İSTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTUTUTE

TOWARDS AN ANALYSIS OF DYNAMIC ENVIRONMENTS

M. Sc. Thesis by
Erdem Salihoğlu

(704031006)

Date of submission : 9 May 2005

Date of defence examination: 2 June 2005

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

Co-Supervisor Dr. Jürgen BRANKE (KÜ.)

Members of the Examining Committee Prof.Dr. Muhittin GÖKMEN (İTÜ.)

Prof.Dr. Hasan DAĞ (İTÜ.)

Asst.Prof.Dr. Şule ÖĞÜDÜCÜ (İTÜ.)

JUNE 2005

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

DİNAMİK ORTAMLARIN ANALİZİNE DOĞRU

YÜKSEK LİSANS TEZİ

Erdem SALİHOĞLU
(704031006)

Tezin Enstitüye Verildiği Tarih : 9 Mayıs 2005
Tezin Savunulduğu Tarih : 2 Haziran 2005

Tez Danışmanı :	Yrd. Doç. Dr. Şima UYAR
Tez Eşdanışmanı:	Dr. Jürgen BRANKE (KÜ.)
Diğer Jüri Üyeleri	Prof.Dr. Muhittin GÖKMEN (İTÜ.)
	Prof.Dr. Hasan DAĞ (İTÜ.)
	Yrd.Doç.Dr. Şule ÖĞÜDÜCÜ (İTÜ.)

HAZİRAN 2005

ACKNOWLEDGEMENTS

Firstly I want to thank Asst. Prof. Dr. Şima Uyar, for refereeing this thesis. I am deeply grateful to her for sparing a lot of her time for me, giving me advises not only for my thesis but also for my career and studies, teaching me so much about evolutionary computation, and encouraging my studies by her enthusiasm for science.

I wish to thank Dr. Juergen Branke for giving me the opportunity to visit Karlsruhe and to join this project. He has helped me so much in starting and finishing this thesis by his great experience and vision in the area.

Finally I remain deeply indebted to my parents, who have always supported me, both financially and mentally, for standing alongside with me my entire life, for encouraging me for my higher education by their own enthusiasm, and most importanly for dedicating their lives to me, my brother and my sister.

MAY 2005 Erdem SALİHOĞLU

INDEX

ABBREVIATIONS

EA	: Evolutionary Algorithm
LHC	: Local Hill Climbing
MKP	: Multidimensional Knapsack Problem
MPB	: Moving Peaks Benchmark

TABLE LIST

FIGURE LIST

ÖZET

Evrimsel algoritmalar Mendel genetiğine ve Darvin'in "en uygun olanın yaşaması" prensibine dayanan hesaplama yöntemleridirler. Birçok problemin bilgisayarlarla çözülmesi için, doğal evrim sürecinin simülasyonudurlar. Evrimsel algoritmalar alanındaki akademik çalışmalar daha çok statik optimizasyon problemleri üzerinde yoğunlaşmıştır. Fakat birçok dinamik gerçek dünya problemi vardır.

Evrimsel algoritmalar temel olarak kara-kutu optimizasyon araçları olarak tanımlanırlar. Evrimsel algoritmaları daha iyi anlayabilmek için evrimsel aramanın dinamiklerinin analizi gerekmektedir. Bu sayede daha iyi evrimsel algoritmalar tasarlanabilir. Statik optimizasyon problemlerinin analizi konusunda bir miktar çalışma bulunmaktadır. Bunula birlikte dinamik ortamların analizi konusundaki çalışmalar daha başlangıç aşamasındadır.

Dinamik ortamların analizi sayesinde, dinamik ortamların doğası hakkında yeni bilgiler edinilebilir. Bu bilgiler 3 temel yönde yararlı olabilirler.

- Değişimin doğasının anlaşılması sayesinde, elde edilen bu bilgileri kullanan daha iyi algoritmalar yazılabilir.

- Dinamik problemlerin gürbüz bir sınıflandırması yapılabilir. Bunun yanında problem sınıflarından algoritma tiplerine eşleşmeler yapılabilir, bu sayede dinamik problemler için algoritmalar daha kolay tasarlanabilir ve problemler daha kolay çözülebilir.

- Gerçek dünya problemlerini temsil eden yapay test problemi üreteçleri geliştirilebilir.

Bu tez çalışmasının ana amacı evrimsel algoritmalar gözüyle dinamik ortamların analizine doğru ilk adımı oluşturmaktır. Bu amaçla, dinamik ortamların analizi için belirli sayıda ölçüt önerilmiştir. Bu tez çalışmasının ana konsepti önerilen ölçütlerdir. Bu ölçütleri kullanarak, bir gerçek dünya problemi (çok boyutlu sırt çantası problemi) üzerinde ve de bir yapay test problemi üreteci (dolaşan tepeler yapay üreteci) üzerinde örnek analizler yapılmıştır.

SUMMARY

Evolutionary algorithms are computational methods based on the concepts of Mendelian genetics and Darwin's principle of "survival of the fittest". It is the simulation of natural evolution for solving a wide range of problems by computers. The academic research on EAs is mostly concentrated on static optimization problems. However there are many dynamic real-world problems.

EAs are used to be considered mainly as black-box optimization tools. To have a better understanding of EAs, which can help us design better EAs, analysis of the dynamics of the evolutionary search is needed. There has been some previous work on the analysis of the static optimization problems. However the work on the analysis of dynamic environments is currently in its very early stages.

By analyzing dynamic environments, new insights can be gained about the nature of the dynamic environments, which can be useful mainly in three ways:

- By understanding the nature of a change, better algorithms can be designed which can exploit this information.

- It can lead us to make a robust classification of the dynamic problems. Furthermore a mapping from a class of problems to an algorithm type can be constructed which will make it easier to design algorithms for dynamic problems and solve them.

- Artificial benchmark problems representing real-world problems can be generated.

The main aim of this thesis is to make a first step towards the analysis of dynamic environments from the point of view of an EA. For this purpose we propose a number of measures to analyze dynamic environments. The key concepts of this thesis are the proposed measures. Using these measures, we make an example analysis on a real-world problem (a multidimensional knapsack problem) and a benchmark problem (moving peaks benchmark).

1- INTRODUCTION

Evolutionary Algorithms (EAs) are computational methods based on the concepts of Mendelian genetics and Darwin's principle of "survival of the fittest". It is the simulation of natural evolution for solving a wide range of problems by computers. One of the main application areas of EAs are optimization problems.

The academic research on EAs is mostly concentrated on static optimization problems. However there are many dynamic real-world problems such as: load balancing of a distributed computer system, where processes arriving over time has to be distributed to the processors, or scheduling problems, where new jobs can arrive and some machines can break down over time [3].

EAs are used to be considered mainly as black-box optimization tools. To have a better understanding of EAs, which can help us design better EAs, analysis of the dynamics of the evolutionary search is needed. Fitness landscape is an important concept investigated for this purpose. Fitness landscapes are characterized by three main components; search space, fitness function and an operator [17].

There has been some previous work on fitness landscapes of static problems e.g., [36], [34], [33]. Ruggedness, neutrality and smoothness are some of the proposed concepts to investigate the characteristics of the fitness landscapes of static environments.

The work on the analysis of dynamic environments is currently in its very early stages.

By analyzing dynamic environments, new insights can be gained about the nature of the dynamic environments, which can be useful mainly in three ways:

- By understanding the nature of a change, better algorithms can be designed which can exploit this information.

- It can lead us to make a robust classification of the dynamic problems. Furthermore a mapping from a class of problems to an algorithm type can be

constructed which will make it easier to design algorithms for dynamic problems and solve them.

- Artificial benchmark problems representing real-world problems can be generated.

The main aim of this thesis is to make a first step towards the analysis of dynamic environments from the point of view of an EA. For this purpose, we propose a number of measures to analyze dynamic environments. Using these measures, we analyze a real-world problem (a multidimensional knapsack problem) and a benchmark problem (moving peaks benchmark).

The structure of the thesis is as follows:

Chapter 2 gives introductory information for the thesis. Overview of EAs, previous work on EAs for dynamic optimization problems, and the focus of the thesis is given in this chapter.

In Chapter 3, the proposed measures to quantitatively analyze, characterize and categorize dynamic environments are introduced.

In Chapter 4, information about knapsack problems, and a special case of knapsack problems, i.e. Multidimensional Knapsack Problems (MKP), are given. Then, EA approaches to multidimensional knapsack problems are investigated. At the end of the chapter, the benchmark problem generator for dynamic multidimensional knapsack problems, which is used in this thesis, is introduced.

Chapter 5 lists the results of the experiments on a number of MKP instances and Moving Peaks Benchmark (MPB), which are done to see the results of the proposed measures.

Chapter 6 concludes the thesis and provides ideas for future work.

2- BACKGROUND INFORMATION

This chapter aims at giving introductory information for the thesis. Section 2.1 gives a general explanation of EAs, and section 2.2 investigates the previous research on the EAs for dynamic environments and concludes with the focus of the thesis.

2.1 General Explanation of EAs

In this section we overview the EAs with a brief history, explain some fundamental concepts of EAs, and conclude with a section giving information about the major application areas of EAs.

2.1.1 Overview and a Brief History

Evolutionary algorithms are computational methods based on the concepts of Mendelian genetics and Darwin's principle of "survival of the fittest". It is the simulation of natural evolution for solving a wide range of problems by computers.

There are four different streams of evolutionary algorithms: evolutionary programming, genetic algorithms, evolution strategies and genetic programming, all of which are inspired from the same philosophy.

Evolutionary programming was invented by Fogel, Owens and Walsh [10]. Before 1990 it was mostly used for prediction tasks, whereas now parameter optimization is the main focus of the evolutionary programming [9].

Genetic algorithms were introduced by Holland. The idea was simulating the adaptation in nature on the computers [16]. Genetic algorithms are mostly used as an optimization tool.

Among the other two main streams of evolutionary algorithms, evolutionary strategies were developed by Rechenberg [29] and Schwefel [32] and is now mainly used in continuous parameter optimization. Lastly, genetic programming was invented by Koza [19], [20], which is mostly used in machine learning applications.

2.1.2 General Scheme of an Evolutionary Algorithm

Even though there are minor differences in every branch of evolutionary algorithms, they actually all use the same algorithm. The general scheme of an evolutionary algorithm is given as a pseudo code below and as a diagram in Figure2.1:

```
generate initial population
calculate fitness of each individual
while not STOP CRITERIA do
        parent selection
        recombine pairs of parents
        mutate offspring
        calculate fitness of offspring
        survivor selection
end_while
```

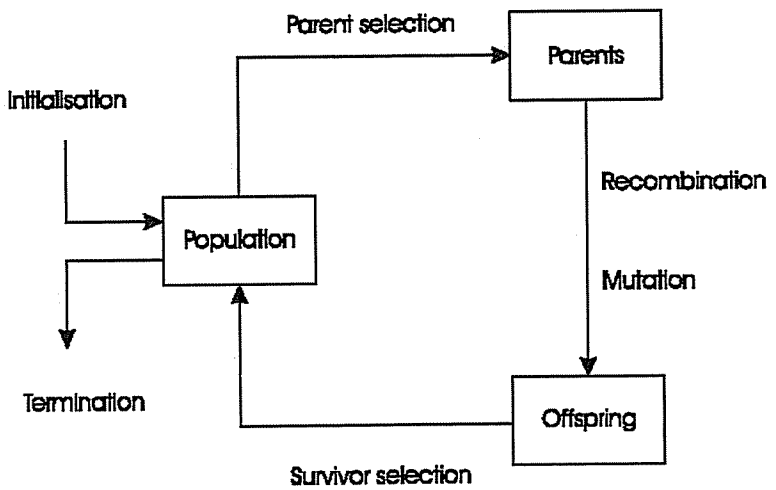


Figure 2.1: General scheme of an evolutionary algorithm [9]

There are 5 main components of EAs. Representation of the individuals, fitness function used, evolving population, selection operators (parent and survivor) and variation operators (recombination and mutation). They are briefly explained below.

- representation: In approaching a problem with EAs, the first fundamental design decision is the representation of the solutions in the EA domain. The solutions in the natural form are called the phenotype and the encodings of them in the EA domain

are called genotypes. Phenotype space and the genotype space can either be identical or distinct. The evolutionary search takes place at the genotype space.

- fitness function: Fitness function is mainly used in parent and survivor selection to compare individuals' relative performances. To obtain the fitness of an individual, first the genotype of the individual is mapped to the corresponding phenotype and then the fitness of the individual is evaluated by the objective function with using the phenotype.

- population: In every EA, a number of individuals are created at the beginning of the algorithm which forms a population. The population evolves in every iteration of the loop until the stopping criteria are met.

- parent selection: In every iteration of the evolution loop, parents are selected according to their fitness values. Generally, the higher the fitness of an individual is, the higher is its chance to be a parent for recombination.

- recombination and mutation: Recombination and mutation are known as variation operators. Both of the operators are stochastic, they are applied with probabilistic rules. In recombination every pair of parents are mated by crossover. Then their offspring are mutated by making random changes in the genotypes. This process in the end generates individuals with different genotypes.

- survivor selection: At the end of each loop we have the new offspring and the old population at hand. The algorithm lastly goes through a selection phase to create a population with higher quality [9].

2.1.3 Major Application Areas of Evolutionary Algorithms

EAs have a wide range of practical applications. Some of them are listed below:

- Optimization

- Machine learning

- Evolutionary art

- Economics

- Ecology

- Prediction

- Engineering design

- Automatic programming

- Population Genetics

- Social systems [35], [23]

2.2- EAs for Dynamic Optimization Problems

This section summarizes the previous work about the EAs for dynamic environments, which are relevant to the focus of this thesis. At the end of the section some open issues and the focus of the thesis is stated.

2.2.1- Classification of EA Approaches to Dynamic Optimization Problems

Many different methods are proposed to solve dynamic optimization problems by evolutionary algorithms. Branke classified the proposed approaches into three main categories [3].

The first type of approaches concentrates on obtaining the necessary diversity for tracking the optimum by reacting to changes in the environment. The most straightforward approach is simply restarting the EA with a totally new random population. This means solving the problem from scratch. A more specific example for this class of approaches is hypermutation [7]. In this approach when a change is detected, the mutation probability is increased for a number of generations to increase the diversity of the population.

The second group of approaches concentrates on avoiding convergence throughout the run. An example of this class is random immigrants [14]. The diversity of the population is kept by replacing a number of individuals with randomly generated ones in every generation.

The last class of approaches concentrates on using a memory. Some approaches use implicit memory while others use explicit memory. An example of implicit memory approach is diploidy mechanism, where a diploid chromosome structure is used. It is shown that diploidy is good for environments alternating between two states [21]. The main idea of using explicit memory is keeping good individuals to use in prospective generations, which are assumed to be useful if the optimum periodically

returns to the previous locations. Some examples of this approach can be found in [22] and [28].

2.2.2- Categorization of Dynamic Environments

There is some previous work on the categorization of dynamic environments. De Jong categorized the landscapes as follows: [8]

- drifting: There are small changes in the structure of these landscapes over time.

- landscapes that have significant morphological changes: These landscapes change significantly over time.

- cycling: These landscapes alternate between a few number of states.

- abrupt and discontinuous: These landscapes go through unexpected changes.

Branke proposed the following criteria to characterize the dynamic environments [3], [5]:

- frequency of change: is the time passed between change instances. Mostly the number of fitness evaluations is used to determine this criteria.

- severity of change: is the strongness of the changes. Usually the distance between the old and the new optimum is used to measure this criterion.

- predictability of change: determines if there is a trend in the changes.

- cycle length/ cycle accuracy: is used for the environments that periodically return to similar states. Cycle length is the time passed between similar states, and the cycle accuracy is the measure of similarity of these states.

Weicker developed a mathematical framework to classify the dynamic optimization problems and proposed a classification that is a combination of De Jong's and Branke's classifications [37].

Within this framework a fitness function is constructed by keeping the maximum of several functions for each point of the search space. In every change instance the functions can take the following actions: coordinate transformation, fitness rescaling and, stretching. He defines these concepts mathematically and proposes a classification based on them.

2.2.3- Moving Peaks Benchmark Problem

Several benchmark problems simulating dynamic environments are proposed to test EAs. Some are: dynamic bit matching problem (e.g. [35]) where the aim is matching the bit string that is changing over time, moving parabola (e.g. [1]) where the fitness function is a parabola in n dimensions, job shop scheduling problem (e.g. [2]) where new jobs arrive over time.

A survey on this topic is available in [3]. We will concentrate on the moving peaks benchmark problem, which we will use in our experiments.

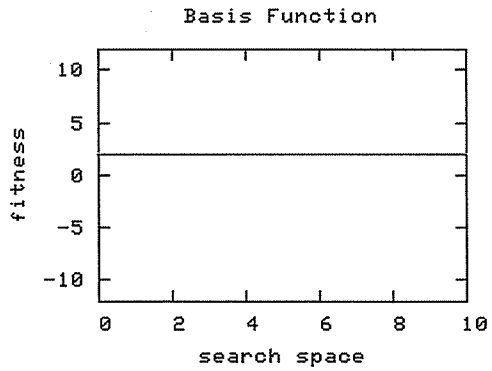
Moving peaks benchmark was developed by Branke in [4] and [3]. A similar benchmark was also proposed by Morrison and De Jong in [24] independently.

MPB is designed for free optimization problems in dynamic environments, in which the encoding is real-valued. The fitness landscape is composed of m peaks and a basis function in n dimensions. Formally the fitness landscape is formed according to the formula (2.1):

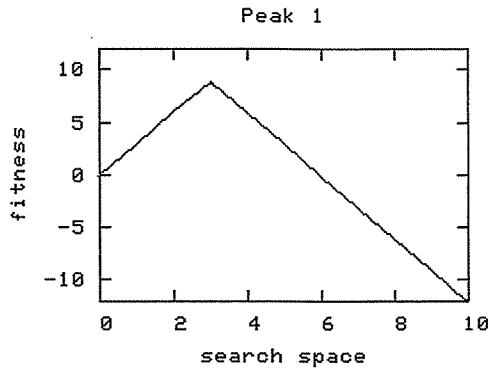
$$F(x, t) = \max(B(x), \max_{i=1...m} (P_i(x, h_i(t), w_i(t), p_i(t)))) \quad \textbf{(2.1)}$$

where B(x) is the basis function and P is the peak function. The parameters of P: x, $h_i(t)$, $w_i(t)$, and $p_i(t)$ are the coordinates of the point, height of the peak i at time t, width of the peak i at time t and the location of the center of the peak i at time t, respectively.

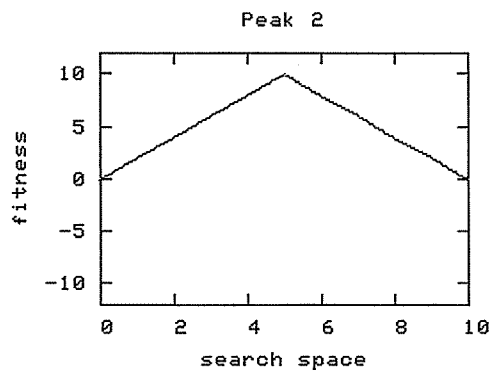
For a better understanding, the composition of a fitness landscape with two peaks of cone shape and a basis function B(x) = 2 in 1 dimensional search space is given graphically in Figure 2.2, below.



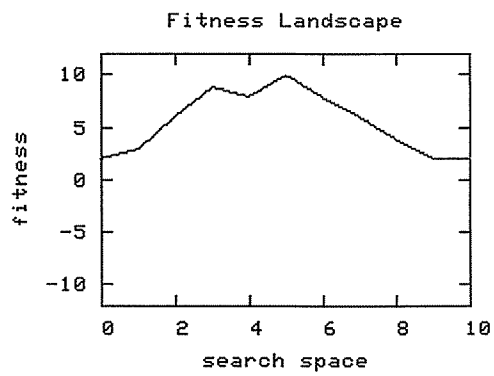
a) Basis function



b) Peak function 1



c) Peak function 2



d) Fitness Landscape

Figure 2.2: Moving Peaks Benchmark

In every change instance, the height, width and the location of every peak may change, while the basis function remains unchanged. The height and width of peaks are changed by adding a Gaussian random variable. The locations of the peaks are controlled by a parameter lambda, which determines the predictability of the movement of the peaks. When lambda is 0 every peak moves randomly and when it is 1 every peak moves in the direction that it has moved in the previous time instance. Therefore the higher the lambda is, the more predictable the movement of the peaks are.

2.2.4- Previous Work on the Analysis of Dynamic Environments

EAs are mostly known to be black-box optimization tools. To have a better understanding of the EAs, which can help us design better EAs, analysis of the

dynamics of the evolutionary search is needed. Fitness landscape is an important concept investigated for this purpose. Fitness landscapes are characterized by three main components; search space, fitness function and an operator [17].

There has been some previous work on fitness landscapes of static problems e.g., [36], [34], [33]. Ruggedness, neutrality and smoothness are some of the proposed concepts to investigate the characteristics of the fitness landscapes of static environments.

The work on the analysis of the dynamic environments is currently in its very early stages. There has been some recent work on the behavior of EAs on dynamic landscapes which can give insights about the dynamics of the fitness landscapes. Morrison investigated the effects of changing the landscape period on the performance of triggered hypermutation [25] and Richter explored the behavior of an EA on chaotically changing fitness landscapes and showed that the tracking capability of the EA decreases with the decrease of the predictability of the changes [30].

3 – MEASURES TO ANALYSE DYNAMIC ENVIRONMENTS

In this chapter we introduce the measures that we have proposed in [6] to quantitatively analyze, characterize and categorize different dynamic fitness landscapes. The main aim of these measures is to capture relevant information regarding the dynamics of the changes.

3.1- Change Severity

Change Severity has been previously defined as the distance between the global optima before and after the change. We also believe this is an important measure and include it in our list of measures. For reasons of comparison we normalize every distance by the maximum of all possible distances. Depending on the magnitude of the severity, different strategies can be chosen to better adapt to the environment.

3.2- Estimated Change Severity

In order to apply the change severity measure, it is necessary to know the global optimum. However often times, this will not be the case. To tackle this problem, we propose estimating the optimum in some reasonable way. In this thesis, our particular approach to this estimation process is using the best sample each time to calculate the change severity.

3.3- Fitness Correlation

We calculate the correlation between the fitness of the samples before and after the change. High correlation implies that the good solutions before the change are likely to remain good in the changed environment and the converse holds for the bad solutions.

3.4- Local Hill Climbing Fitness Correlation

Although we do random sampling in analyzing our search spaces, we implicitly assume that at the worst case, our algorithms will do local hill-climbing (LHC). Here we are proposing to measure the fitness correlation after LHC prior to and after the change. If there is a high correlation between the local optima before and after the change, then it is likely that simple hill-climbing after the change will yield good solutions.

3.5- Fitness Change Correlation of Similar Points

Here we measure the correlation of the changes in fitness of points that are a fixed distance, d, apart from each other. Then we repeat this process for different distances. We expect to see that correlation decreases with increasing d. High correlation implies that points which are within a fixed distance, change similarly and vice versa.

3.6- Estimated Value of Last-Stage Local Optima

Here we measure the benefit of using good solutions from the last stage. In order to do so, we compare the fitness we obtain by LHC from random points with last stage's best k solutions. Since, it is almost impossible to calculate last stage's best k local optima, we use the best k sample points as estimates of the local optima.

3.7- Value of Past Optima

It is usually assumed that keeping information regarding the global opima of previous stages is beneficial in cyclic environments. To test this assumption, we compare the fitness we obtain by LHC from random points with k previous global optima.

3.8- Estimated Value of Past Optima

In some cases it is not possible or practical to find the global optima. In these cases, we use the point obtained by LHC from the best sample as an estimate for the global optima. Then we calculate the above measure using this estimate.

4- DYNAMIC MULTIDIMENSIONAL KNAPSACK PROBLEM

In this chapter, firstly information about knapsack problems is given. Secondly a special case of these types of problems, multidimensional knapsack problems, are explained. After that EA approaches to multidimensional knapsack problems are explored and two successful approaches, Penalty_9 and Weight Coded, which we have used in our experiments are explained. The dynamic multidimensional knapsack problem generator we use in this thesis was proposed in [6] and is introduced at the end of this chapter.

4.1 Multidimensional Knapsack Problem

Knapsack problems can be formulated as follows:

$$\max \sum_{j=1}^{n} pj * xj$$

$$\text{subject to } \sum_{j=1}^{n} wj * xj \leq c$$

$$xj \in \{0,1\}, j = 1,\ldots,n \qquad (4.1)$$

where n is the number of items, p_j and w_j are the profits and weights of the item , respectively, c is the capacity of the knapsack, xj is 1 if the item is packed into the knapsack and 0 if the item is not packed into the knapsack [12].

A more intuitive explanation is as follows: Consider a mountaineer who will take a trip to a mountain. He has n items all of which may give him comfort, and has a volume. He has to choose a number of items to take with him in his bag while maximizing the comfort and not exceeding the capacity of the bag [18].

There are many applications of knapsack problems and its variations both in academic researches and real-world problems. The investment problem is an illustrative example of a real-world example. It is defined as follows: There is an investor with c amount of money which he wants to put into some businesses. There

are n different investments all of which requires w_j amount of money and gives p_j amount of profit. The aim is choosing a number of investments such that the overall profit is maximized while not exceeding the amount of money. There has been a lot of work in academic research related to knapsack problems.

There are many variants of knapsack problems such as: subset sum problem, bounded and unbounded knapsack problems, multidimensional knapsack problem, multiple knapsack problem etc. [18]

In recent years multidimensional knapsack problem is used as a test problem for Evolutionary Algorithms in many studies, because of its difficulty.

The multidimensional knapsack problem is defined as follows:

$$\max \sum_{j=1}^{n} pj * xj$$

$$\text{subject to } \sum_{j=1}^{n} wij * xj \leq c, \qquad i = 1,\ldots,n$$

$$xj \in \{0,1\}, \qquad j = 1,\ldots,n. \tag{4.2}$$

where p_j's are profits, w_{ij}'s are weights and c_i's are capacities of the problem [13].

One of the important concepts about MKP is the tightness ratio. The tightness ratio of a given knapsack is defined as the ratio of the capacity of that knapsack to the sum of the weights of the items for that knapsack. The minimum, maximum and average tightness ratios of a MKP problem is the minimum of all the tightness ratios, maximum of all tightness ratios and average of all of the tightness ratios, respectively for that problem. Tightness ratio is the heuristic estimation of the feasibility ratio for a given MKP instance. Mostly minimum tightness ratio is expected to be the strongest indicator of the feasibility of the given MKP problem instance [12].

4.2 Evolutionary Algorithms for Multidimensional Knapsack Problems

MKP is a constrained optimization problem. Gottlieb [12] classified the proposed constraint handling techniques into three categories:

4.2.1- Direct Search in the Complete Search Space

In this approach, constrained optimization problem is converted to a free optimization problem. Generally penalty functions are used for this purpose. The fitness function of the MKP f(x) is converted to Equation 4.3 where the penalty term is formulated as given in Equation 4.4:

$$f(x) - penalty(x) \quad \textbf{(4.3)}$$

$$penalty(x) = 0 \text{ if } x \text{ is feasible,}$$

$$penalty(x) > 0 \text{ if } x \text{ is unfeasible} \quad \textbf{(4.4)}$$

4.2.2- Direct Search in the Feasible Search Space

In this approach search takes place in the feasible region of the search space. This is achieved in two ways. First way is using a repair operator. Classical operators are used (mutation, crossover) in the evolutionary search. However when unfeasible solutions are generated, they are repaired by the repair operator. The repair operator can be formulated as follows:

$$repair(x) = x, \text{ if } x \text{ is feasible,}$$

$$repair(x) = y(\text{feasible}), \text{ if } x \text{ is unfeasible} \quad \textbf{(4.5)}$$

so the fitness function is defined as follows:

$$f(repair(x)) \quad \textbf{(4.6)}$$

The second method is using specialized operators (mutation, crossover) that are preserving the feasibility of the solutions. By this way the search again takes place in the feasible search space.

4.2.3- Indirect Search in the Feasible Search Space

Evolutionary Algorithms work on an arbitrary search space (T) with ordinary operators. The fitness of each individual is evaluated by the following way:

$$f(decode(u)) \quad \textbf{(4.7)}$$

where u is the solution in the search space T and f is the objective function of the constrained optimization problem. The decoder function always maps solutions from T to the feasible region of the search space.

4.3 – Different Representations for the Multidimensional Knapsack Problems

4.3.1- Binary Representation

Mostly direct search in the complete search space and direct search in the feasible search space based methods use a binary representation. Penalty based methods uses the complete search space while repair operator based methods use only the feasible region of the search space. For our purpose of analyzing the landscapes we decided to use a penalty based approach from the binary represented approaches.

In [12] it is stated that the main problem of penalty based methods is the feasibility problem that is, the search does not concentrate on the feasible regions and the penalty function does not guide the population to the feasible boundary of the search space. Another problem is that most of the techniques are affected very much from the initialization method used.

Gottlieb [12] introduced the monotony concepts and stated that penalty functions satisfying the strongest monotony concept will not suffer from initialization technique and feasibility problem. A new penalty function, Penalty_9, is proposed in [12] satisfying the most severe monotony concepts.

Penalty_9 is calculated as follows:

$$\text{Penalty\_9} = ((p_{max} + 1) / w_{min}) * \max \{CV(x, i) \mid i \in I\}$$

$$p_{max} = \max\{p_j \mid j \in J\}, J = 1, \ldots, n$$

$$w_{min} = \min\{w_{ij} \mid i \in I, j \in J\}, I = 1, \ldots, m$$

$$CV(x, i) = \max(0, \sum_{j \in J} w_{ij} * x_j - c_i) \qquad \textbf{(4.8)}$$

where $w_{ij} > 0$, p_{max} is the maximum profit, w_{min} is the minimum resource consumption, and CV(x, i) is the maximum constraint violation, n in the number of items and m is the number of knapsacks.

In the empirical analysis in [12] it is shown that Penalty_9 is guiding the population towards the feasible boundary and does not suffer from initialization routine used.

4.3.2- Real-Valued Representation

The weight-coded approach for MKP is the best of the proposed indirect search in the feasible search space methods [12]. In this approach chromosomes are real-valued genes (weights). The evolutionary search takes place in the real-valued space, so the ordinary operators (crossover, mutation) for real-valued EAs can be used. To obtain the phenotype, first the problem is biased by the weights, and then the decoding heuristic is applied to the biased problem. Feasibility of the problems are guaranteed by the decoder operator [27].

Four different biasing techniques are proposed in [27]. Even though the technique based on multiplication of profits with log normally distributed weights approach has small advantages, it is stated that all of them work well over a range of biasing strengths which is a strategy parameter. We used the biasing method based on multiplication of profits with logarithmically distributed weights because with this method genes have certain bounds which was needed in our work for deciding the step size for local hill climbing. The biasing method used is as follow:

$$p_j' = p_j * w_j,\ \ w_j = (1+\gamma)^{R(-1,1)} \qquad \textbf{(4.9)}$$

Profits of the original problem are multiplied by weights and then a decoding heuristic is applied to the biased MKP instance.

Raidl [27] compared two decoding heuristics and noted that surrogate relaxation based method is superior to the Lagrangian relaxation based method because the first one needs less computational effort and has a slightly better performance.

In this approach MKP is transformed to a single constraint knapsack problem by using surrogate multipliers.

Pirkul [26] proposed several methods to obtain surrogate multipliers. One of them is solving linear programming relaxed MKP in which every variable can take values in the range 0 – 1, and the dual variables are used as surrogate multipliers.

We used the heuristic given in [27]. After the conversion of the problem to the single constraint problem using Pirkul's method, the items are sorted in decreasing order of

profit / pseudo-resource consumption ratios. Lastly the items are packed into the knapsacks in the sorted order. If an item violates a consumption ratio, it is not packed into the knapsacks. The pseudo code is given below:

The Algorithm of Surrogate Relaxation Based Method:

At the beginning of the run:

- determine surrogate multipliers for the original problem

- evaluate pseudo-resource consumption ratios

In each evaluation of an individual:

- evaluate profit/pseudo-resource consumption ratios

- sort profit/pseudo-resource consumption ratios in decreasing order

- pack the items to the knapsacks in the sorted order, if an item does not violate any of the constraints, then pack the item into the knapsacks.

To keep computational effort less, the surrogate multipliers are evaluated only once for the original problem.

4.4- The Benchmark Dynamic MKP

In the dynamic MKP we implemented for this study; profits, weights and capacities are changing in each change instance. At the beginning of the run for each profit, weight and capacity, a lower and upper bound is determined according to the formula (4.5):

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

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

lower bound of $w_{ij} = w_{ij} * (0.8)$ for each resource consumption

upper bound of $w_{ij} = w_{ij} * (1.2)$

lower bound of $c_i = c_i * (0.8)$ for each capacity

upper bound of $c_i = c_i * (1.2)$ **(4.10)**

At each change instance, each of the profits, weights and capacities are changed multiplicatively as follows:

$p_j = p_j * (1 + N(0, 0.05))$; for each profit

$w_{ij} = w_{ij} * (1 + N(0, 0.05))$; for each resource consumption

$c_i = c_i * (1 + N(0, 0.05))$; for each capacity **(4.11)**

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

If any of the numbers generated with the above formulae is outside the bounds of that variable, than it is bounced back from the bounds until it is inside the bounds.

5- EXPERIMENTS

In this chapter we list the experiments which we have done to see the results of the proposed measures. Tests are done with a number of MKP instances and MPB.

Stratified sampling is used, when generating sample points, to improve the estimates. In some cases local hill climbing is needed. We used steepest ascent hill climbing. For binary representation (in Penalty_9 approaches) bit-flip neighborhood and for real-valued representations (in Weight-Coded and Moving Peaks approaches) 1/5 of the gene's search space range is used as a fixed step size. For binary representation each of the neighbors with hamming distance 1 is evaluated in local hill climbing. In real-valued representation, for each gene of the chomosome the step side is added and substracted to the gene value and two neighbors are constructed, and these neighbors are used in local hill climbing.

For generating random numbers, GNU Scientific Library is used [15]. Optimum and linear relaxed solutions of the MKP instances are calculated by GNU Linear Programming Kit [11].

The MKP data sets used in the tests are: PB5, HP1, WEING03, and HARD1. The first three of the data sets are real-world problems, and the last one is a generated hard problem from Chu and Beasley's benchmark suite. All of the data sets are taken from the Operations Research Library [31].

In the PB5 data set, the number of items is 20, the number of knapsacks is 10, minimum, average and maximum tightness ratios are 0.44, 0.54 and 0.61, respectively.

In HP1 data set, the number of items is 28, the number of knapsacks is 4 and minimum, average and maximum tightness ratios are:0.55, 0.58 and 0.60, respectively.

WEISH02 data set includes 5 knapsacks and 30 items. The minimum, average and maximum tightness ratios are: 0.31, 0.50, 0.71, respectively.

In HARD1 data set, the number of items is 100, the number of knapsacks is 5 and all knapsacks have a tightness ratio of 0.25.

In the tests with MKP instances, the number of samples for Penalty_9 approach is more than the number of samples for Weight-Coded approach because Weight-Coded approach covers only the feasible region which is a subset of the whole search space covered by the Penalty_9 technique .

In all the experiments with Weight-Coded approach, 1 is used for the biasing strength.

In the experiments with MPB, we used different parameter settings to test different aspects of that bencmark suite. These experiments don't examine the benchmark suite comprehensively, since this is only a case study.

5.1- Experiments for Change Severity

Change severity is known to be an important measure in dynamic environments, and usually defined as the distance between the global optima before and after the change. We also agree on the idea that this is an important measure and include it in our list of measures. In some cases it is not practical or possible to find the global optimum. In these cases, we use the best sample found as an estimator for the global optimum.

5.1.1- Details of Experiments

This measure involves a distance between solutions; we used Hamming distance for binary representation and Euclidean distance for real-valued representation. For reasons of comparison we normalize every distance by the maximum of all distances. To apply this measure, the global optimum needs to be known. However, in some cases this is not possible or practical. In these cases, we use the best sample in each time instance as an estimate for this measure.

When the global optimum is needed, and it is practical to find it, we only find one of the global optima.

We used 32768 samples for Penalty_9 approach, and 5000 samples for Weight-Coded approach when calculating the measures for data sets PB5 and HARD1. In

each of the three experiments of MPB we used 2000 sample points. The default parameters of MPB are listed in Table 5.1

For each data set and representation for MKP instances, and for MPB instances the experiments are ran 20 times with the same environment but different samples.

There are three parameters affecting change severity in MPB, namely vlength, width_severity and height_severity. We did three tests with small, medium and large changes in the environment. In small change severity, the values of the parameters about severity are, vlenght = 1, height_severity = 1, and width_severity = 0.25. In medium change severity the values of these parameters are: vlength = 1.5, height_severity = 1.5, width_severity = 0.375 and in large change severity the values of these parameters are as follows: vlength = 2, height_severity = 2 and width_severity = 0.5.

Table 5.1: MPB parameters for change severity

movrandseed	1
geno_size	5
lambda	0.5
number_of_peaks	50
use_basis_function	0
mincoordinate	0
maxcoordinate	100
minheight	30
maxheight	70
standardheight	0
minwidth	1
maxwidth	12
standardwidth	0
peak_function	peak_function_cone

5.1.2- Results

All of the figures in Figure 5.1 involve estimated change severity. Since estimated change severity depends on the samples selected, plus/minus standard deviation is also plotted besides mean over 20 runs.

The actual and estimated change severity for penalty approach to PB5 data set is given in Figure 5.1 (a). The actual change severity has values between 0.1 and 0.4 and the average is 0.235. The expecteddistance between two random points is 0.5 for

binary representation, which is the case for random change of the optimum. Even though the average of the actual change severity is less than the half of the random change, it is a high quantity for simple strategies like local hill climbing and hypermutation. The average of the estimated change severity is 0.279 which is close to the average of the change severity measure. However, the correlation coefficient between estimated change severity and change severity is -0.139 which leads us to conclude that they are uncorrelated. Therefore we conclude that the estimated change severity does not estimate good for a single change, but it estimates good in average for this problem instance. Finally we can say that as the number of samples increase the accuracy of the estimator will increase, up to a perfect match when the complete search space is sampled.

The estimated change severity plot for Weight Coded approach to PB5 data set is given in Figure 5.1 (b). Encoding from phenotype space the genotype space in Weight Coded encoding is not obvious, so we can not plot actual change severity for Weight Coded encoding. The average normalized estimated change severity is 0.345 which is higher than the change severity of Penalty approach. Moreover, we generated 1000 random pairs of points and we measured the distance between two points for each random pair. We found the average distance between two random points as 0.4 for Weight Coded encoding. The average estimated severity is close to that value, so the changes are close to a random change in the environment. Therefore, according to estimated change severity measure we conclude that with the same change sequence in the data sets, there is a greater change in the fitness landscape of Weight-Coded approach than fitness landscape of Penalty_9 approach for these experiments. Besides that the standard deviation of Weight Coded encoding is 0.052, which is significantly lower than the standard deviation of Penalty approach.

In Figure 5.1 (c) the actual and estimated change severity for penalty approach to HARD1 data set is given. The actual change severity has values between 0.05 and 0.15 and the average is 0.106. The average of the actual change severity is less than a quarter of the random changes, so using hypermutation can be beneficial. Even though the average of the actual change severity is low, we don't think that using local hill climbing strategies will be beneficial because not only it is still rather high, but also according to our experiments the modality of the fitness landscape of MKP

is very high. The average of the estimated change severity is 0.161 which is close to the average of the change severity measure. However, the correlation coefficient between estimated change severity and change severity is 0.259 which leads us to conclude that they are uncorrelated. Moreover, the standard deviation is 0.198, which is rather large. Therefore we also conclude for this data set that the estimated change severity does not estimate good for a single change, but it estimates good in average for this problem instance.

The estimated change severity plot for Weight Coded approach to HARD1 data set is given in Figure 5.1 (d). The average normalized estimated change severity is 0.369 which is higher than the change severity of Penalty approach to the same data set. Since this value is close to 0.4, the changes are similar to a random change in the environment. Therefore, according to estimated change severity measure also for this data set we conclude that with the same change sequence in the data sets, there is a greater change in the fitness landscape of Weight-Coded approach than fitness landscape of Penalty_9 approach for these experiments. Besides that the standard deviation of Weight Coded encoding is 0.075, which is significantly lower than the standard deviation of Penalty approach.

In Figure 5.1 (e, f, g) the actual and estimated change severities for MPB with small , medium and large severity are given. The average change of actual change severity for small, medium and large changes are 0.139, 0.246, 0.303, respectively. It is clearly seen that as the change severity parameters increase, the average actual change severity also increases. This leads us to conclude that the MPB parameters about change severity are effectively controlling the change severity of the environment. The average distance between two random points is 0.39 for MPB. As the values of the parameters of the change severity are increased, the change severity gets close to random change. The average of the estimated change severities for small, medium and large change experiments are 0.044, 0.083 and 0.119, respectively. It is seen that as the parameters of change severity increase, the average estimated change severity increases. These results show that the measures; average estimated change severity and average actual change severity are correlated. However the correlation coefficients between actual and estimated change severities for small, medium and large change are 0.230, 0.274, 0.402, respectively. The correlation coefficients are not high as to conclude that estimated change severity

and change severity are correlated. Therefore the estimated change severity does not estimate actual change severity good for a single change. The ratios of average actual change severity to average estimated change severities are 3.16, 2.96 and 2.55, respectively for small, medium and large change. These results show that the average estimated change severity does not estimate good the average actual change severity for MPB.

To sum up, the results of the experiments with the change severity measure are as follows:

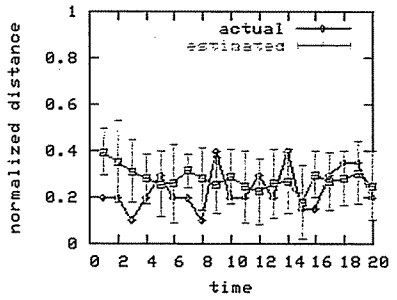
- The actual change severity measure can give us ideas about which strategy (e.g, local hill climbing, random restart, hypermutation) to use in that environment.

- The estimated change severity is not a good estimator for a single change, but it estimates well on average for the experiments with MKP instances with Penalty_9 approach.

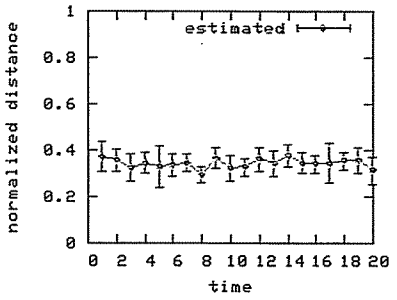
- For the tests with MKP instances, according to the estimated change severity measure, with the same change sequence in the data sets, there is a greater change in the fitness landscape of Weight-Coded approach than the fitness landscape of the Penalty_9 approach.

- MPB parameters that control change severity are effectively controlling the actual and estimated change severity of the environment.

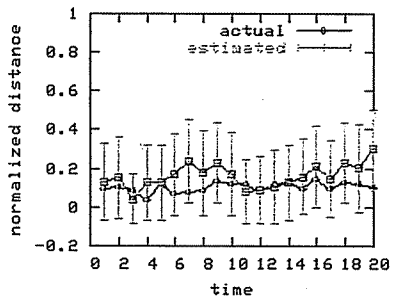
- For MPB, the estimated change severity does not estimate good actual change severity for a single change. Moreover, the average estimated change severity does not estimate well the average actual change severity. However, the average estimated change severity and average actual change severity are correlated for MPB, that is as average actual change severity increases, the average estimated change severity also increases.



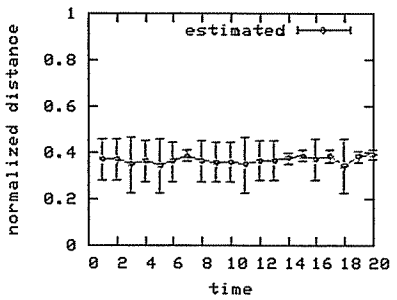
(a) Penalty_9 approach, PB5 data set



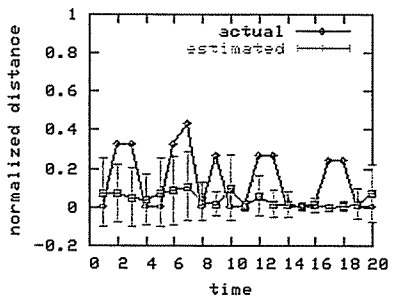
(b) Weight-Coded approach, PB5 data set



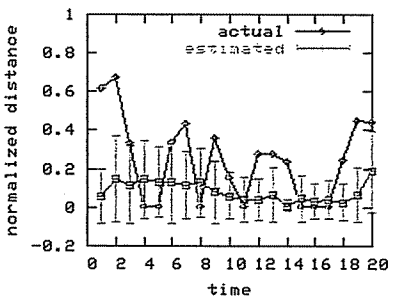
(c) Penalty_9 approach, HARD1 data set



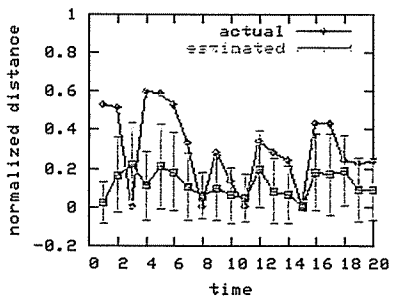
(d) Weight-Coded approach, HARD1 data set



(e) MPB approach with small severity



(f) MPB approach with medium severity



(g) MPB approach with large severity

Fig 5.1: Change Severity

5.2- Experiments for Fitness Correlation

We calculate the correlation between the fitness of the samples before and after the change. High correlation implies that the good solutions before the change are likely to remain good in the changed environment and the converse holds for the bad solutions. If the correlation is high then it is highly possible that strategies using memory, and using information from previous stages can be successful.

One of the simplest approaches that the algorithms can do is LHC. Therefore measuring fitness correlation after LHC prior to and after the change can be helpful in deciding whether simple hill climbing after change will yield good solutions. If there is a high correlation it is likely that simple hill-climbing after the change can be a good strategy to use.

5.2.1- Details of Experiments

In these experiments we measured fitness correlations, LHC fitness correlations and fitness correlations depending on time lag.

In the experiments with fitness correlations, fitness of each point is evaluated in each time instance of the change sequence. In each time instance the correlation coefficient between the fitness of points in the previous time instance and fitness of points in that time instance is calculated. At the end of the run, the average and standard deviation of the correlation coefficients are calculated. In the experiment we have done, 20 changes are applied to the environment. For the Penalty_9 approach we also implemented the same measure according to the ranks of the points to see the effect of the strong penalty applied to the infeasible solutions. The ranks are evaluated according to the fitness of the points.

In calculating LHC fitness correlations, LHC is applied to each point, and the fitness of the new points and the new points, which are local optima, are saved. After the change, LHC is applied to these previously local optima points, and the fitness of the resulting points are saved. Then we evaluate the correlation coefficient between the fitness of previously local optima and new local optima. This measure is evaluated in each time instance, and we apply 20 changes to the environment, as a result we give the standard deviation and average of the correlation coefficients. Also this measure

is calculated according to the ranks of the points for Penalty_9 approach to see the effect of the huge penalty applied to the infeasible solutions.

Lastly we measured the fitness correlation depending on time lag. To calculate this measure we generate 1000 points at the beginning of the run. We apply 60 change instances, and each fitness of each point is kept. At the end of the run, for time lag = x, where x takes value between 1 and 20, we evaluate the fitness correlation between the fitness of points at t = k, and t = k + x where k takes values between 0 and 39. Then we evaluated the average values for each time lag x, and generate the graphs.

We used 32768 samples for Penalty_9 approach, and 5000 points for Weight-Coded approach when calculating the fitness, rank, LHC fitness and LHC rank correlations, for data sets PB5, HP1, WEISH02 and HARD1. In each of the three experiments of MPB we used 2000 sample points. The default parameters of MPB are listed in Table 5.2.

We did three tests with small, medium and large changes in the environment for MPB, and evaluated all the results for each of these change severities. The same parameter sets that are used in Section 5.1 for generating small, medium, and large change severities are also used for these experiments.

Table 5.2: MPB parameters for fitness correlation

movrandseed	1
geno_size	5
lambda	0.5
number_of_peaks	50
use_basis_function	0
mincoordinate	0
maxcoordinate	100
minheight	30
maxheight	70
standardheight	0
minwidth	1
maxwidth	12
standardwidth	0
peak_function	peak_function_cone

5.1.2- Results

Table 5.3 shows the fitness correlation of samples before and after a change for MKP instances. For the Penalty_9 approach, the average correlation coefficient is extremely high (0.984-0.990). Table 5.4 shows the rank correlation of the samples before and after a change for Penalty_9 approach. In the test we have done to see if these values are affected by the strong penalties applied to infeasible solutions, it is seen that the average correlation coefficient of ranks before and after a change is also high (0.939-0.989), so we concluded that in this environment good solutions are likely to be found where good solutions in the previous stage were located, and strategies that are using the information from the previous stages can be helpful. The LHC fitness correlations and LHC rank correlations for the MKP instances with using Penalty_9 approach are given in Table 5.5 and Table 5.6, respectively. It is seen that LHC fitness and rank correlations are high. Therefore we think that algorithms that are using LHC based strategies can be successful in Penalty_9 approaches. As can be seen, LHC rank correlations and LHC fitness correlations are

also similar. Therefore we think that using rank based or fitness based algorithms do not have a significant effect on the performance of the strategies that are using information from previous stages or doing LHC for Penalty_9 approach of MKP. As can be seen there is not a significant difference in the results of real-world data sets (PB5, HP1, WEISH02) and generated hard data set (HARD1) in the fitness landscape of Penalty_9 approaches according to the fitness, rank, fitness-LHC, and rank LHC correlations. Thus it can be said that the characteristics of real-world data sets and hard, generated data set are similar according to fitness and fitness LHC correlations for Penalty_9 approach.

The average fitness correlations for the Weight Coded encoding are in the range 0.528-0.714, which are significantly lower than the correlations of Penalty_9 approach. Therefore it can be said that in the fitness landscape of Penalty_9 approach good solutions are more likely to be found where good solutions in the previous stage are than in the fitness landscape of Weight Coded, so it is a high probability that strategies that are using the information from the previous stages can be more helpful for Penalty_9 approach than Weight Coded approach. The average LHC fitness correlations are in the range 0.517-0.643 for Weight Coded encoding, which are less than the results of this measure for Penalty_9 approach. Thus it is a high probability that Penalty_9 approach can be more successful than Weight Coded approach with strategies based on LHC for MKP. Also for Weight Coded approach there is not a significant difference in the results of real-world and generated data sets according to the fitness and fitness LHC correlations. Thus it can be said that the characteristics of real-world data sets and hard, generated data set are also similar according to fitness and fitness LHC correlations for Weight Coded approach.

Table 5.3: Fitness correlations for MKP instances

	Penalty_9				Weight-Coded			
	PB5	HP1	WEISH02	HARD1	PB5	HP1	WEISH02	HARD1
avg	0.987	0.984	0.990	0.985	0.604	0.528	0.559	0.714
std	0.009	0.013	0.008	0.009	0.092	0.128	0.159	0.063

Table 5.4 Rank correlations for Penalty_9 approach

	PB5	HP1	WEISH02	HARD1
avg	0.983	0.939	0.989	0.984
std	0.010	0.028	0.009	0.010

Table 5.5 LHC fitness correlation for MKP instances

	Penalty_9				Weight-Coded			
	PB5	HP1	WEISH02	HARD1	PB5	HP1	WEISH02	HARD1
avg	0.764	0.771	0.902	0.860	0.643	0.517	0.525	0.629
std	0.058	0.101	0.027	0.028	0.084	0.198	0.187	0.037

Table 5.6: LHC rank correlations for Penalty_9 approach

	PB5	HP1	WEISH02	HARD1
avg	0.751	0.757	0.900	0.848
std	0.061	0.100	0.029	0.301

Fitness correlations and LHC fitness correlations for MPB are given in Table 5.7 and Table 5.8, respectively. As can be seen, as the change severity increases both the fitness correlation and LHC fitness correlation decreases. This is an expected result for the real world problems, and the experimental results show that the change severity control is realistic according to the measures; fitness correlation and fitness LHC correlation for MPB.

Table 5.7 Fitness correlations for MPB instances

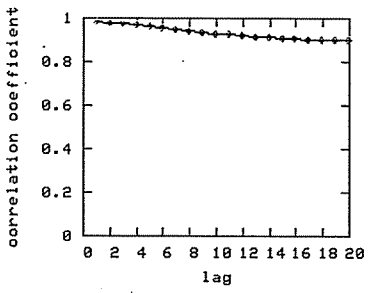
	Small	Medium	Large
avg	0.959	0.927	0.882
std	0.037	0.049	0.082

Table: 5.8 LHC fitness correlations for MPB instances

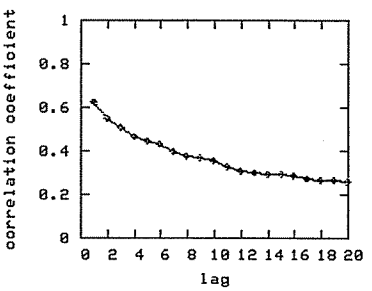
	Small	Medium	Large
avg	0.958	0.926	0.893
std	0.027	0.048	0.050

Figure 5.2 shows how the fitness correlation depends on the time lag. As expected, the correlation coefficient decreases with increasing time lag for both of the Penalty_9 and Weight Coded approaches for both of the real-world and generated data set. Therefore it can be said that the characteristics of real-world data set and hard, generated data set are similar according to fitness correlation depending of time lag for both approaches. Moreover the decrease is stronger for Weight Coded approach, than Penalty_9 approach. These results are in parallel with the results of fitness correlation, fitness LHC correlation, and change severity correlation of Weight Coded approach. As can be seen the expected result that the correlation decreases with time lag holds for also MPB. As a conclusion it can be said that MPB

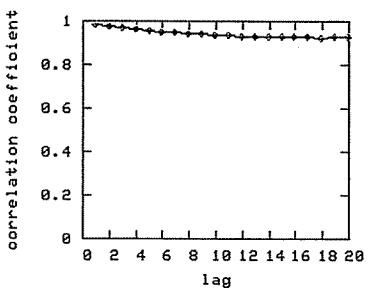
is also realistic according to these results. However there is not a great effect of change severity on MPB for this measure, since the graphics 5.2 (e), 5.2 (f) and 5.2 (g) are similar to each other.



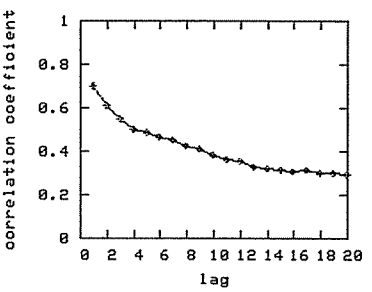
(a) Penalty_9 approach, PB5 data set



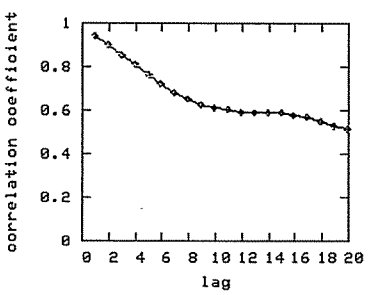
(b) Weight-Coded approach, PB5 data set



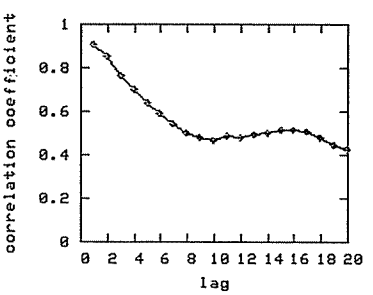
(c) Penalty_9 approach, HARD1 data set



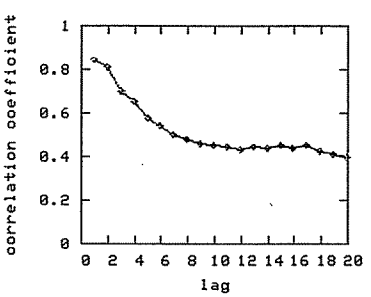
(d) Weight-Coded approach, HARD1 data set



(e) MPB approach, small change severity



(f) MPB approach, medium change severity



(g) MPB approach, large change severity

Figure 5.2: Fitness correlation depending on time lag

The results of the experiments can be summarized as follows:

- In comparison of Penalty_9 and Weight Coded approaches according to fitness correlation measure, it can be said that in the fitness landscape of Penalty_9 approach good solutions are more likely to be found where good solutions in the previous stages are than in the fitness landscape of Weight Coded. Therefore, using the information from the previous stages can be more helpful in Penalty_9 than Weight Coded.
- In comparison of Penalty_9 and Weight Coded approaches according to the LHC fitness correlation measure, it is seen that the correlation is higher in Penalty_9. Thus it is a high possibility that using strategies based on LHC in change instances can be more helpful in Penalty_9 approaches than in Weight Coded approaches.
- In comparison of rank based and fitness based measures, it is seen that the results are similar. Therefore we conclude that using rank based or fitness based algorithms do not have a significant effect on the performance of the strategies that are using information from previous stages or doing LHC for Penalty_9 approach for the MKP.
- It can be said that the characteristics of real-world data sets and hard, generated data set are similar according to fitness, fitness LHC correlations and fitness correlations depending on time lag measures for both of the Penalty_9 and Weight Coded approaches to Weight Coded.
- The decrease in the fitness correlation depending on time lag is stronger in Weight Coded than in Penalty_9.
- The change severity control is realistic according to the measures; fitness correlation and fitness LHC correlation for MPB.
- The expected result that the correlation decreases with time lag holds for also MPB. Therefore MPB is also realistic according to this measure. However there is not a great effect of change severity on MPB for this measure.

5.3- Experiments for Fitness Change Correlation of Similar Points

The correlation coefficients of fitness changes of points with distance d are calculated. We expect to see that correlation decreases with increasing d. High correlation implies that points which are within a fixed distance, change similarly and vice versa, and the structure of the landscape does not change too much.

5.3.1- Details of Experiments

This measure involves a distance between solutions. For comparison reasons we used normalized distance in this measure.

For the Penalty_9 approach, at the beginning of the run we generate 1000 random points for each distance value d, and then for each 1000 random points we generate 1000 points with distance d to these points. The number of different distances is equal to the number of objects, n, of the data set, that is the chromosome length in the genetic algorithm, e.g., 20 for PB5 data set.

For the Weight Coded approach, we also generate 1000 random points at the beginning of the run. However these points are not totally random. All the gene values are only let to take the maximum gene value or minimum gene value. This process is similar to converting the Weight Coded space into a binary space. To be able to compare with Penalty_9 approach and to be able to generate points with a fixed distance we used such a procedure. After generating these random points, for each 1000 random point, 1000 points with distance d to these points are generated. Also the number of different distances is equal to the number of objects in this approach.

This measure is calculated in each time instance, and at the end of the run all of the graphs are averaged over 20 runs.

The implementation of this measure to the MPB approach is same as the implementation of Weight Coded approach.

We did tests with PB5 and HARD1 data sets of MKP. For the tests with MPB, we designed three tests instances with different size of spaces. For controlling the space size we changed the geno_size parameter of MPB. Three experiments are done with

small, medium and large space size, with the parameter geno_size values of 10, 15, and 20, respectively. The other parameters of the MPB are given in the Table 5.9.

Table 5.9: MPB parameters for fitness change correlation

movrandseed	1
vlength	1
height_severity	1
width_severity	0.25
lambda	0.5
number_of_peaks	50
use_basis_function	0
mincoordinate	0
maxcoordinate	100
minheight	30
maxheight	70
standardheight	0
minwidth	1
maxwidth	12
standardwidth	0
peak_function	peak_function_cone

5.3.2- Results

The results of the experiments are shown in Figure 5.3. For the real-world data set it is seen that for d = 0.05, which is the minimum value for d, the correlation coefficient is 0.705 for Penalty_9 approach, and 0.455 for Weight-Coded approach. For d = 1, which is the maximum distance that can be achieved in the corresponding spaces, the correlation coefficient is -0.352 and -0.074 for Penalty_9 and Weight-Coded approaches, respectively. As can be seen the correlation of close points (d =

0.05) for Penalty_9 approach are higher than the correlation of close points for Weight-Coded approach. Besides that the correlation of further points (d = 1) for Penalty_9 approach are less than the correlation of close points for Weight-Coded approach. Thus it can be said that the structure of the landscape according to similar points changes more for Weight-Coded approach. However the structure of the landscape according to further points changes more for Penalty_9 approach. For both of the approaches the correlation decreases with distance. It is seen that the correlation coefficient turns to negative between d = 0.45 – 0.5 for Penalty_9 approach, and it turns negative between d = 0.5 – 0.55 for Weight-Coded approach. The distances where the correlation coefficient turns to negative are similar to each other. The reason of the correlation to turn to negative perhaps can be exploited algorithmically in further studies.

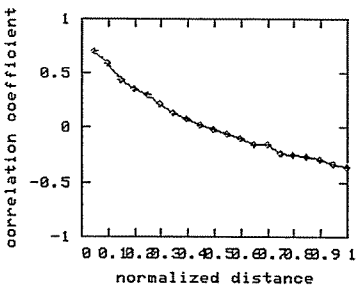
For the generated, hard data set it is seen that for d = 0.01, which is the minimum value for d, the correlation coefficient is 0.937 for Penalty_9 approach, and 0.811 for Weight-Coded approach. For d = 1 the correlation coefficient is -0.481 for Penalty_9 approach and -0.161 for Weight-Coded approach. As can be seen the correlation of close points (d = 0.01) for Penalty_9 approach are higher than the correlation of close points for Weight-Coded approach. Besides that the correlation of further points (d = 1) for Penalty_9 approach are less than the correlation of close points for Weight-Coded approach. Thus it can be said that the structure of landscape according to similar points changes more for Weight-Coded approach than for Penalty_9 approach also for the hard data set. However the structure of landscape according to further points changes much for Penalty_9 approach. For both of the approaches, the correlation decreases with distance. It is seen that the correlation coefficient turns to negative between d = 0.49 – 0.5 for Penalty_9 approach, and it turns negative between d = 0.48 – 0.49 for Weight-Coded approach. Also for the hard data set the point where the correlation turns to negative are close to 0.5.

As a comparison of the real-world and generated data sets it can be said that the characteristics of the plots for real-world and hard data set are similar in all of the investigated points, that are the structure of landscape according to similar points changes more for Weight-Coded approach than Penalty_9 approach, and the structure of landscape according to further points changes more for Penalty_9 approach than Weight-Coded approach. Moreover there are two aspects of the

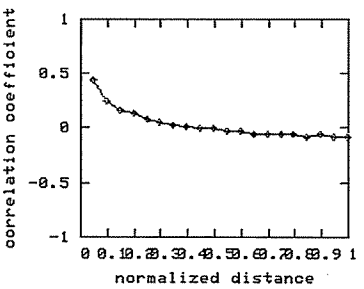
measure that are common for both of the data sets and approaches. First of them is that the correlation decreases with distance, and the second one is that the correlation turns to negative at distances close to 0.5. Therefore it can be concluded that the characteristics of the real-world and generated data sets are similar according to this measure.

For the MPB approach, we compare the correlation coefficient values for normalized distance d = 0.1 for all of the search spaces. However, for medium search space d = 0.1 is not a valid distance according to the algorithm of the measure, but d = 0.0666, and d = 0.1333 are valid distances. The average of the correlation coefficients of d = 0.0666 and 0.1333 are used as an estimate for the correlation coefficient value of d = 0.1. The correlation coefficients for d = 0.1 are 0.592, 0.661, and 0.698 for small, medium and large search spaces, respectively. It can be said that as the size of the search space, geno_size parameter of MPB increases, the correlation of close points increases, that is the degree of change in the structure of landscape according to similar points increases. The correlation coefficient values for d = 1 are -0.445, -0.536, and -0.614 for small, medium and large search spaces, respectively. As can be seen as the size of the search space increases the correlation of further points decreases. Thus the degree of change in structure of the landscape according to further points decreases. In the experiments for MKP instances, it is seen that the higher the correlation for similar points the less the correlation for further points. This rule also holds for MPB experiments. Moreover MPB is capable of generating fitness landscapes that have different characteristics of fitness change correlations with manipulating the geno_size parameter. It is capable of both generating fitness landscapes that have similar characteristics to Penalty_9 approach, and to Weight-Coded approach according to fitness change correlation of similar points measure. The point where correlation coefficient turns to negative for MPB experiments are 0.4-0.5, 0.467-0.533, 0.5-0.55, for small, medium and large search spaces, respectively. All of the turning points are close to 0.5, which is the first rule holding for both of the MKP instances and approaches. Therefore it can be said that the fitness landscapes generated by MPB has similar characteristics to MKP instances according to this property. Moreover, as expected, the correlation coefficient decreases with distance for also MPB instances, which is the second rule holding for

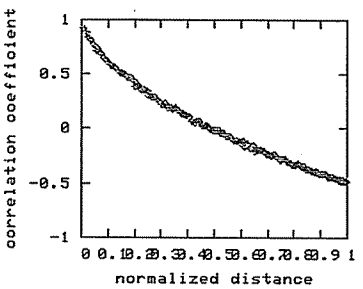
both of the MKP instances and approaches. Therefore it can be said that the MPB has similar characteristics with MPB instances according to also this property.



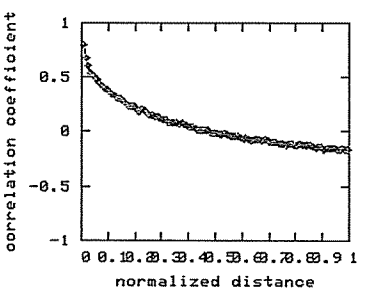
(a) Penalty_9 approach, PB5 data set



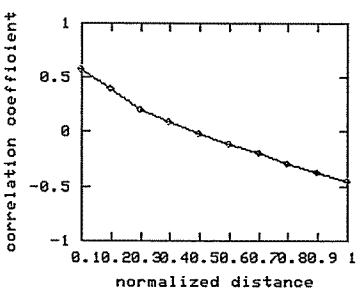
(b) Weight-Coded approach, PB5 data set



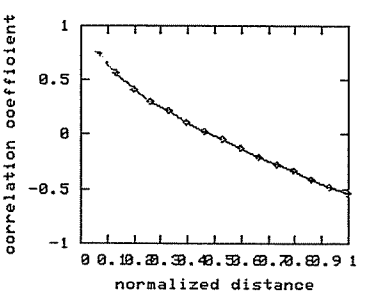
(c) Penalty_9 approach, HARD1 data set



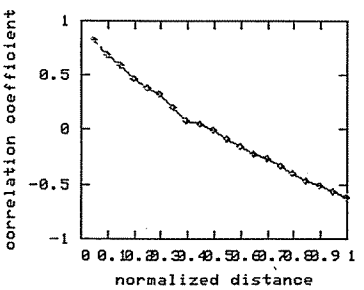
(d) Weight-Coded approach, HARD1 data set



(e) MPB approach with small searh space



(f) MPB approach with medium search space



(g) MPB approach with large search space

Figure 5.3: Fitness change correlations between random individuals of normalized distance d.

As a summary of the results it can be said that:

- The structure of landscape according to similar points changes more for Weight-Coded approach than for Penalty_9 approach for both of the real-world and generated data sets. However the structure of landscape according to further points changes more for Penalty_9 approach than Weight-Coded approach for both of the real-world and generated data sets. Moreover there are two common points for both of the data sets and approaches, first of which is that the correlation decreases with distance, and the correlation turns to negative at a distance close to 0.5. Therefore, it can be said that the real-world and generated data sets have similar characteristics according to this measure.
- MPB is capable of both generating fitness landscapes that have similar characteristics to Penalty_9 approach, and to Weight-Coded approach according to fitness change correlation of similar points measure. More generally MPB is capable of generating fitness landscapes that have different characteristics for this measure by controlling the parameter geno_size.
- The two common properties which hold for both of the MKP instances and approaches also hold for MPB. Therefore MPB is realistic also in this sense.

5.4- Experiments for Estimated Value of Last-Stage Local Optima

Here we measure the benefit of using last stage's good solutions. In order to do so, we compare the fitness we obtain by LHC from random points with last stage's LHC based best k sample points.

5.4.1- Details of Experiments

We did experiments on one real-world data set, PB5, one hard data set, HARD1, from Chu and Beasley's benchmark suite, and on MPB problem generator.

In evaluating average fitness of running local hill climbing from random points, 20 environmental changes are done, in each change instance, average of the fitnesses reached by LHC from 50 random points is calculated. Since we use information between the environments 10 to 20 in the next section, also in this section we average these values over the environments 10-20.

The average optimal value is evaluated by simply averaging the global optimum of 10 to 20 environmental changes.

In each time instance, k best samples are found. Then LHC is run from all of these best samples, and they are sorted. The solutions are called k LHC based best samples. Value of good solutions from previous stages is evaluated as follows: In each time instance, local hill climbing is done from previous k LHC based best samples, for k = 1, k = 2, and so on, and stored as the value for each k. At the end of the run the average for each k is evaluated.

In evaluating combined value of k best solutions from previous stage, local hill climbing is done from each of the 50 previous LHC based best samples. The fitness reached by the first sample is assigned to k = 1, the maximum of the fitness reached by 1^{st} and 2^{nd} best samples of last-stage is assigned to k = 2 and, so on. This process is done in each time instance, and at the end of the run the average is evaluated for each of the k values. A similar algorithm is used when evaluating the combined value of the k random solutions. In each time instance 50 random points are generated. The fitness reached by the first random point is assigned to k = 1. The maximum of the fitness reached by LHC from first and second random points is assigned to k = 2 and, so on. At the end of the run averages for each of the k values are calculated.

We used 32768 samples for Penalty_9 approach, and 5000 points for Weight-Coded approach when calculating the measures for data sets PB5 and HARD1. In the experiment with MPB we used 2000 sample points. The parameters of MPB are listed in Table 5.10

Table 5.10: MPB parameters for estimated value of last-stage local optima

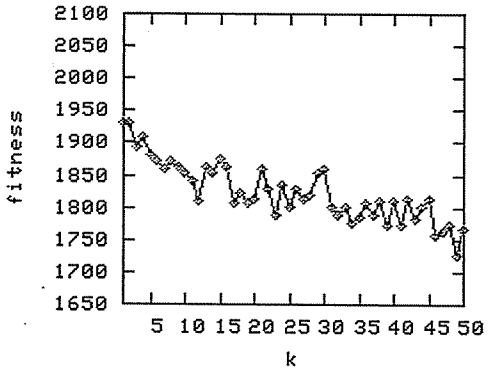
movrandseed	1
geno_size	4
vlength	1.5
height_severity	1.5
width_severity	0.375
lambda	0.5
number_of_peaks	50
use_basis_function	0
mincoordinate	0
maxcoordinate	100
minheight	30
maxheight	70
standardheight	0
minwidth	8
maxwidth	12
standardwidth	0
peak_function	peak_function_cone

5.4.2- Results

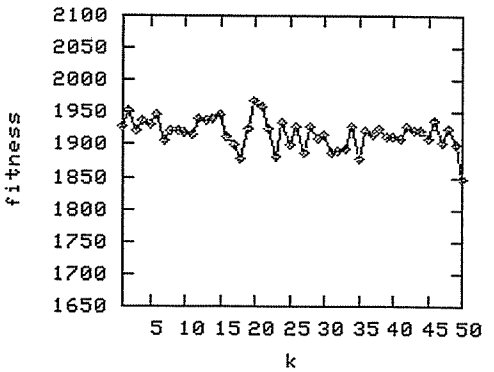
For the PB5 data set, the average fitness obtained by running LHC from a random solution is approximately 1642 for the Penalty_9 approach, and 1778 for the Weight Coded approach, and the average of the optimal value is 2043.4. Figure 5.4 shows the fitness values obtained by running LHC from the k-th best sample of last-stage. It can be seen that running LHC from any of the k LHC based best samples yields better solution than running LHC from a random point for both of the approaches. If we compare the fitness obtained when starting from random points and k-th best samples of last-stage, Weight Coded yields better solutions than Penalty_9 in both

cases. Moreover in Penalty_9 approach the quality of the solutions decreases with k, but in Weight Coded this is not the case.

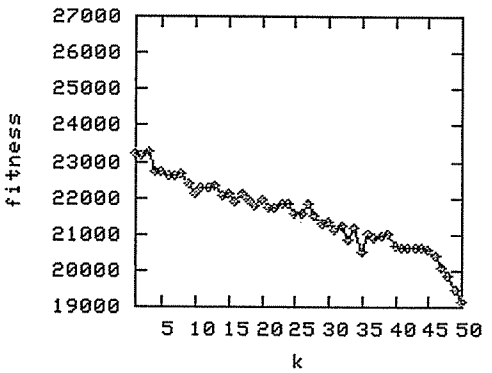
Figure 5.5 shows the combined value of k best solutions from the previous stage. As is to be expected the the fitness increases with k. It is seen that running LHC from LHC based best samples of previous stage is more successful than running LHC from random samples for both of the approaches. As can be seen the combined value does not increase significantly after k = 10 -15 for both of the approaches. Therefore keeping more points in the memory does not seem very beneficial. The fitness value obtained by keeping 15 LHC based best samples in the memory is 2008.44 and 2020.99 for Penalty_9 and Weight-Coded approaches, respectively. These fitness values are in the %2 gap of the average optimal fitness value (2043.4). Therefore running LHC from the k best samples of the previous stage is a successful method. The fitness value obtained by keeping 15 random samples in the memory for Weight Coded approach is 1891.75, which is in the % 8 gap of the average optimal fitness value and 1972.99 for Weight-Coded approaches, which is in the %4 gap of the average optimal fitness value. For both of the approaches using last-stage best samples gives better results than random numbers. However using random points in the Penalty_9 approach gives significantly worse results than using random points in the Weight Coded approach. As a conclusion it can be said that keeping high-quality solutions from the previous stage is beneficial for both of the approaches for this data set.



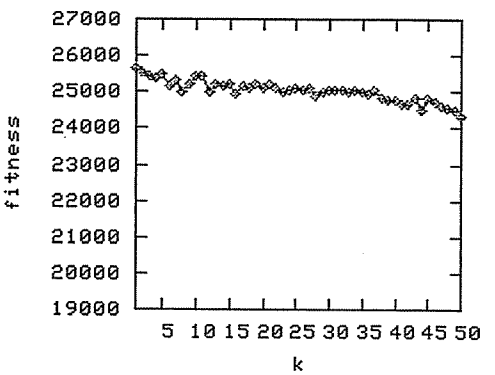
(a) Penalty_9 approach, PB5 data set



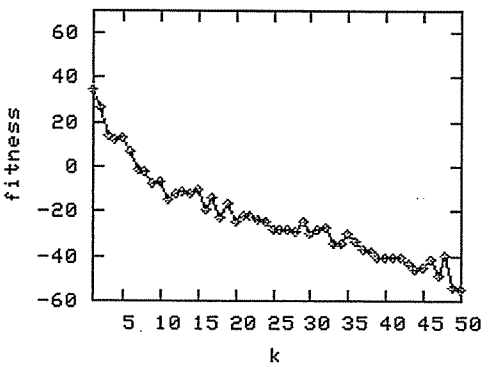
(b) Weight-Coded apptroach, PB5 data set



(c) Penalty_9 approach, HARD1 data set



(d) Weight-Coded approach, HARD1 data set



(e) MPB

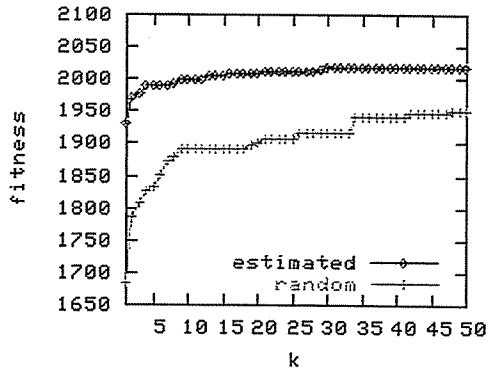
Figure 5.4: Estimated value of good solutions from previous stage

For the HARD1 data set, the average fitness obtained by running LHC from a random solution is approximately 21736 for the Penalty_9 approach, and 23794 for the Weight Coded approach, and the average of the optimal value is 27443.9. Figure 5.4 shows the fitness values obtained by running LHC from the k-th best sample of last-stage.

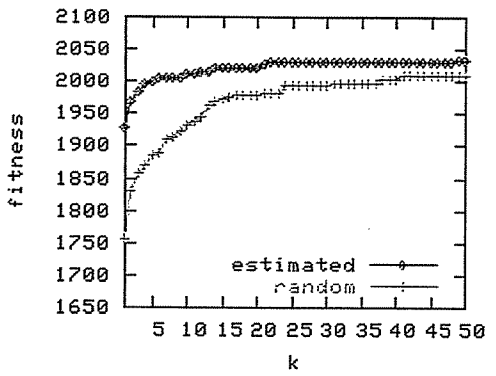
It can be seen that running LHC from any of the k LHC based best samples yields better solution than running LHC from a random point for Weight-Coded approach. However for Penalty_9 approach best samples perform better than random samples for $k < 25$. If we compare the fitness obtained when starting from random points and k-th best samples of last-stage, Weight Coded yields better solutions than Penalty_9 in both cases. Moreover in both of the approaches there is a decrease in quality with increasing k. However, the decrease is slower in Weight Coded than Penalty_9.

In the comparison of the combined value of last-stage best samples and random points, it is seen that running LHC from LHC based best samples of previous stage is more successful than running LHC from random samples for both of the approaches.

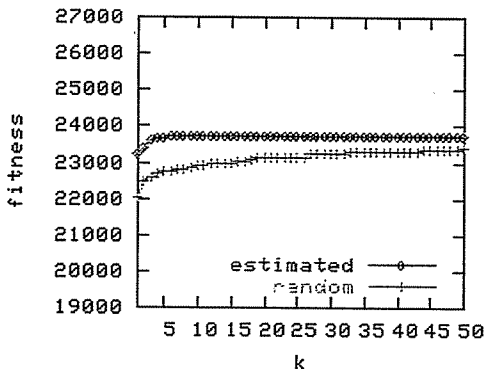
It can be said that the combined value does not increase after $k = 5, 10$ for both of the approaches. Therefore keeping more points in the memory isn't necessary. The averages of combined value of last stage local optima are 25201.98, 26092.34 for random points and estimated local optima respectively for Weight Coded approach. For Penalty_9 approach 23095.79, 23734.98 are the average of the combined value of past optima respectively for random points and estimated local optima. It can be said that running LHC gives better results in Weight Coded than Penalty_9. Interestingly even running LHC from random points of Weight Coded gives better results than running LHC from estimated local optima of Penalty_9 approach. Probably this result is because of the different structures of the fitness landscape of the approaches. The fitness value obtained by keeping 10 LHC based best samples in the memory is 23757 and 26103 for Penalty_9 and Weight-Coded approaches, respectively. These fitness values are in the %2 gap and %1 gap of the average optimal fitness value (27443,9). The fitness value obtained by keeping 10 random samples in the memory for Weight Coded approach is 22929.5, which is in the % 2 gap of the average optimal fitness value and 25011,1 for Weight-Coded approaches, which is in the % 1 gap of the average optimal fitness value. It can be said that running LHC from random samples or LHC based best samples of last-stage is beneficial. Even though running LHC from high quality solutions are better than running LHC form random solutions, the difference is not significant.



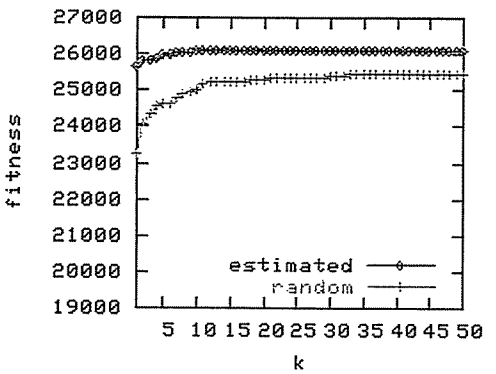
(a) Penalty_9 approach, PB5 data set



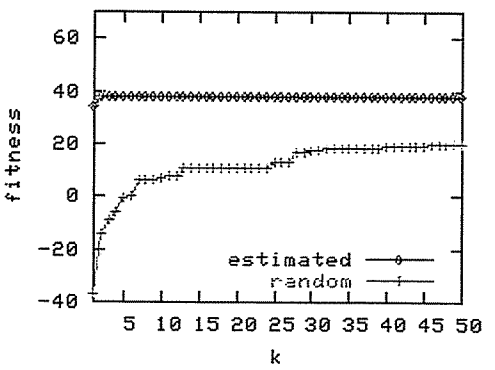
(b) Weight-Coded approach, PB5 data set



(c) Penalty_9 approach, HARD1 data set



(d) Weight-Coded approach, HARD1 data set



(e) MPB

Figure 5.5: Combined value of k best solutions from previous stage

For the MPB approach, the average fitness obtained by running LHC from a random solution is approximately -45, and the average of the optimal value is 69.4959. According to the parameter set used each peak takes values between 30 and 70. However the fitness of a point after LHC takes value of -45. The reason for this is that the step size of the LHC is very large according to the slope of the peaks and

size of the search space. Figure 5.4 shows the fitness values obtained by running LHC from the k-th best sample of last-stage. It can be seen that running LHC from any of the k LHC based best samples yields better solution than running LHC from a random point for $k < 40$. Moreover in MPB the quality of the solutions decreases with k, as in Penalty_9 approaches.

Figure 5.5 shows the combined value of k best solutions from the previous stage for MPB approach. As is to be expected the fitness increases with k. It is seen that running LHC from LHC based best samples of previous stage is more successful than running LHC from random samples. As can be seen the combined value does not increase after $k = 2$. Therefore keeping more points in the memory does not seem very beneficial. As a conclusion it can be said that keeping a few high-quality solutions from the previous stage is beneficial for MPB approach for this parameter set.

The results of the experiments are summarized in the following paragraphs:

- For both of the approaches, keeping a few of the best samples from the previous stage is reasonable for the real-world small data set.

- For the used parameter set for MPB, as in MKP approaches, keeping a few of the high quality solutions from the previous stage is beneficial.

5.5- Experiments for Value of Past Optima

Here we measure the benefit of using previous stages' global optima. It is usually assumed that keeping information regarding the global optima of previous stages in cyclic environments is beneficial. To test this assumption, we compare the fitness we obtain by LHC from random points with k previous global optima. In some cases it is not possible or practical to find the global optima. In these cases, we use the point obtained by LHC from the best sample as an estimate for the global optima. Then we calculate the above measure using this estimate.

5.5.1- Details of Experiments

Here we compare the fitness we obtain by LHC from random points with k previous global optima. In the cases when finding global optimum is impractical or impossible

we use the point obtained by doing LHC from the best sample as an estimate of the global optimum. If there are more than one global optimum, we choose randomly one of them and use it in our experiments with the idea that EA's mostly focus on one global optimum.

Test are done on three problem instance, one of them is a real-world data set, PB5, the other one is HARD1 data set, which is from Chu and Beasley's benchmark suite and known to be hard, and the last experiment is done on MPB suite. The parameters of the MPB are same as the parameters used in section 5.4.

In evaluating average fitness of running local hill climbing from random points, 20 environmental changes are done, in each change, average of the fitness reached by LHC from 50 random points is calculated and at the end of the run average of these values is calculated.

The average optimal value is evaluated by simply averaging the global optimum of 10 to 20 environmental changes.

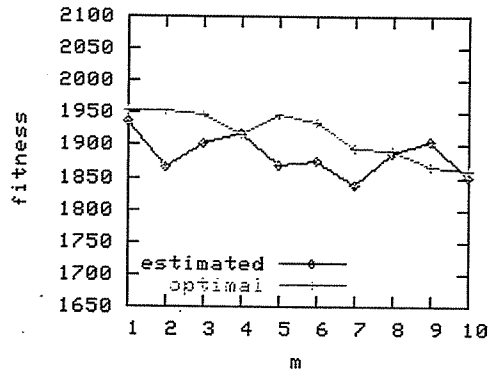
The value of best solution from m-th previous stage is calculated as follows: In every time instance the best solution is found, if there are more than one solution with equal fitness randomly one of them is chosen. Then LHC is done to make the point an optimum. Previous 10 global optima is always stored in the memory, so the results are taken in the time instances between t = 10 and t = 20 and averaged. In each time instance, for each m the fitness reached by LHC from m-th previous global optima is used. And the same procedure is used for the estimate of the global optima.

Combined value of optimal solutions from m previous stages is evaluated as follows. In each time instance from t = 10 to t = 20 the fitness reached by LHC from the 1st previous global optimum is assigned to m = 1, and the maximum of the fitness reached by LHC from the 1^{st} and 2^{nd} previous global optima is stored to m = 2, and, so on for other m values. At the end of the run averages are evaluated for each m. The same procedure is used for the estimated value of the global optima. For the measures with random points, 10 random points are generated. Between the time instances t = 10 and t = 20, in each time instance, the point reached by LHC from the first random point is assigned to m = 1, the maximum of the fitness reached by LHC from the first and second random points is assigned to m = 2, and so on for the other m values. At the end of the run average for each m is evaluated.

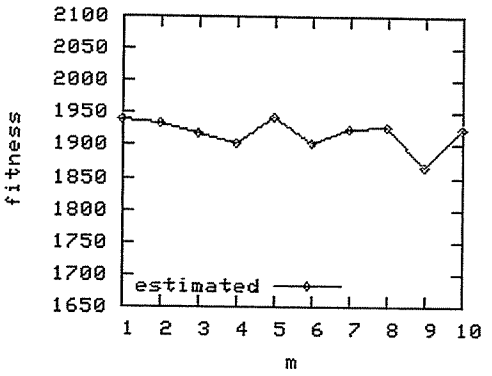
5.5.2- Results

Here the performance of keeping best solutions of previous stages is compared to the performance of random points. For the PB5 data set, the average fitness obtained by running LHC from a random solution is approximately 1642 for the Penalty_9 approach, and 1778 for the Weight Coded approach. The result of comparing single solutions can be seen in the Figure 5.6. It can be said that starting LHC from a local optimum of previous stages gives significantly better performance than starting LHC from random points for both of the approaches. Moreover for the Penalty_9 approach the performance of starting from the true optima of previous stages is also plotted. It can be said that the estimated measure is a reasonable approximation.

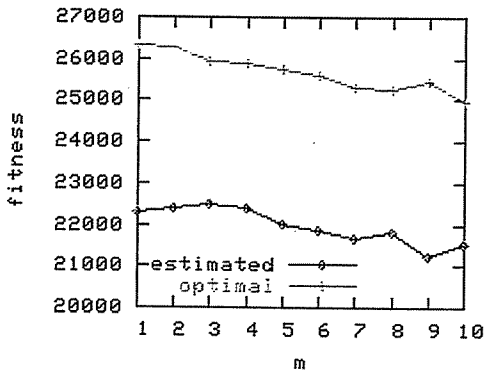
In the comparison of the combined value of several previous best samples and random points for PB5 data set, it is seen that running LHC from LHC based best samples of previous stages is more successful than running LHC from random samples for both of the approaches (Figure 5.7). It can be seen that the increase in the performance after 3 environmental stages ago is rather small.



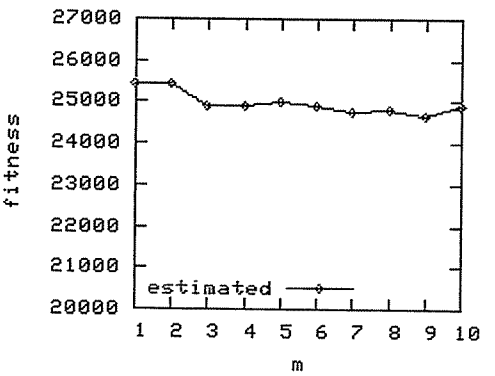
(a) Penalty_9 approach, PB5 data set



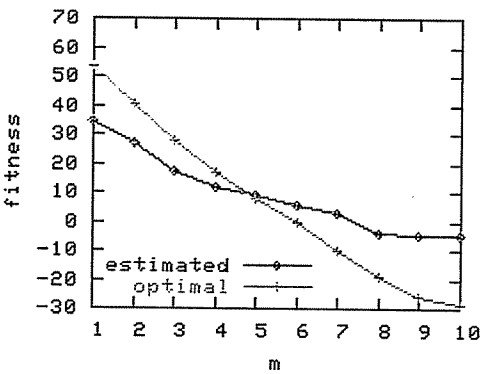
(b) Weight-Coded approach, PB5 data set



(c) Penalty_9 approach, HARD1 data set



(d) Weight-Coded approach, HARD1 data set



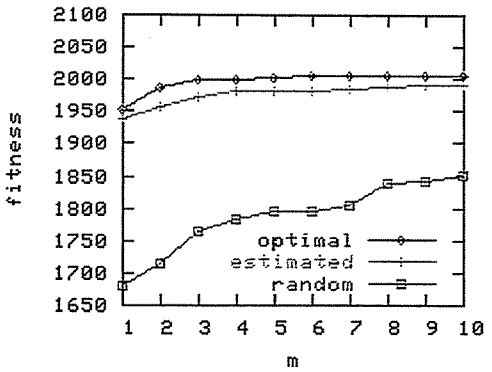
(e) MPB

Figure 5.6: Value of best solutions from m-th previous stage

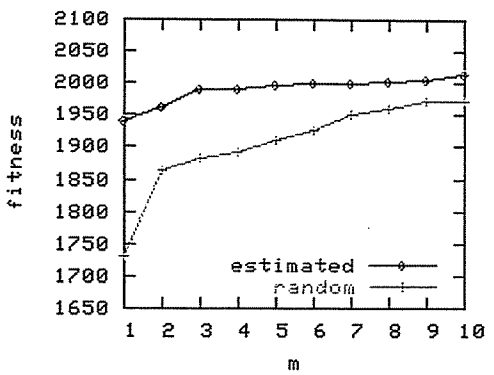
For the HARD1 data set, the average fitness obtained by running LHC from a random solution is approximately 21736 for the Penalty_9 approach, and 23794 for the Weight Coded approach. The result of comparing single solutions can be seen in the Figure 5.6. It can be said that starting LHC from a local optimum of previous stages gives significantly better performance than starting LHC from random points for Weight-Coded approach. However for Penalty_9 approach, there is a slight

benefit of using LHC based best samples only for m < 5. Moreover for the Penalty_9 approach the performance of starting from the true optima of previous stages is also plotted. As can be seen starting LHC from the true optima of previous stages is significantly better than starting LHC from random points. Thus it can be said that the estimated measure is not a very good approximation of the true optima for this data set.

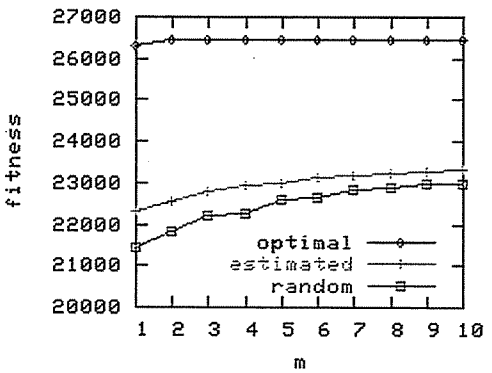
In the comparison of the combined value of several previous best samples and random points for HARD1 data set, it is seen that running LHC from LHC based best samples of previous stages gives significantly better performance than running LHC from random samples for Weight-Coded approach (Figure 5.7). Even though for the Penalty_9 approach LHC based best samples give better performance than random samples, there is not a great difference. It can be seen that the increase in the performance after 2 or 3 environmental stages ago is rather small.



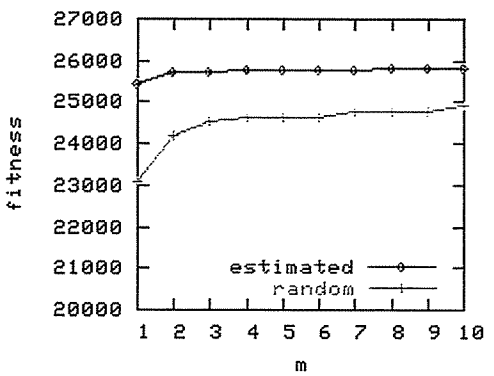
(a) Penalty_9 approach, PB5 data set



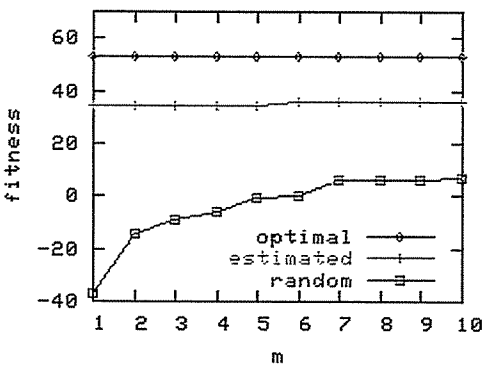
(b) Weight-Coded approach, PB5 data set



(c) Penalty_9 approach, HARD1 data set



(d) Weight-Coded approach, HARD1 data set



(e) MPB

Figure 5.7: Combined value of best solutions from m previous stages

For the MPB data set, the average fitness obtained by running LHC from a random solution is approximately -45. In the comparison of single solutions, it can be said that both starting LHC from a local optimum of previous stages and true optimum gives significantly better performance than starting LHC from random points. The performance of true optimum is close to the performance of the estimated measure. T

In the comparison of the combined value of several previous best samples and random points for MPB, it is seen that running LHC from LHC based best samples and true optima of previous stages is more successful than running LHC from random samples (Figure 5.7). It can be seen that keeping only the LHC based best

sample of previous stage is reasonable. In the comparison of the plots of the MPB and Penalty approach of PB5 data set, the characteristics of the graphs are similar. Therefore we conclude that this parameter set of the MPB generates a similar environment to the Penalty_9 approach of PB5 data set.

The results of the experiments are summarized in the following paragraphs:

- Keeping the high quality solutions of a few previous stages is reasonable for both of the approaches for MKP instances and MPB.
- The MPB instance used gave similar characteristics to the real-world data set PB5 for the Penalty_9 approach.

6- CONCLUSION

EAs are mostly known to be black-box optimization tools. Analysis of the dynamics of the EAs is needed to better understand the nature of the EAs. In the last decade there has been some recent work on the analysis of the static environments. However the work on the analysis of the dynamic environments is currently in its very early stages. By analyzing dynamic environments, new insights can be gained about the nature of the dynamic environments.

The aim of this thesis was to make a first step towards the analysis of dynamic environments. For this purpose we propose a number of measures to analyze the dynamic environments. The measures and the information that can be gained by applying these measures are summarized as follows:

- Change severity and estimated change severity
- Fitness correlation
- LHC fitness correlation
- Fitness change correlation of similar points
- Estimated value of last-stage local optima
- Value and estimated value of past optima

A real-world problem, i.e.the multidimensional knapsack problem, and a benchmark problem, which is MPB problem are analyzed by using these measures.

A new dynamic benchmark problem, the dynamic MKP, for generating dynamic multidimensional knapsack problems is proposed.

The analysis of the MKP instances gave interesting results that can be used in the design of algorithms. Moreover it is seen that MPB is capable of generating wide variety of problems with different characteristics.

In the future work, further real-world problems can be analysed by using these measures and by using the result better algorithms can be designed for these problems. This is a work in progress. Moreover classification of the real-world problems can be done and a mapping from the classes of different types of dynamic problems to different types of evolutionary algorithms can be established. Furthermore artificial benchmark problems that are capable of representing a wide variety of real-world problems can be generated.

REFERENCES

[1] **Angeline, P.J., Fogel, D.B. and Fogel L.J.,** 1996. A comparison of self-adaptation methods for finite state machines in dynamic environments. In L. J. Fogel et al., editor, *Proceedings of the fifth annual Conference on Evolutionary Programming*, pages 441–449.

[2] **Bierwirth, C. and Mattfeld, D.C.,** 1999. Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1):1–18.

[3] **Branke, J.,** 1999. Evolutionary algorithms for dynamic optimization problems - a survey. Technical Report 387, Insitute AIFB, University of Karlsruhe, February.

[4] **Branke, J.,** 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In *Congress on Evolutionary Computation CEC99*, volume 3, pages 1875–1882. IEEE.

[5] **Branke, J.,** 2001. Evoutionary Optimization in Dynamic Environments, Kluwer.

[6] **Branke, J., Salihoğlu, E. and Etaner, Ş., 2005.** Towards an Analysis of Dynamic Environments, GECCO 2005: Genetic and Evolutionary Computation Conference 2005, ACM, in press.

[7] **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 AIC-90-001, Naval Research Laboratory, Washington DC.

[8] **De Jong, K.A.,** 1999. Evolving in a changing world. In ISMIS '99: Proceedings of the 11th International Symposium on Foundations of Intelligent Systems, pages 512-519. Springer.

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

[10] **Fogel, L.J. Owens, A.J. and Walsh, M.J.,** 1965. Artificial intelligence through a simulation of evolution. In: A. Callahan, M. Maxfield, L. J. Fogel, Eds., Biophysics and Cybernetic Systems. Spartan, Washington DC, pp. 131-156.

[11] **GLPK.** GNU Linear Programming Kit, online, http://www.gnu.org/software/glpk/glpk.html

[12] **Gottlieb, J.,** 1999. *Evolutionary Algorithms for Combinatorial Optimization Problems.* PhD Thesis, Mathematisch-Naturwissenschaftlichen Fakultaet der Technischen Universitaet Clausthal.

[13] **Gottlieb, J.,** 2001. On the feasibility problem of penalty-based evolutionary algorithms for knapsack problems. In E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, and H. Tijink, editors, *Applications of Evolutionary Computing : Proceedings of EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM*, volume 2037 of *Lecture Notes in Computer Science*, pages 50–60. Springer.

[14] **Grefenstette, J.J.,** 1992. Genetic algorithms for changing environments. In R. Maenner and B. Manderick, editors, Paralel Problem Solving from Nature 2, pages 137-144 North-Holland.

[15] **GSL.** GNU Scientific Library, online, http://www.gnu.org/software/gsl/

[16] **Holland, J.H.,** 1973. Genetic algorithms and the optimal allocation of trials. SIAM J. Of Computing, 2 pp. 88-105.

[17] **Jones T.,** 1995. Evolutionary Algorithms, Fitness Landscapes and Search, PhD Thesis, The University of New Mexico.

[18] **Kellerer, H., Pferschy, U., and Pisinger, D.,** 2004. *Knapsack Problems*. Springer.

[19] **Koza, J.R.,** 1992. Genetic Programming. MIT Pres, Cambridge, MA

[20] **Koza, J.R.,** 1994. Genetic Programming II. MIT Pres, Cambridge, MA.

[21] **Lewis, J. Hart, W. and Ritchie, G.,** 1998. A comparison of dominance mechanisms and simple mutation on nonstationary problems. In Eiben, A.E., Back, T., Schoenauer, M. and Schewefel, H.P. editors. Paralel Problem Solving From Nature, volume 1498 of LNSC. Springer, pp. 139-148.

[22] **Louis, S.Z. and Xu, Z.,** 1996. Genetic algorithms for open shop scheduling and rescheduling. In M. E. Cohen and D. L. Hudson, editors, ISCA Eleventh International Conference on Computers and their Applications, pages 99-102.

[23] **Mitchell, M.,** 2002. An introduction to Genetic Algorithms, PHI.

[24] **Morrison, R.W. and DeJong, K.A.,** 1999. A test problem generator for non-stationary environments. In *Congress on Evolutionary Computation*, volume 3, pages 2047–2053. IEEE.

[25] **Morrison, R.W., and De Jong, K.A.,** 2002. Triggered Hypermutation revisited. In proceedings of Congress on Evolutionary Computation, IEEE, pp.1025-1032.

[26] **Pirkul, H.,** 1987. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 34:161–172.

[27] **Raidl, G.R.,** 1999. Weight-codings in a genetic algorithm for the multiconstraint knapsack problem. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 596–603. IEEE Press.

[28] **Ramsey, C.L. and Grefenstette, J.J.,** 1993. Case-based initialization of genetic algorithms. In S. Forrest, editor, Fifth International Conference on Genetic Algorithms, pages 84-91. Morgan-Kauffman.

[29] **Rechenberg, I.,** 1973. Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologishen Evolution. Fromman-Hozlboog Verlag, Stutgart.

[30] **Richter, H.,** 2004. Behavior of evolutionary algorithms in chaotically changing fitness landscapes. In Xin Yao et. al., editor, *Proceedings of Parallel Problem Solving from Nature VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 111–121. Springer.

[31] **OR Library.** Operations Research Library, online, http://people.brunel.ac.uk/~masjjb/jeb/info.html.

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

[33] **Smith, T., Husbands, P., Layzell, P., and O'Shea, M.,** 2002. Fitness Landscapes and Evolvability, Evolutionary Computation, Volume: 10, Issue:1, pp. 1-34.

[34] **Stadler, P.F.,** 2002. Fitness Landscapes In: M. Lässig and A. Valleriani (eds.): *Biological Evolution and Statistical Physics* Springer-Verlag, Berlin, pp. 187-207.

[35] **Stanhope, S.A. and Daida, J.M.,** 1999. Genetic algorithm fitness dynamics in a changing environment. In *Congress on Evolutionary Computation*, volume 3, pages 1851–1858. IEEE.

[36] **Vassilev, V.K., Fogarty, T.C. and Miller, J.F.,** 2003. Smoothness, Ruggedness and Neutrality of Fitness Landscapes: From Theory to Application, Springer-Verlag, pp 3-44.

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