To look at the pair wise ordering of individuals according to their fitness before and after change can clarify the situation. Refer to Test 1 performed in Section 4.3.3.

2nd Method: 2nd Method has better performance in BMP relatively than in SKP. This idea can be supported by the work in [32]. In this work feasible SKP space is defined as an area which is surrounded by the boundary of feasibility. The penalty method causes individuals trying to move towards the border. Individuals outside the border are unfeasible individuals. It is possible that individuals which are close to converging before the change may become unfeasible after the change because the feasible space relocates and the individuals may fall outside the boundary. For this reason, 2nd Method can have its performance decreasing in SKP. Why its performance is same as 4th Method's performance can be shown by Test 5 in Section 4.3.3 by calculating the diversity of them in the next generation just after change.

3rd Method: It performs better with larger periods because of the reason stated in the experiment done on BMP. Thus the bad performance of method 3 in smaller periods or in quick change could be because of the fact that it spends some of its time for reevaluating and has less time for trying to find the new optimum. With increasing change periods, the performance of the 3rd Method also increases since it has some more time to converge.

4th Method: 4th Method is the worst. Possible reasons as in other experiments can be that since it ignores the change, in its time given to converge it continues to converge around the wrong peak. Thus it is worse than the 3rd Method where it also losies time by reevaluating half of the population according to the new environment, because it loses time as well as diversity by continuing to converge to the wrong direction.

Table 4.5 : SKP with Best Fitness

Simple Knapsack Problem												
Best Fitness												
Method	I			II			III			IV		
Periods	3	10	25	3	10	25	3	10	25	3	10	25
Overall Best fitness before change	40465.180	43151.707	43452.746	40295.383	43149.867	43453.266	40292.938	43155.449	43456.008	40821.547	43177.785	43461.316
Fitness just after change	23779.584	-452502.125	-466997.875	7910.250	-498276.000	-538392.563	24554.998	-536500.375	-557693.438	12386.626	-451686.969	-516873.094
Overall Best Fitness at z evaluation	42520.1	43915.65	44092.21	42520.24	43284.63	43440.54	42160.12	43883.97	44090.6	42380.9	43248.543	43405.230
Order	2	1	1	1	3	3	4	2	2	3	4	4
Standard Error at z	42.64	39.76	39.33	49.24	62.36	62.49	37.07	39.05	38.13	48.11	62.15	62.33
Overall Best Fitness at 4000 evaluation	43996.836	44333.137	44330.227	43933.941	43910.090	43781.406	43986.555	44330.391	44328.906	43927.42	43912.97	43783.44
Order	1	1	1	3	4	4	2	2	2	4	3	3
Standard Error at 4000	62.37	48.86	45.56	65.10	65.73	65.66	61.66	48.65	45.13	65.52	65.59	65.89

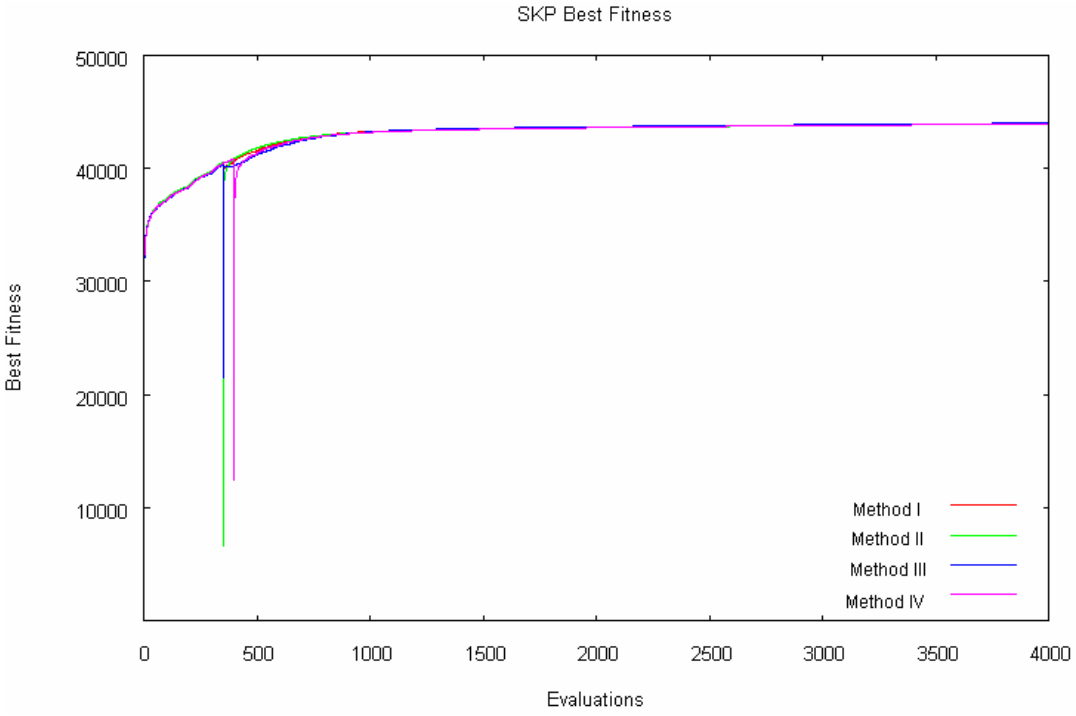

Figure 4.18 : Methods for Period 3 on SKP w.r.t. Best Fitness Performance

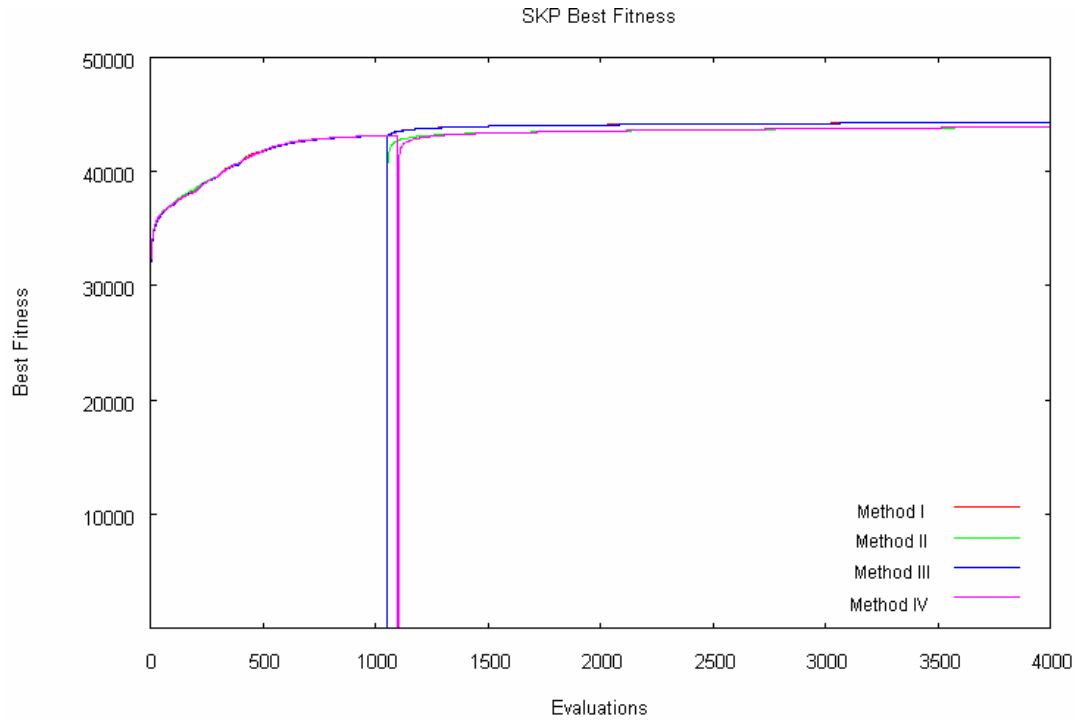

Figure 4.19 : Methods for Period 10 on SKP w.r.t. Best Fitness Performance

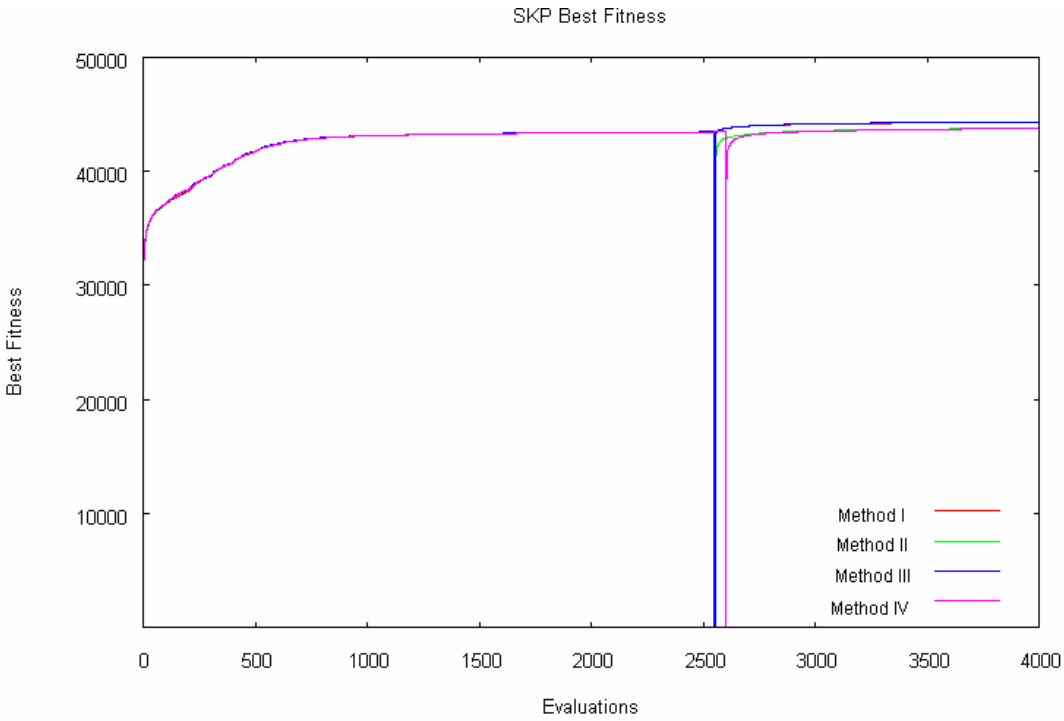

Figure 4.20 : Methods for Period 25 on SKP w.r.t. Best Fitness Performance

Table 4.6 : Intervals of Methods at z Evaluations

	For period 3	For period 10	For period 25
Method 1	[42477.6;42562.7]	[43876.1;43955.4]	[44052.9;44131.5]
Method 3	[42123.1;42197.2]	[43844.9;43923.0]	[44052.5;44128.7]
Method 2	[42470.9;42569.5]	[43222.3;43347.1]	[43378.1;43503.0]
Method 4	[42332.8;42429.0]	[43186.4;43310.7]	[43342.9;43467.6]

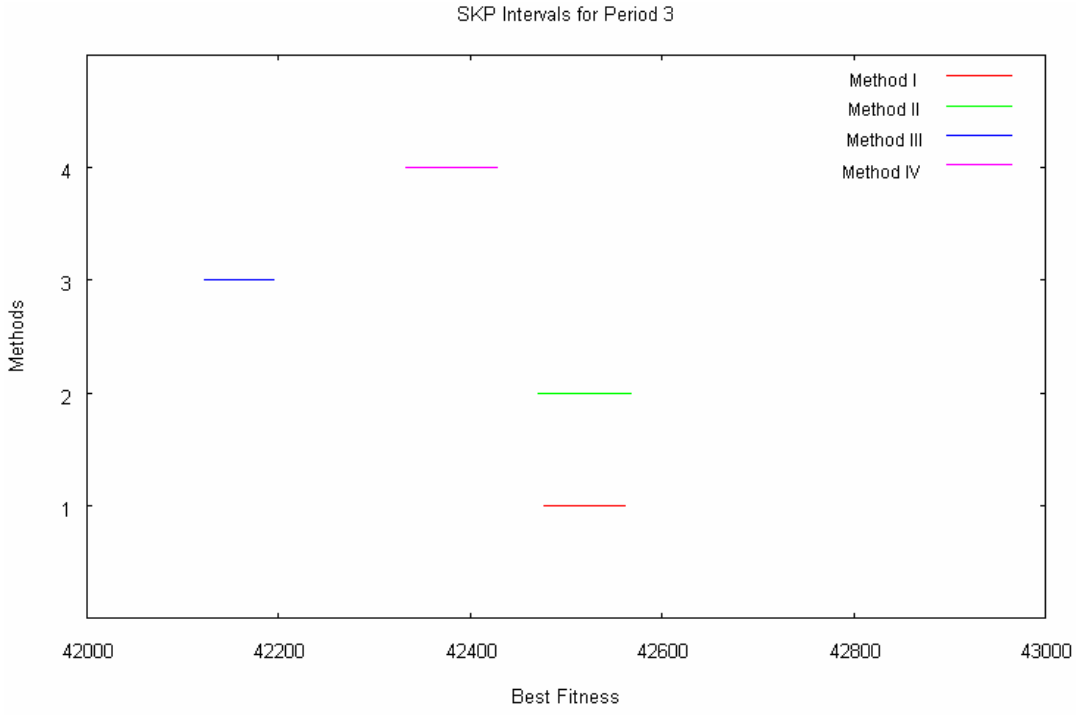

Figure 4-21 : Intervals for Period 3 on SKP w.r.t. Best Fitness Performance

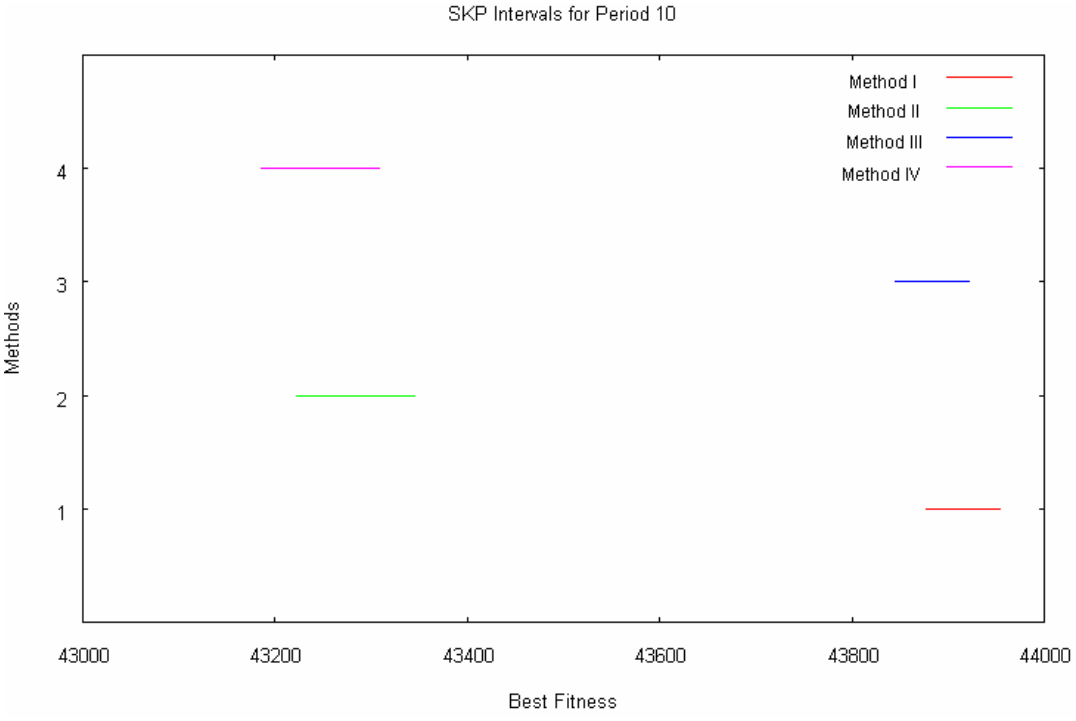

Figure 4-22 : Intervals for Period 10 on SKP w.r.t. Best Fitness Performance

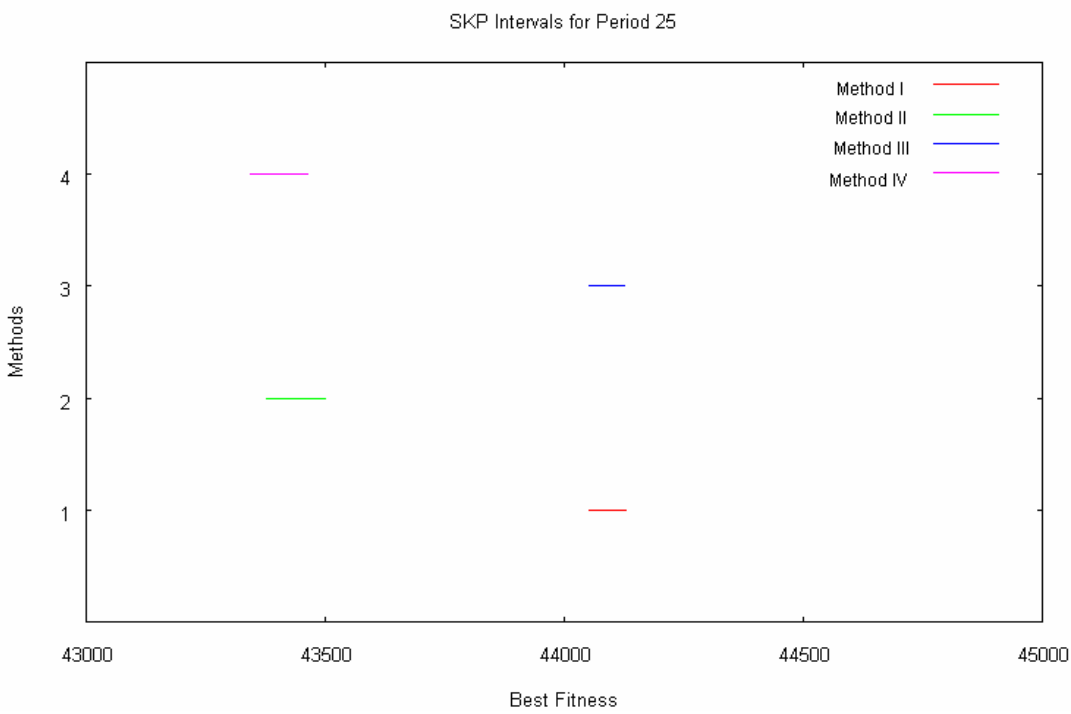

Figure 4.23 : Intervals for Period 25 on SKP w.r.t. Best Fitness Performance

4.3.3 Tests Performed According to Results of Experiment on Single Knapsack Problem

Test 1: The aim of test is to show the reason of 1st Method having better performance than in BMP [Figure 4.13].

- All individuals in the population (half with old fitness, half with new fitness) at the generation where change has happened are ordered-it is the first string
- First half of population is re-evaluated according to the new environment and all individuals in the population (all with new fitness) are ordered-it is the second string
- Pair wise "better" relationships between all individual pairs in both strings are compared. The number of differences (worst case is n*(n-1)/2 where n is the no of individuals) are calculated.

Results:

for period 3: 96.7 differences out of 4990

for period 10: 680.60 differences out of 4990

for period 25: 751.6 differences out of 4990

As a result, it is shown that similarity between populations before and after the change is much smaller than in BMP. Thus it explains the relatively better performance of 1st Method in SKP.

Test 2: The aim of the test is to have a look at diversity of the reduced population after the change according to different periods and explain the behavior of having a worse performance than in BMP. In order to show it, as it was explained before, the average hamming distance of the half population before the change is calculated for different periods of change time [Figure 4.14].

Results:

for period 3: 23.287

for period 10: 19.629

for period 25: 18.351

Although the diversity of the population is much larger than in the experiment on BMP, it is not sufficient to find the optimum, since SKP is a multimodal problem which is harder that the BMP.

Test 3: The aim of the test is to explain the poor performance of 4th Method. In order to do that the average hamming distance of the half population, which is AVR1, before change is calculated and compared to the average hamming distance of the whole population in that generation which is defined as AVR2 [Figure 4.15].

Results:

for period 3: AVR1=22.841; AVR2= 23.328

for period 10: AVR1=19.280; AVR2= 19.675

for period 25: AVR1=17.662; AVR2= 18.024

Results show as in Test 2, although the diversity of population is much larger than in BMP, it is not enough to converge.

Test 4: The aim of the test is to show the reason of the significant difference between 1st Method which is the best in SKP and 4th Method which is the worst in SKP. In the BMP experiment, Test 4 has shown that although M1< M4, because of

the methods interpretation M1 has better performance. It is expected in SKP that M1>M4. In order to show this[Figure 4.16]:

a) keep the old fitness values in the current generation (M4), look at the diversity in next generation

b) use new fitness values after the change (M1), look at diversity in next generation.

Results:

a) Avr. Diversity of M1: period 3=22.028; period 10 = 19.416; period 25 = 18.267

b) Avr. Diversity of M4: period 3=22.028; period 10 = 19.260; period 25 = 17.909

As expected M1>M4, thus supports the idea stated above.

Test 5: The aim of the test is to have a look at similar performance of 2^{nd} Method with 4^{th} Method. In experiment on BMP, 2^{nd} Method has much better performance than 4^{th} Method and in Test 5 in Section 4.2.4 it was shown that the diversity of population in 2^{nd} Method is larger than in 4^{th} Method. In SKP because of their nearly same performance it is expected that M2<M4 or M2=M4 [Figure 4.17].

a) In M4 look at the diversity of population in next generation just after change

b) In M2 look at the diversity of population in next generation just after change

Results:

a) Avr. Diversity of M2: period 3 = 21.313; period 10 = 18.858; period 25 = 17.767

b) Avr. Diversity of M4: period 3 = 22.028; period 10 = 19.260; period 25 = 17.909

Results have shown that M2<M4 and it explains their similar performance in SKP.

The results obtained from the experiments can be summarized as below:

- If changes happen in the early stage of the run, independently of the problem, 2^{nd} Method can be a variant for managing changes within a generation in designing a suitable algorithm.

- If change with a small severity has happened, i.e. environments before and after the change are similar, 1st Method can be preferred as a variant for 3rd Method, where application is costly if changes happen frequently.

- On the other hand, looking at the results for the 4th Method, it can be said that it is not a suitable way to ignore the changes in a generation. Thus ignoring the current change affects performance by converging late because of continuing converging around the wrong peak.

Therefore now we know that assuming changes to occur only between generations can lead us to unsuitable, late converging populations. The reason is if change has happened almost within a generation, this interpretation becomes the same as the 4th Method and we see that it is the worst way of managing changes within generations.

4.4 Further Experiments

4.4.1 Appropriateness of Periods

Defining of periods of change time is analyzed by plotting the best fitness graph of BMP and SKP without any change. As a result, it is seen that defining periods of change time as 3, 10 and 25 is quite meaningful according to the plot. Graphs are given below in Figure 4.24 and 4.25, where the averages of the best fitness at each evaluation over 1000 runs are plotted.

It is seen that period 3 in Figure 4.24 with the early stage of overall run. The entire run can be divided into two parts where at first part tangent of a curve is bigger than the one in the second part. Thus the period 10 is in the middle of the first part, that it is appropriate to observe the run in the middle of the progress of performance. It can be seen that the period 25 is near towards the early end of the run.

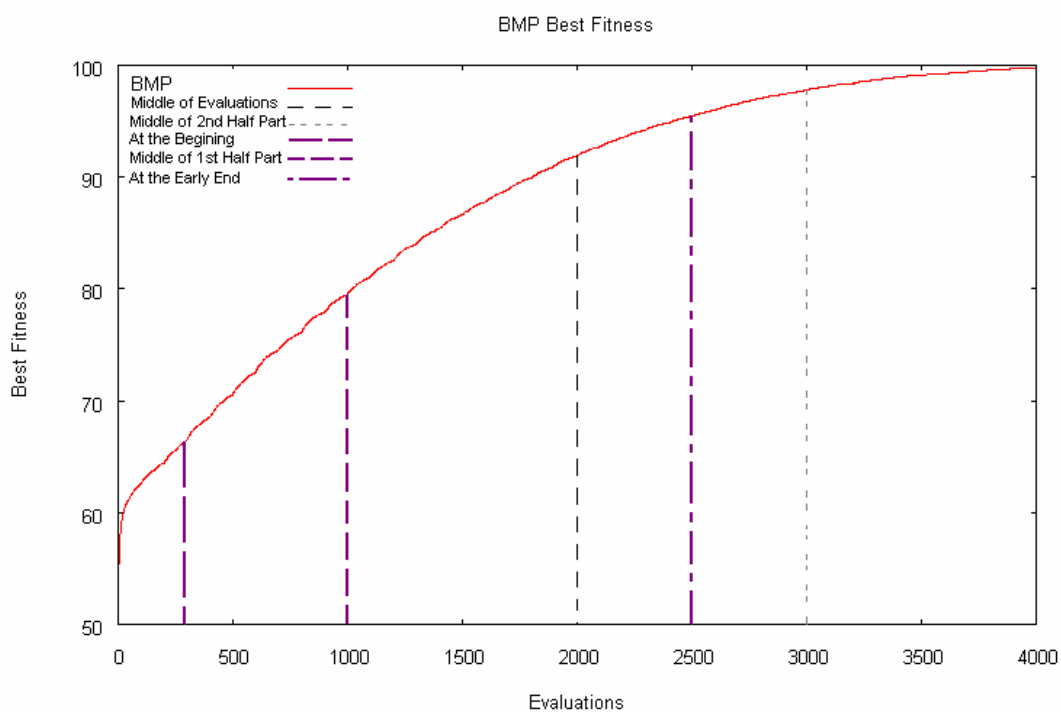

Figure 4.24 : BMP Best Fitness without Change

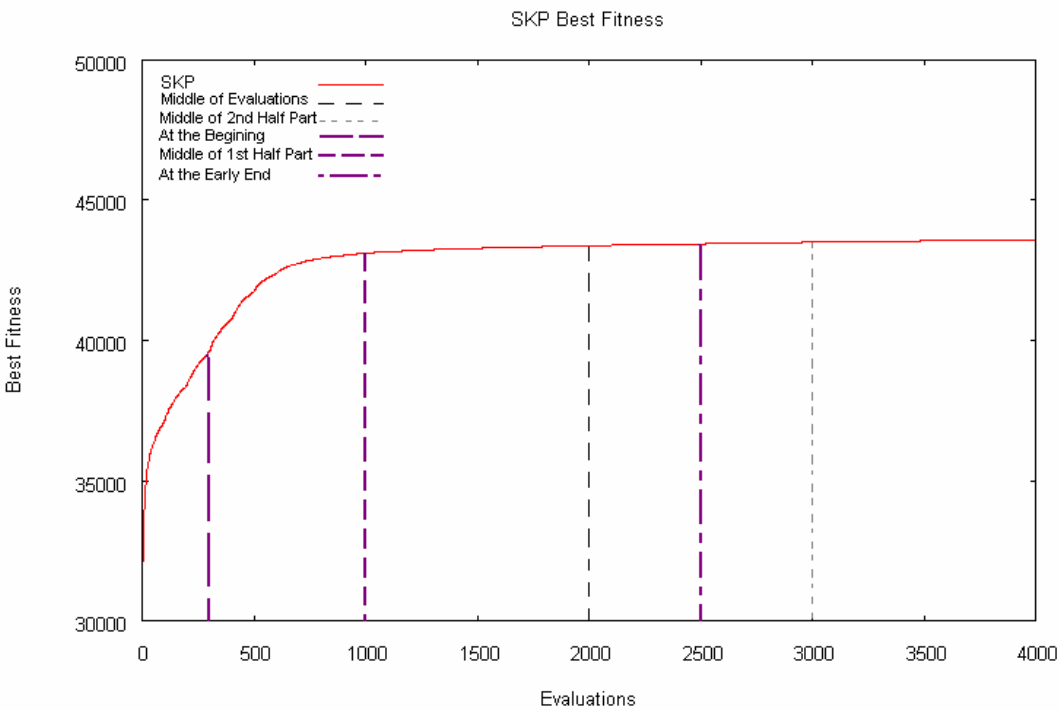

Figure 4.25 : SKP Best Fitness without Change

It is seen that period 3 in Figure 4.2, which visualizes the SKP performance in 4000 evaluations over the 1000 runs, coincides with the middle stage of the converging progress. It is observed that the period 10 and period 25 are in the nearly similar stage of converging progress. Thus, it is explanation for the similarity in results relevant to periods 10 and 25.

4.4.2 Additional Experiments on 2nd Method

The results of experiments on BMP have shown that 2nd Method's performance is relatively good. Also remember that the offset for change was 50 where change happened at 50th individuals' evaluation and it account for the middle of population. Here we have tested 2nd Method with offsets of 10 and 25. It means that environment has changed at 10th and 25th individuals' evaluation. Therefore the results are given in Table 4.7.

It can be seen that in the 2nd Method, performance of the method at offset 10 is worse than the performance of it at offset 25. Also performance of the method at offset 25 is worse than the performance of it at offset 50. Ordering, according to the value at Z evaluations after change, is the same in experiments with offset 25 and offset 50. However it is seen that the performance is of the 2nd Method in experiment with offset 10 is the relatively worst.

In order to look at diversity of population at different offsets in 2nd Method, Test 2 [Figure 4.14] is reapplied. Results are given in Table 4.8. It can be said that results are in an expected way that diversity of the least population where change happens at offset 10, is considerably least. Also, the diversity of population at offset 25 is much near the value of diversity of population at offset 50.

In summary, the decreasing performance of 2nd Method with offset 10 is clearly explained by the decreasing diversity of population.

Table 4.7 : Various Offsets on Method II

Bit Matching Problem									
Best Fitness Performance									
Method	II								
	Offset 10			Offset 25			Offset 50		
Periods For Change	3	10	25	3	10	25	3	10	25
Change Happened at Given Fitness Evaluation	310	1010	2510	325	1025	2525	350	1050	2550
Change Time + Z	610	1310	2810	625	1325	2825	650	1350	2850
Value at Z^{th} Evaluation after change	72.477	80.67	89.819	72.508	80.964	89.827	72.603	80.923	89.963
Value at 4000^{th} Evaluation	99.619	99.251	96.167	99.688	99.279	96.307	99.700	99.293	96.334
Std. Error of 1000 Runs at 4000^{th} evaluations	0.0161	0.0212	0.0330	0.0149	0.0212	0.0324	0.0148	0.0192	0.0319
Std. Error of 1000 Runs at Z^{th} evaluations after change	0.0744	0.0647	0.0418	0.0704	0.0624	0.0393	0.0659	0.0626	0.0381
Ordering according to value at Z^{th} evaluations after change	**1**	**3**	**4**	**1**	**2**	**3**	**1**	**2**	**3**

Table 4.8 : The Diversity of Population in Method II

Offsets / Periods	50	10	25
3	23.309	21.225	22.826
10	18.060	16.657	17.658
25	8.282	7.622	7.996

4.4.3 Experiments on higher severity of environmental changes

In experiments done so far, severity of environment changes was not high and it is concluded in unexpected good performance of 1st Method. Here we have increased the severity of environmental changes in BMP up to 0.4 from 0.1, and in SKP up to 0.1 from 0.05. Results are given in Table 4.11. In order to compare the performance of methods following tables 4.9 and 4.10 are formed where old and new performances according to ordering of the methods are shown separated by (/).

The compared results of BMP are shown in Table 4.9. It can be seen that performance of 4th Method remained the same, that it has still the worst performance. As it is expected, performance of 1st Method is getting worse, but still better than 2nd and 4th Methods. Performance of 3rd Method is getting better with large periods. The reason can be that at an early stage of run it looses time, but towards the end it can settle the deficiency. Performance of 2nd Method is getting much worse than before.

Therefore as a summary, if change in environment is severe, we have to use the 3rd Method, however if time is not available for reevaluation, the 1st Method can be preferred by compromising the performance.

Table 4.9 : Comparing Ordering Performance According to Severity in BMP

	Period 3	Period 10	Period 25
Method 1	2/1	1/2	1/2
Method 2	1/4	2/3	3/3
Method 3	4/2	3/1	2/1
Method 4	3/3	4/4	4/4

The compared results of SKP are shown in Table 4.9. Here it can be seen that the 1st Method has a bit decreasing performance, while the 3rd Method has an increasing performance. The 2nd and the 4th Methods' performances are the same as before. The important issue is that in spite of slight differences, performances of all methods are remaining the same. The reason for that is in period 10 and 25 SKP has almost similar converging performance [Figure 4.25].

Table 4.10: Comparing Ordering Performance According to Severity in SKP

	Period 3	Period 10	Period 25
Method 1	2/1	1/1	1/2
Method 2	1/2	3/3	3/3
Method 3	4/3	2/2	2/1
Method 4	3/4	4/4	4/4

Table 4.11 : BMP with Environment Severity of 0.4

Bit Matching Problem												
Best Fitness Performance												
Method	I			II			III			IV		
Periods For Change	3	10	25	3	10	25	3	10	25	3	10	25
Change Happened at Given Fitness Evaluation	350	1050	2550	350	1050	2550	350	1050	2550	400	1100	2600
Change Time + Z	650	1350	2850	650	1350	2850	650	1350	2850	650	1350	2850
Value at Z^{th} Evaluation after change	68.8	79.7	95.14	66.41	67.35	65.76	68.08	79.86	95.17	66.71	66.3	65.74
Order	1	2	2	4	3	3	2	1	1	3	4	4

Table 4.12 : SKP with Environment Severity of 0.1

Simple Knapsack Problem												
Best Fitness												
Method	I			II			III			IV		
Periods	3	10	25	3	10	25	3	10	25	3	10	25
Change Happened at Given Fitness Evaluation	350	1050	2550	350	1050	2550	350	1050	2550	400	1100	2600
Overall Best fitness before change	40280.031	43149.090	43450.980	40352.852	43146.070	43481.539	40305.180	43148.738	43464.031	40737.422	43168.879	43484.969
Standard Error at z	204.32	195.25	200.83	253.89	358.04	357.96	170.98	192.58	199.24	249.64	351.57	551.06
Order	1	1	2	2	3	3	3	2	1	4	4	4

4.4.4 Experiments on more environmental changes

Up to now the methods are experienced on the basis of one change in environment. Here the behaviors of methods are analyzed according to the 20 changes in the overall run. To clarify, in these experiments period 3 means change happens frequently at every 3 generations passed. Value of the best fitness at Z^{th} evaluations after last change is compared in all methods. Results are given in Table 4.15 and 4.16. In order to compare ordering performances of methods according to environment with one change and environment with 20 changes Table 4.13 and Table 4.14 separately for BMP and SKP are formed.

By looking at results it can be said that the 1^{st} Method has a good performance. Since the 3^{rd} Method has considerably better performance than others, performance of the 1^{st} Method seems getting worse. However, it has a good performance. While performance of the 2^{nd} Method is getting worse, performance of the 4^{th} Method is remaining the same, thus the worst.

In summary, in situations where changes are frequent and if time is available for reevaluation we can continue with the 3^{rd} Method after changes. In a limited time conditions 1^{st} Method can be preferred.

Table 4.13 : Comparing Ordering Performance According to Frequent Change in BMP

	Period 3	Period 10	Period 25
Method 1	2/1	1/2	1/2
Method 2	1/2	2/3	3/3
Method 3	4/4	3/1	2/1
Method 4	3/3	4/4	4/4

As for results shown in Table 4.14, it can be said that the 1^{st} Method has the best performance. The 3^{rd} Method has poor performance. The reason can be because of the time consuming reevaluation feature of 3^{rd} Method and special structure of SKP. The 2^{nd} Method has relatively the same performance. Due to the worsening of performance of the 3^{rd} Method, performance of 2^{nd} Method seems getting better. Performance of the 4^{th} Method is still the worst one.

In summary, dependent on the problem, if changes are frequent and severity of change is not high 1^{st} Method can suit best.

Table 4.14 : Comparing Ordering Performance According to Frequent Change in SKP

	Period 3	Period 10	Period 25
Method 1	2/1	1/1	1/1
Method 2	1/4	3/2	3/2
Method 3	4/2	2/3	2/3
Method 4	3/3	4/4	4/4

Table 4.15 : Bit Matching Problem with 20 changes

Bit Matching Problem 20 change												
Best Fitness Performance												
Method	I			II			III			IV		
Periods	3	10	25	3	10	25	3	10	25	3	10	25
1st Change Happened at Given Fitness Evaluation	350	1050	2550	350	1050	2550	350	1050	2550	400	1100	2600
Value at Z^{th} Evaluation after change	77.08	88.59	98.26	76.83	85.06	90.67	74.81	89.14	98.59	76.44	84.62	90.3
Std. Error of 1000 Runs at Z^{th} evaluations after change	0.1818	0.1577	0.0848	0.2301	0.1638	0.0779	0.2053	0.1450	0.0753	0.2100	0.1619	0.0759
Order	1	2	2	2	3	3	4	1	1	3	4	4

Table 4.16 : Simple Knapsack Problem with 20 changes

Simple Knapsack Problem 20 change												
Best Fitness												
Method	I			II			III			IV		
Periods	3	10	25	3	10	25	3	10	25	3	10	25
Overall Best Fitness at z evaluation	46441.61	48023.41	48440.6	45543.07	46559.31	47404.29	45923.08	46698.73	47122.26	45791.84	46456.41	47006.78
Standard Error at z	970.18	916.65	899.62	980.73	1043.28	921.701	1275.9	1348.417	1294.21	981.465	1049.42	028.0
Order	1	1	1	4	2	2	2	3	3	3	4	4

5 CONCLUSION

EAs are known as heuristic algorithms inspired from nature and for that reason they are suitable to the real world dynamic problems. It is known that in nature changes are happening in a stochastic manner. This has to be taken into account in the EAs design. However, almost all researches performed in this area, assumed that changes are happening between generations. Although this was a convenient assumption, it needs to be examined in a detailed way.

Thus, the aim of this thesis is to compare four methods of managing changes within generations and to do some empirical works on those methods.

Experiments have been performed on the Bit Matching and the Single Knapsack Problems where the former is a unimodal and the latter is a multimodal problem. The results derived according to empirical works provided interesting insights which can be used in design of more suitable algorithms according to the nature of the change. Thus, the results obtained from the experiments can be summarized as:

- If change with a small severity has happened, the 1st Method can be preferred as a variant for 3rd Method, where application is costly if changes happen frequently.

- On the other hand if period of change is in an early stage of the run, the 2nd Method can also be a variant for managing changes within a generation.

- In addition, the 4th Method is not a suitable way since it ignores the changes until the end of the generation in which they occur.

As a future work, real world problems can be analyzed where the severity is high and changes happen at different stages of the generation. Since this work is in progress, further results can enhance designing more suitable algorithms for real world problems where changes are happening in a stochastic manner.

REFERENCES

[1] **Grefenstette,J.J and Ramsey, C.L.,** 1993. Case-Based Initialization of Genetic Algorithms. *Proc. Fifth Intl. Conf. (ICGA93)*, San Mateo: Morgan Kaufmann, 84-91.

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

[3] **Branke, J. and Wang, W**. 2002. Theoretical Analysis Of Simple Evolution Strategies In Quickly Changing Environments. Technical Report 423, Institut AIFB, Karlsruhe, Germany.

[4] **Branke, J.,** 2002. Evolutionary Optimization in Dynamic Environments, Kluwer Academic.

[5] **Branke, J., and Schmeck, H.,** 2002 Designing Evolutionary Algorithms for Dynamic Optimization Problems. Springer-Verlag, Heidelberg, 239-262.

[6] **Richter, H., 2005.** A Study of Dynamic Severity in Chaotic Fitness Landscapes. *IEEE*, **5**, 2824-2831, http://ieeexplore.ieee.org/servlet/opac?punumber=10417

[7] **Jin, Y. and Branke, J.,** 2005. Evolutionary Optimization in Uncertain Environments - A Survey. IEEE Transactions on Evolutionary Computation, **9**(3), 303-312.

[8] **Heitkoetter, J. and Beasley,** D. eds. 2001. The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ), comp.ai.genetic. Available via anonymous FTP from rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic. (18.07.2007)

[9] **Branke, J.** 2007. *Personal communication*, Karlsruhe University, Germany.

[10] **Solomkina, J**. Issledovaniye Primenimosti Geneticheskih Algoritmov Dlya Optimizacii Neyrosetevyh Sistem, *M.Sc. Thesis*, www.doc.ic.ac.uk/~nd/surprise_96/ journal /vol4/tcw2/report.html

[11] **Miller, B. L. and D. E. Goldberg**. 1996. Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.,* **9**,193-212.

[12] **Smith, A.E. and Tate, D.M.,** Expected allele coverage and the Role of Mutation in Genetic Algorithms, *The Fifth International Conference on Genetic Algorithms*, University of Illinois, 31-37.

[13] **Goldberg** 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Professional. New York.

[14] **Phillip D**. 2001. Stroud Kalman-extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Evaluations. *Evolutionary Computation,* IEEE Transactions, **5** (1), 66-77.

[15] **Holland, J.H.,** 1973. Genetic Algorithms and the optimal allocation of trials. SICOMP, **2** (2), 88-105.

[16] **Grefenstette, J.J.,** 1992. Genetic Algorithms for changing environments, In R.Maenner and B.Manderick, Eds: Proceedings of Parallel Problem Solving From Nature (PPSN-2), Elsevier, Brussels, 137-144.

[17] **Schwefel, H.P.,** 1995. Evolution and Optimum Seeking. Wiley, New York.

[18] **Andersen, H.,** 1991. An investigation into Genetic Algorithms, And The Relationship Between Speciation And The Tracking Of Optima In Dynamic Functions. *Honours Thesis*, Queensland University of Technology, Brisbane.

[19] **Syswerda, Gilbert,** 1989. "Uniform Crossover in Genetic Algorithms." Morgan Kaufmann Publishers Inc., San Francisco.

[20] **Stephens, C. R. and Waelbroeck H.,** 1997. Effective Degrees Of Freedom In Genetic Algorithms And The Block Hypothesis. ICGA, Morgan-Kaufmann Publishers Inc., 34-41.

[21] **Blickle, T. and Thiele,L.** 1995. A Comparison of Selection Schemes used in Genetic Algorithm, *Computer Engineering and Communication*

Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), 2 Edition.

[22] **Salihoğlu, E.,** Uyar, S., Branke, J. 2005. Towards an Analysis of Dynamic Environments. *MSc Thesis*, ITU, Institute of Science and Technology.

[23] **Stanhope, S.A. and J.M. Daida,** 1999. (1+1) Genetic Algorithm Fitness Dynamics in a Changing Environment, in *CEC-99: Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway: IEEE Press., **3**, 1851-1858.

[24] **Droste, S.,** 2002**.** Analysis of the (1+1) EA For Dynamically Changing One Max-Variant, *LS Informatik 2*, Dortmund Univ., 55-60.

[26] **Arnold, D.V. and Beyer, H.G.,** 2000. Efficiency and Mutation Strength Adaptation of the (μ/ μi, λ)- ES in a Noisy Environment, in M. Schoenauer, K. Deb, G. Rudolph, et al., editors, *Parallel Problem Solving from Nature* - PPSN VI, Sixth Int'l Conf., Paris, France September 18-20, 2000, Proc., Springer, Berlin. 39-48.

[27] **Branke, J.,** 1999. Evolutionary Algorithms For Dynamic Optimization Problems-A Survey. *Technical Report 387*, Institute AIFB, University of Karlsruhe, February.

[28] **Branke, J.,** 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In Congress on Evolutionary Computation. *CEC99*, **3**, 1875-1882.

[29] **Cobb, H.G**., 1990.An Investigation Into The Use Of Hypermutation As An Adaptive Operator In Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments. Technical Report.AIZC-90-001, *Naval Research Laboratory*. Washington DC.

[30] **F. Vavak, K. Jukes, and T. C. Fogarty**. 1997. Adaptive Combustion Balancing In Multiple Burner Boiler Using A Genetic Algorithm With Variable Range Of Local Search. *International Conference on Genetic Algorithms*, Morgan Kaufmann, 719-726.

[31] **Mori, N., Kita, H., and Nishikawa, Y.,** 1998. Adaptation to a Changing Environment by Means of the Feedback Thermodynamical Genetic

Algorithm. *Proc. Parallel Problem Sol. From Nature.* Springer Berlin: Heidelberg. **1411**, 513-522.

[32] **Gottlieb, J.**, 1999. Algorithms for Constrained Optimization Problems. *Phd Thesis*, Technical University of Clausthal, Germany.

[33] **Pisinger, D.,** Knapsack Problems: Generation of Test Instances, online, http://www.diku.dk/~pisinger/codes.html (18.07.2007)

AUTOBIOGRAPHY

Gulshat Kulzhabayeva was born in Taraz, Kazakhstan in 1977. She was graduated from Talgar Kazakh-Turkish Technical High School. She gained 1st place at the Pascal Olympiad, in Talgar region in 1996. In 1997, she was ranked first at graduation from High school. She was accepted to Middle East Technical University in 1997 and was graduated in 2002. After graduation from the university, in 2003 she entered the graduate programme in computer engineering department offered by Istanbul Technical University. This work is her graduate thesis.