ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A CLASSIFICATION-BASED HEURISTIC APPROACH FOR DYNAMIC ENVIRONMENTS

M.Sc. THESIS

Şeyda YILDIRIM BİLGİÇ

Department of Computer Engineering

Computer Engineering Programme

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A CLASSIFICATION-BASED HEURISTIC APPROACH FOR DYNAMIC ENVIRONMENTS

M.Sc. THESIS

Şeyda YILDIRIM BİLGİÇ
(504151525)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Ayşe Şima ETANER-UYAR

JUNE 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

DİNAMİK ORTAMLAR İÇİN TASARLANMIŞ SINIFLANDIRICI TABANLI SEZGİSEL BİR YAKLAŞIM

YÜKSEK LİSANS TEZİ

Şeyda YILDIRIM BİLGİÇ
(504151525)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Ayşe Şima ETANER-UYAR

HAZİRAN 2019

Şeyda YILDIRIM BİLGİÇ, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 504151525 successfully defended the thesis entitled "A CLASSIFICATION-BASED HEURISTIC APPROACH FOR DYNAMIC ENVIRONMENTS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Ayşe Şima ETANER-UYAR** Istanbul Technical University	.....
Jury Members :	**Assoc. Prof. Dr. Sanem SARIEL UZER** Istanbul Technical University	.....
	Asst. Prof. Dr. Berna KİRAZ FSM Vakif University	.....

Date of Submission : **02 May 2019**
Date of Defense : **13 June 2019**

To my loved ones,

FOREWORD

Special thanks to my advisor, Assoc. Prof. Dr. A. Şima Etaner-Uyar for all support and contributions.

Thanks to my family for their encouragement and support in my life.

June 2019

Şeyda YILDIRIM BİLGİÇ
(Computer Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

CF	**:** Choice Function
CM	**:** Confusion Matrix
EDA	**:** Estimation of Distribution Algorithm
EI-MMS	**:** Environment Identification-based Memory Management Scheme
HM	**:** Hyper-mutation
IE	**:** Improving and Equal
KNN	**:** K-nearest Neighbors
LHC	**:** Local Hill Climbing
MIA	**:** Memory Indexing Algorithm
MLP	**:** Multi-layer Perceptron
MPB	**:** Moving Peaks Benchmark
PBIL	**:** Population Based Incremental Learning
SVM	**:** Support Vector Machine

LIST OF TABLES

Page

LIST OF FIGURES

Page

A CLASSIFICATION-BASED HEURISTIC APPROACH FOR DYNAMIC ENVIRONMENTS

SUMMARY

Most of the state-of-the-art methodologies focus on stationary optimization problems. However, real-world optimization problems often have various types of uncertainties. One source of such uncertainties is time-varying fitness functions. This occurs in dynamic optimization problems which consist of the instance, the objectives and the constraints that may change in time separately or simultaneously. In such dynamic environments, an optimization algorithm must adapt to the changes in the environment and be able to track the optimum. A solver can be considered successful by taking into consideration its adaptation capacity and speed for reacting changes occurs in a dynamic environment.

This challenging task has attracted a wide range of studies over the last decades. Different solution approaches have been introduced for this challenging research topic. It is difficult to demonstrate the superiority of an approach to another since dynamism characteristics can vary in different environments. These characteristics can be categorized according to the frequency of change, the severity of change, the predictability of change and the periodicity of change. An optimization method may be useful in environments with specific change characteristics but may fail for a different environment. Therefore, the characteristic of changes is a crucial issue that needs to be addressed.

Some of the earlier studies focus on understanding the nature of the changes. However, very few of them use the information obtained to characterize the change for designing better solver algorithms.

In this thesis, a classification-based single point search algorithm, which makes use of the characterization information to react differently under different change characteristics, is introduced. The proposed method includes a classification mechanism that allows it to adapt to changes in various type automatically. The mechanisms it employs to react to the changes resemble hyper-heuristic approaches previously proposed for dynamic environments. During each change in the landscape, the proposed algorithm first categorizes the change that occurs. To adapt to those changes, it applies the predetermined steps according to the change category until the next change in the environment.

In the first stage of this thesis, we applied several measures to extract features from the dynamic landscape. These features provide information for the classification process of dynamic environments.

The proposed algorithm is tested using the Moving Peaks Benchmark generator. In the second stage of this thesis, experiments are performed to understand the underlying components of the proposed method. The method is analyzed to have a better comprehension by simply dividing it into separate algorithms. The performances of

the separated algorithms are compared. Also, different combinations and variations of these algorithms have been described and analyzed. As a result of these experiments, it is observed that the best combination is the method presented in this study.

In the third stage of this thesis, we compare the performance of our suggested approach with similar single point search-based hyper-heuristic approaches for dynamic environments in literature. The experimental results are promising and show the strength of the proposed heuristic approach as a dynamic optimization solver. In addition to this experiment, for maintaining dynamism as close as to real life, a test environment with random changes in the dynamic environment is established, and methods are tested in this experimental setup. As a result of this experiment, it has been shown that the adaptation ability of our proposed approach is better than other methods.

In the final stage of this thesis, we run the experiments for different settings to prove the capability of our mechanism. It is observed that the classifier approach maintains its effectiveness in environments with different dynamism characteristics.

Overall, experimental results are promising and demonstrate the appropriateness of the proposed approach as a dynamic optimization method. On the other hand, our method has certain limitations. The fact that the process of generating input data for classifier costs computational complexity directly proportional to the number of agents. However, the method turns that into an advantage by using the best agent as a candidate solution to find the optimum solution.

Moreover, the trade-off between the agent count and the accuracy of the classifier can be examined further to be optimized. Improvements in this aspect will decrease computational complexity and save time to the approach.

Finally, the proposed method is developed as a single point search algorithm. Still, the main idea in this study can be applied to population-based search algorithms.

DİNAMİK ORTAMLAR İÇİN TASARLANMIŞ SINIFLANDIRICI TABANLI SEZGİSEL BİR YAKLAŞIM

ÖZET

Son zamanlarda önerilen eniyileme yöntemlerinin çoğu statik eniyileme problemlerine odaklanmaktadır. Fakat, gerçek dünyadaki eniyileme problemleri çoğu zaman çeşitli belirsizliklere sahiptir. Bu gibi belirsizliklerin bir kaynağı da zamana göre değişen seçilim değeri fonksiyonlarıdır. Bu durum, zaman içinde ayrı ayrı veya eşzamanlı olarak değişebilecek problemin tanımlı değerleri, eniyilemede kullanılan amaç fonksiyonları ve kısıtlardan oluşan dinamik optimizasyon problemlerinde ortaya çıkar. Bu tür dinamik ortamlarda, bir eniyileme algoritması ortamdaki değişimlere uyum sağlamalı ve optimum olanı izleyebilmelidir. Bu durumda bir eniyileme metodu, dinamik bir ortamda meydana gelen değişimlere olan adaptasyon kapasitesi ve tepki verme hızı göz önünde bulundurarak başarılı sayılabilir.

Son birkaç on yıl içinde, bu zorlu görev çeşitli sayıda çalışmaya ilham olmuştur. İlgi çeken bu araştırma konusu için farkı çözüm yaklaşımları ortaya koyulmuştur. Önerilen yöntemlerin birbirinden üstünlüğünü göstermek zordur. Çünkü, dinamizm özellikleri ortamdan ortama değişebilmektedir. Bu özellikler değişim sıklığı, değişim şiddeti, değişimin öngörülebilirliği ve değişimin periyodik olup olmamasına göre kategorize edilebilmektedir. Bir eniyileme yöntemi belirli değişim özelliklerine sahip ortamlarda etkin olabilirken, farklı bir ortam için başarısız olabilir. Bu nedenle, değişim özellikleri dinamik ortamlarda çalışacak efektif bir yöntem tasarımı için ele alınması gereken bir konudur. Bu tezin temelinde de değişim özelliklerini kullanan başarılı bir yaklaşım tasarımı amacı yatmaktadır.

Önceki çalışmalardan bazıları dinamizmin doğasını anlamaya odaklanmıştır. Ancak, aralarından çok azı, dinamizmi karakterize ederek elde edilen bilgiyi daha iyi eniyileme algoritmaları tasarlamak için kullanmaktadır.

Bu tezde, farklı değişim özellikleri altında farklı tepki vermek için karakterizasyon bilgilerinden faydalanan bir sınıflandırma tabanlı tek nokta arama algoritması tanıtılmıştır. Önerilen yöntem, dışarıdan bir müdahale gerekmeksizin farklı özellikteki değişimlere adapte olabilmesini sağlayan bir sınıflandırma mekanizması içermektedir. Değişimlere tepki vermek için kullandığı bu mekanizmalar, dinamik ortamlar için daha önceleri önerilen üst-sezgisel yaklaşımlara benzemektedir.

Ortamdaki her değişim esnasında, önerilen algoritma öncelikle oluşan değişimi kategorize eder, değişime uyum sağlamak için bir sonraki değişime kadar değişim kategorisine uygun ve önceden belirlenmiş adımları uygular. Böylelikle arama uzayındaki tek nokta en iyi çözümü aranırken ortamda değişim olduğunda rastgele hareketler yapmak yerine belirli bir süre zarfında en iyi çözümü akıllıca takip etmesini kolaylaştıracak adımları uygular. Bu nedenle algoritma hızlı adapte olabilen genelleştirilmiş bir eniyileme çözümüdür.

Daha önceki çalışmalarda dinamik ortam değişimlerini kategorize etmek için sunulan etkinliği kanıtlanmış metrikler bu çalışmadaki sınıflandırma yapısında kullanılmak üzere seçilmiştir. Bu metriklere ek olarak basit bir metrik de sunulmuştur. Etkin bir sınıflandırma için arama uzayına hakim olacak ajanlara ihtiyaç duyulmuştur. Bu ajanları dinamik ortama gözcü noktalar olarak verimli şekilde dağıtacak yöntem belirlenmiştir. Bu tezin ilk aşamasında, farklı tipte dinamizme sahip ortamlardan arama uzayını kapsayıcı ajanlar kullanılarak belirlenen metrikler hesaplanmıştır. Sınıflandırıcı modelini geliştirmek için gerekli öznitelikler hesaplanan metrikler ile sağlanmıştır ve elde edilen veri kümesine uygun sınıflandırma algoritması yapılan testler sonucu belirlenmiştir. Sınıflandırıcılar bu testlerin sonucunda doğruluk, hız ve gürbüzlük açısından değerlendirilmişlerdir.

Önerilen algoritma, yapay oluşturulmuş test problemleri (Moving Peaks Benchmark) üzerinde test edilmiştir.

Bu tezde sunulan yaklaşımın analizi için parametrelerinin başarıma etkisi hem sınıflandırma yöntemi hem de genel algoritma için etraflıca incelenmiştir.

Bu tezin ikinci aşamasında, önerilen yöntemi oluşturan bileşenleri anlamak için deneyler yapılmıştır. Yöntemin daha iyi anlaşılması için kendini oluşturan ayrı algoritmalara bölünerek analiz edilmiştir. Ayrılan algoritmaların tek başına başarımı ölçülüp karşılaştırılmıştır. Ayrıca bu algoritmaların farklı kombinasyonları ve varyasyonları tanımlanıp yapılan deneyler sonucunda en iyi performansı veren kombinasyonun bu çalışmada sunulan yöntem olduğu gözlemlenmiştir.

Tezin üçüncü aşamasında, önerilen yaklaşımımızın performansı literatürde dinamik ortamlar için geliştirilmiş benzer yapıdaki tek nokta arama tabanlı üst-sezgisel yaklaşımlarla karşılaştırılmıştır. Sunulan yaklaşımın karşılaştırılan yöntemlerden istatistiksel olarak daha etkin olduğu gösterilmiştir.

Bu deneye ek olarak, gerçek hayata yakın bir dinamizmin yakalanması için dinamik ortamdaki değişimlerin rastgele olduğu bir test ortamı kurulup karşılaştırılan yöntemler bu deney kurulumda test edilmiştir. Yapılan deney sonucunda önerilen yaklaşımımızın adaptasyon yeteneğinin diğer yöntemlere kıyasla daha iyi olduğu gösterilmiştir.

Bu tezin son aşamasında, sunulan mekanizmamızın kabiliyetini kanıtlamak için farklı dinamizm özelliklerine sahip ortamlar için de deneyler yapılmıştır. Bu sayede sınıflandırıcı yaklaşımın farklı dinamizm özelliklerine sahip ortamlarda da etkinliğini koruduğu gözlemlenmiştir.

Deneysel sonuçlar umut vericidir ve önerilen sezgisel yaklaşımın dinamik eniyileme yöntemi olarak uygunluğunu ve gücünü göstermektedir. Bunun yanında yöntemimizin belirli kısıtları da bulunmaktadır. Kullanılan sınıflandırıcı için girdi verisi oluşturma işleminin doğrudan ajan sayısıyla doğru orantılı olarak hesaplama karmaşıklığına mal olduğu söylenebilir. Fakat, yöntemin bütününde ajanlar optimal çözümü bulmada aday çözüm olarak kullanılarak avantaj sağlamaktadırlar. Ajan sayısı ile sınıflandırıcının doğruluğu arasındaki ödünleşim daha optimize hale getirilerek, hesaplama karmaşıklığı ve zamanı açısından yöntemde iyileşme sağlanabilir.

Ayrıca dinamik ortamdaki tepe sayısı değişimi sonucunda modelin yeniden eğitilme ihtiyacını keşfetmek için analizler yapılmıştır. Yapılan analizler sunulan sınıflandırıcının dinamik ortamdaki tepe sayısı parametresinden az da olsa etk-

ilendiğini fakat eğitim kümesi farklı tepe sayısı bulunan ortamlarda alınan veriler ile genişletildiğinde daha genel bir sınıflandırıcı elde edilebildiği gösterilmiştir.

Buna ek olarak, önerilen metot tek nokta arama tabanlı olarak geliştirilmiştir. Ancak, popülasyon tabanlı arama yöntemi olarak da uygulanabilir.

1. INTRODUCTION

Real-world optimization problems are mostly dynamic in nature. The aim of a dynamic optimization method is not just finding a stationary optimum solution but also to track the changing optimum [1]. A solver can be considered successful by comparison to its adaptation capacity and speed for reacting changes occurs in a dynamic environment. There are several mainstream methodologies that are applied in dynamic optimization problems. These methodologies can be categorized into four groups according to Jin Y. et al. [2]. These are the maintenance of diversity, reacting to changes in the environment, making use of memory and lastly using multiple populations.

Every approach may perform differently for a specific type of dynamism in the environment. For the reason that, knowing the type of dynamism can be useful when selecting a proper approach for a dynamic optimization problem. The dynamism in the environment might also change over time and choosing an approach can become even an impossible task. Therefore, we must think a mechanism capable of reacting different type of landscape changes during a run. There are some studies that focus on representing different environments for adapting to changes. There are also prior studies about the characterization of dynamic environments [3–5]. Branke et al. [6] proposed a number of measures to characterize the nature of a change. The approach proposed in our study expands this previous work by using the information about the change characteristic for building a more effective solver. The novelty of this work lies in making use of the properties of the change for adapting to a changing environment continuously. In this thesis, we focus on creating a classification-based single point search approach for dynamic optimization problems. The method proposed in this study uses the characteristics of the change for adapting to the changing environment.

In the first stage of this thesis, we applied several measures to extract features from the dynamic landscape. These features provide information for the classification process of dynamic environments.

In the second stage of this thesis, experiments are performed to understand the underlying components of the proposed method. The method is analyzed to have a better comprehension by simply dividing it into separate algorithms.

The mechanisms that we propose in our study to respond to changes are similar to the previously proposed hyper-heuristic approaches for dynamic environments. It chooses proper mutation rates for a specific change characteristic like a selection hyper-heuristic chooses low-level heuristics. To this end, we compare the performance of our suggested approach with similar single point search-based hyper-heuristic approaches for dynamic environments in literature in the third stage of this thesis. The experimental results are promising and show the strength of the proposed heuristic approach as a dynamic optimization solver. We conduct another experiment for expanding this analysis. In the experiment, for maintaining dynamism as close as to real life, a test setting with random changes in the dynamic environment is established, and compared methods are tested in this experimental setup. As a result of this analysis, it has been shown that the adaptation ability of our proposed approach is better than other methods.

In the last stage of this thesis, we run the experiments for different settings to prove the capability of our mechanism.

The rest of the thesis is organized as follows. The background information and the related work from the literature are detailed in Section 2. The proposed approach is introduced in Section 3. The experimental design, settings and the results are given in Section 4. Finally, the paper concludes in Section 5 with suggestions for future work.

2. BACKGROUND

This chapter aims to give introductory information for the thesis. Firstly, the dynamic environment concept is explained, and the following subjects are mentioned: dynamic optimization problems, the Moving Peaks Benchmark (MPB), measuring performance and the related work.

Comprehensive surveys on dynamic environments can be found in [1, 2, 5, 7–9].

2.1 Dynamic Environments

Real-world optimization problems often have various types of uncertainties. One source of such uncertainties is time-varying fitness functions. An environment is called dynamic or changing in case of the optimization function, the problem instance or some restrictions of the environment change in time separately or simultaneously. Dynamic environments can be described with the characteristics of the changes. These characteristics are presented in [5] as follows:

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

The frequency of change determines how often a change happens in an environment. The severity of change is a criterion that defines the intensity of a change. The predictability of change controls the similarity between changes. Lastly, the cycle length/cycle accuracy is used for describing the cyclic behavior of an environment. These measures define the cycle of occurrence the similar states in an environment and how similar these states. The characteristics of the change decide the dynamic features of an environment and create various type of dynamism in the environment. An optimum value may change over time because of this dynamism. In order to

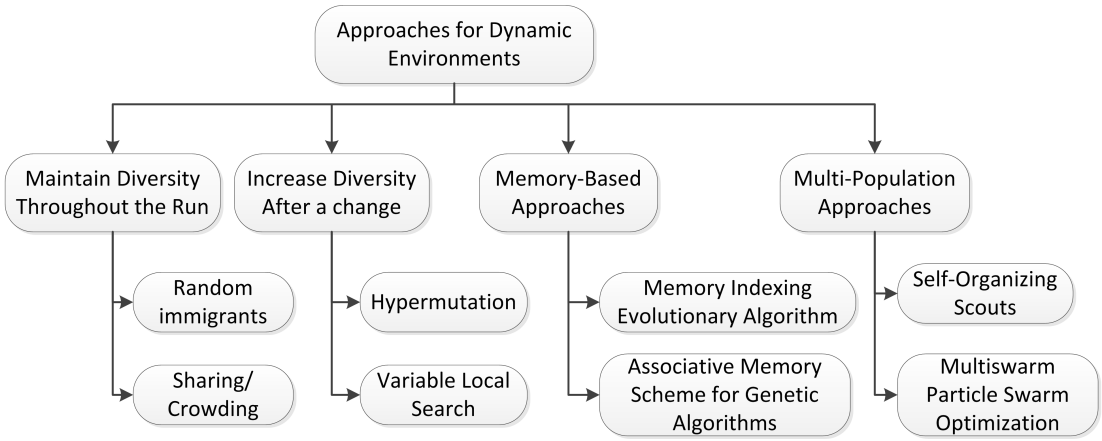

Figure 2.1 : Four main approach groups for dynamic optimization problems.

handle dynamism, unlike a stationary optimization problem, an optimization method for dynamic environments must be capable of tracking the changing optimum [1]. The simplest way to make a solver operate in a dynamic environment is to consider each stationary state between changes as separate optimization problems. This approach can work with dynamic environments that have severe changes. However, most of the cases, good solutions are not very far from the previous ones. Therefore, transferring some information between the following states may be a more useful approach. An optimization method can be considered successful by comparison to its adaptation capacity and speed for reacting changes occurs in a dynamic environment.

There are several mainstream methodologies that are applied in dynamic optimization problems. These methodologies can be categorized into four groups according to Jin Y. et al. [2] (see Figure 2.1). These are the maintenance of diversity [10–12], reacting to changes in the environment [13–15], making use of memory [5, 16–21] and lastly using multiple populations [22–27].

The algorithms that practice the principle of the maintenance of diversity do not take particular actions when a change occurs. The algorithms aim to maintain diversity high through the run to avoid convergence. Introducing randomly generated individuals to a population called Random Immigrants was proposed by Grefenstette [10] is an example to this group.

Another group of approach focuses on increasing exploration of solution space after a change is detected. Hyper-mutation [13] is an example of these approaches. After every change in an environment, the hyper-mutation method increases the mutation

rate in genetic algorithm drastically for some number of generations. Another example is the extended compact genetic algorithm [14] that keeps diversity in the population by doing random restarts of the population at every change.

Multi-population algorithms use several populations to find and trace multiple optima in the search space such as self-organizing scouts [22] and multiswarm particle swarm optimization methods [23, 25].

In memory-based approaches, results produced in earlier search steps can be useful later on, such as storing good solutions in memory to reuse them again. The storing procedure can be done with implicit or explicit memory mechanisms [16]. The use of memory mechanisms is more functional when cyclic changes occur in an environment, or the previous environment resembles the new environment.

2.1.1 Dynamic optimization problems

Various approaches have been introduced to solve different types of dynamic optimization problems, up to this date. Benchmark problems are needed for comparison among different methods. There are different kinds of commonly used benchmark problems. Most of the studies are done on syntactically-generated benchmarks [4, 6, 28–31], where the complexity of the problem and the level of dynamism in an environment is controllable [1]. There are also benchmarks that are real-world dynamic optimization problems [21, 32–35].

The MPB is one of the commonly used synthetic problems [1]. The MPB is a test benchmark for dynamic optimization problems, created by Branke [4]. The MPB is used in this study for examining the performance of the proposed algorithm and comparing it with similar hyper-heuristic approaches from the literature. Also, in the dataset creation process of this study, we extract features of various landscapes that are generated by the benchmark. The MPB is suitable for this study since it presents similar characteristics like in real-world problems [6] as well as it has several parameters for creating several types of dynamic environments.

The benchmark generates dynamic landscapes with a number of peaks; every change in the environment creates differences in heights, widths, and locations of each peak.

The generated dynamic landscapes with cone-shaped peaks can be defined with the following Equation 2.1:

$$F\left(\vec{x},t\right) = \max\left\{B\left(\vec{x}\right), \max_{i=1..m}\left\{H_i\left(t\right) - W_i\left(t\right) * \sqrt{\sum_{j=i}^{dim}\left(x_j - X_{ij}\left(t\right)\right)^2}\right\}\right\} \tag{2.1}$$

where $B\left(\vec{x}\right)$ is the base function, dim is the number of dimensions of the landscape, m is the number of peaks and X_{ij} is the position of the peaks in each dimension. Each peak has its heights and widths which are H_i and W_i.

In the moving peaks problem, the task of a solver algorithm is to trace the highest peak in a landscape of various peaks while the landscape is periodically changing. The MPB has several parameters for creating different scenarios. In every environmental change, the height, width and the location can change according to those parameters. The frequency of change is determined as numbers of evaluations available between the changes. When a change occurs, the heights and the widths of the peaks are changed by adding Gaussian variables to current states. The peaks also move by a shift vector $\vec{v}_i$. The changes in the landscape can be formulated with the following Equation 2.2, Equation 2.3, Equation 2.4, Equation 2.5 and Equation 2.6:

$$\rho \varepsilon N\left(\mu, \sigma^2\right) \tag{2.2}$$

$$H_i\left(t\right) = H_i\left(t-1\right) + heightSeverity * \rho \tag{2.3}$$

$$W_i\left(t\right) = W_i\left(t-1\right) + widthSeverity * \rho \tag{2.4}$$

$$\vec{X}_i\left(t\right) = \vec{X}_i\left(t-1\right) + \vec{v}_i\left(t\right) \tag{2.5}$$

$$\vec{v}_i\left(t\right) = \frac{s}{\left|\vec{r} + \vec{v}_i\left(t-1\right)\right|}\left(\left(1-\lambda\right)\vec{r} + \lambda\vec{v}_i\left(t-1\right)\right) \tag{2.6}$$

Here, ρ is a random value which is produced from Gaussian distribution of $N(\mu, \sigma^2)$ where μ (mean) is set to 0, and σ (standard deviation) is 1. The shifting vector $\vec{v}_i\left(t\right)$ is determined according to the previous shifting vector $\vec{v}_i\left(t-1\right)$, a random shift vector $\vec{r}$, λ which is the correlation coefficient and parameter s. The λ parameter ($0 \leq \lambda \leq 1$) defines the predictability of the changes in the shift vector $\vec{v}_i$. When the lambda is set to 1, the movement directions of the peaks depends on the direction of the previous movement. Whereas when the lambda is 0, the movement directions are decided

Table 2.1 : Severity settings of different change scenarios which create different types of changes in environments using the MPB.

Scenario	s	$heightSeverity$	$widthSeverity$
scenario-1a	1.0	0	0
scenario-1b	5.0	0	0
scenario-2a	0	1.0	0
scenario-2b	0	5.0	0
scenario-3a	0	0	0.5
scenario-3b	0	0	1.0
scenario-4a	1.0	1.0	0.5
scenario-4b	5.0	5.0	1.0

randomly. The severity of the change in the height, width, and location of each peak is controlled by the fixed parameters of $heightSeverity$, $widthSeverity$ and s respectively.

A 2-dimensional landscape with five peaks is produced with the MPB to illustrate the effect of changes one by one and also together. All scenarios are listed in Table 2.1. Figure 2.2 a, d, g and j shows the initial state of the peaks at time t. Each scenario is applied for ten following changes individually to the initial state. The results are present with their scenario names as a caption.

2.1.2 Performance evaluation criteria

Performance evaluation is essential for understanding the effectiveness of different approaches over changing environments. Several measures are for problem-specific as well as general purposes. In stationary optimization problems, the best solution obtained by an approach at the end of its run can be a meaningful performance criterion. Another criterion can be run time or usage of memory until the best solution is achieved. However, in case of changing environments, these measures are not enough to judge an approach. There are additional aspects of a solver must be considered such as adaptation capacity and speed for reacting changes.

In the case of dynamic optimization, the performance of the solver can be measured with the online and offline performance measures which are proposed by De Jong [36] [5]. For judging how good the performance, the online performance is described as an average fitness of all evaluations from a run. The offline performance is defined as an average fitness of all best solutions obtained so far since the last change. Unlike online performance, offline performance considers only the fitness of the best solutions.

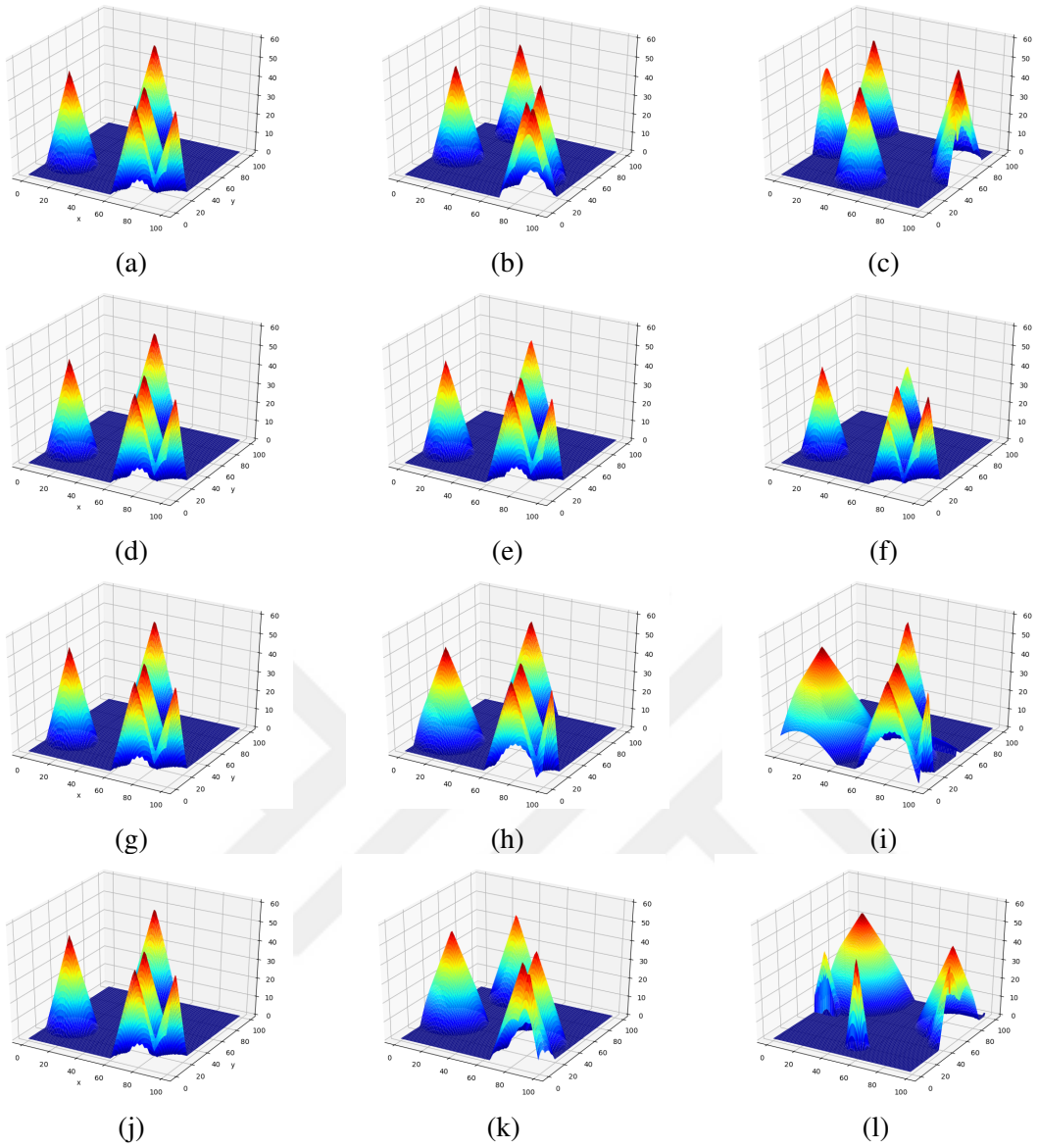

Figure 2.2 : (a), (d), (g) and (j) initial state. Results of change scenarios: (b) scenario-1a. (c) scenario-1b. (e) scenario-2a. (f) scenario-2b. (h) scenario-3a. (i) scenario-3b. (k) scenario-4a. (l) scenario-4b.

Another measure, the offline error is based on offline performance. One of the commonly used performance measure, the offline error [5], is described as the average of the errors of the best solutions found so far since the last change of the environment.

Each candidate solution (individual) can be evaluated with their distance to the optimum as given in Equation 2.7. In order to calculate the error value of a candidate solution $\vec{x}$ at time t, the current optimum value must be known.

$$error\,(\vec{x},t) = |F\,(optimum,t) - F\,(\vec{x},t)| \tag{2.7}$$

where $F(optimum,t)$ is the fitness of the global optimum whereas $F(\vec{x},t)$ is the fitness value of a candidate solution $\vec{x}$.

Every evaluation step, the offline error is defined with the following Equation 2.8 and Equation 2.9 :

$$offlineError = \frac{1}{T_{evaluation}} \sum_{t=1}^{T_{evaluation}} \left(error(x_{\vec{best}},t)' \right) \tag{2.8}$$

$$error(x_{\vec{best}},t)' = min\{error(x_{\vec{best}},\tau), error(x_{\vec{best}},\tau+1), ..., error(x_{\vec{best}},t)\} \tag{2.9}$$

Here $error(x_{\vec{best}},t)'$ is the error values of the best solutions $x_{\vec{best}}$ found so far between the last change and a considered time t and $T_{evaluation}$ is the evaluation count.

Excellent tracking of the optimum means lower overall offline error values even closer to zero for a solver algorithm.

Moreover, there are some other measures like Collective Mean Fitness [37] and Mean Fitness Error [19]...etc.

In this thesis, the offline error measure is used in all experiments since the MPB provides the optimum value at any given time.

Performance measure results alone are not enough for a proper comparison. A fair comparison of different methods needs statistical tests for empirical confidence that validates an approach performs better than another one. Some of the widely used statistical analyses are Student t-test, Mann-Whitney U test, Wilcoxon's non-parametric test, ANOVA and Tukey HSD test. In this thesis, the One-way ANOVA test at a confidence level of 95% is used. Statistical results of compared algorithms are expressed with symbols that are S+', 'S-', '>' and '<' indicating statistically better, statistically worse, better and worse respectively.

2.2 Related Works

Some studies focus on representing different environments for adapting to changes.

Explicit memory mechanism using the associative memory concept is investigated for dynamic optimization problems. In the associative memory schemes, good solutions are stored with associated environmental data. In this manner, when a new environment

which is similar to a collected environment instance occurs, the linked good solutions can be employed for generating new solutions [38]. The associative memory scheme has been developed for Population-based Incremental Learning (PBIL) algorithms for dynamic environments [39]. The associative memory scheme has also been introduced into Estimation of Distribution Algorithms (EDAs) which was firstly proposed by Mühlenbein et al. [40]. The Memory Indexing Algorithm (MIA) indexes environments with problem-specific knowledge such as the quality of the environment [17]. MIA applies EDAs and hyper-mutation mechanism together to react on environmental changes. If the environment is similar to previously seen ones, MIA uses a distribution array to initialize the population. Peng et al. [41] have presented an environment identification-based memory management scheme (EI-MMS). The EI-MMS uses, the probability models to characterize and store the landscape of dynamic environments. Then, it applies this stored data to adjust EDAs in compliance with environmental changes. The study proposed an environment identification method for finding best-fitting memory elements for new environments.

There are also prior studies about the characterization of dynamic environments [3–6, 42] which is one of the objectives of this thesis. Branke [5] introduced the frequency of change, the severity of change, the predictability of change and the cycle length/cycle accuracy criteria for categorization. Duhain et al. [42] reviews the previous methods used for characterization and suggests a unified classification system. In the paper, new behavioral classes; static environments, progressively changing environments, abruptly changing environments and chaotic changing environments are proposed.

Branke et al. [6] proposed a number of measures to characterize the nature of a change. The approach proposed in this thesis expands this previous work by using the information about the change characteristic for building a more effective solver.

Hyper-heuristics are new search methodologies that have proven to be effective solvers in dynamic environment optimization. Hyper-heuristics are high-level methods that operate on top of a set of heuristics. There are two main categories of hyper-heuristics [43]: heuristic selection and heuristic generation approaches. One of the previous studies on hyper-heuristics in dynamic environments, Kiraz et al. [44] shows the appropriateness of selection hyper-heuristics as solvers in dynamic optimization problems. The empirical results of the study demonstrate that selection

hyper-heuristics are capable of reacting and tracking well in different types of changes. The mechanisms that we propose in our study to respond to changes are similar to the previously proposed hyper-heuristic approaches for dynamic environments. It chooses proper mutation rates for a specific change characteristic like a selection hyper-heuristic chooses low-level heuristics.

3. PROPOSED METHOD

We introduce here a classification-based single point search algorithm for solving dynamic optimization problems. Single point search is a local search method that operates on a particular solution candidate in search space by exploring its neighborhood with a collection of moves.

The steps of this study can be summarized as follows:

- Collect data from dynamic environments by calculating measures for dataset creation
- Build a classifier model to classify different type of landscape changes
- Modify a single point search algorithm by adding a new mechanism capable of reacting different kinds of landscape changes during a run

The proposed algorithm consists of three components: change characterization, mutation rate selection, and process of using sentinels as solutions. Briefly, at the change characterization step, a learning mechanism is employed to classify the landscape change with the data from the measure calculations. Then at the mutation phase, the algorithm draws a standard deviation value within the given predetermined interval for that class like a hyper-heuristic approach chooses a low-level heuristic. Finally, sentinels that are used for measure calculations, also contribute as solutions if the best sentinel is better than the current solution. The algorithm of the proposed method is given in Figure 3.1.

3.1 Change Characterization

The ability to characterize the landscape changes of a dynamic environment provides valuable information for adapting to those changes. In change characterization section of this study, agents called sentinels, points in the search space, are used for computing measures that represent the characteristics of a change. Next, a model has been

Algorithm 1 Classification-based single-point search algorithm

```
1: sentinelPlacement()
2: individual ← createRandomIndividual()
3: while numberOfIterations do
4:     if environmentChanged then
5:         mutationStepCount ← 0
6:         measures ← calculateMeasures(sentinels)
7:         bestSentinel ← findBestSentinel()
8:         if bestSentinel.fitness is better individual.fitness then
9:             copySentinel(individual, bestSentinel)
10:        end if
11:        classResult ← findClass(model, measures)
12:    end if
13:    if classResult.isDetermined() then
14:        if mutationStepCount <= maxNumOfMutation then
15:            stdev ← selectStdev(classResult)
16:        else
17:            stdev ← defaultStdev
18:        end if
19:    end if
20:    mutant ← mutate(individual, μ = 0, stdev)
21:    mutationStepCount ← mutationStepCount + 1
22:    if mutant.fitness is better than or equal individual.fitness then
23:        individual ← mutant
24:    end if
25: end while
```

Figure 3.1 : Classification-based single-point search algorithm.

developed for change classification by using the data from dynamic environments which have different change characteristics. The change characterization steps are detailed in the following subsections.

3.1.1 Sentinel placement

The Sentinel Placement method proposed by Morrison [45] is initially a change detection method. The algorithm places sentinels in the landscape better than randomly spread agents [46]. The sentinels take over the search space since they are distributed evenly in higher dimensions. Therefore, in order to characterize changes efficiently, instead of using randomly spread agents, we use the Sentinel Placement method to position the sentinels.

The heuristic sentinel placement algorithm works as follows for N-dimensional space:

- Scale and offset all search space dimensions from 0 to 1.

- Randomly place the initial sentinel point x_i, for i equals 1 to N.
- Create other sentinel points with Equation 3.1 and Equation 3.2.

$$x_{i,p} = mod1\left[x_{i,p-1} + Z_i\phi\right] \tag{3.1}$$

$$Z_i \varepsilon \{41,43,47,59,83,107,109,173,311,373,401,409,...\} \tag{3.2}$$

Here p equals 1 to *sentinel count*, ϕ is golden ratio equals to $(\sqrt{5}+1)/2$, $mod1$ is the modulo 1 operation and Z_i values are set according to heuristic rules specified in Morrison's study [45].

There are two different agent placement examples in Figure 3.2. In two examples, 50 agents are distributed in 3-dimensional (100x100x100) search space. While sub-figures a, c and e show the result of the random distribution for each dimension pair, sub-figures b, d and f show the result of the sentinel placement method for each dimension pair. These two examples demonstrate the difference between the two approaches. According to the experiment, the sentinel placement method has obvious superiority compare to a random placement. It is more effective for landscape exploration. If we want to use a small number of agents and higher multi-dimensional space, the advantage of the sentinel placement method will become more evident. However, the prime numbers, Z_i, used in Equation 3.1, are known to produce reasonably uniform distributions for 200 agents in dimensions up to 12 [45].

3.1.2 Measure calculation

In this thesis, several measures proposed by Branke et al. [6] have been used for representing the nature of a change. We also include a simple measure called fitness difference to the set of measures. We employed measures to extract features from the dynamic landscape. These features provide information for the classification process of dynamic environments. Calculations of the measures have been outlined below.

- Fitness Correlation: The fitness correlation is determined by fitness correlation between the fitness of sentinels during the change steps [6]. The fitness correlation is calculated using Equation 3.3.

$$FC = \frac{\sum_{i=1}^{count}(s_{before_i} - \overline{s_{before}})(s_{after_i} - \overline{s_{after}})}{\sqrt{\sum_{i=1}^{count}(s_{before_i} - \overline{s_{before}})^2 (s_{after_i} - \overline{s_{after}})^2}} \tag{3.3}$$

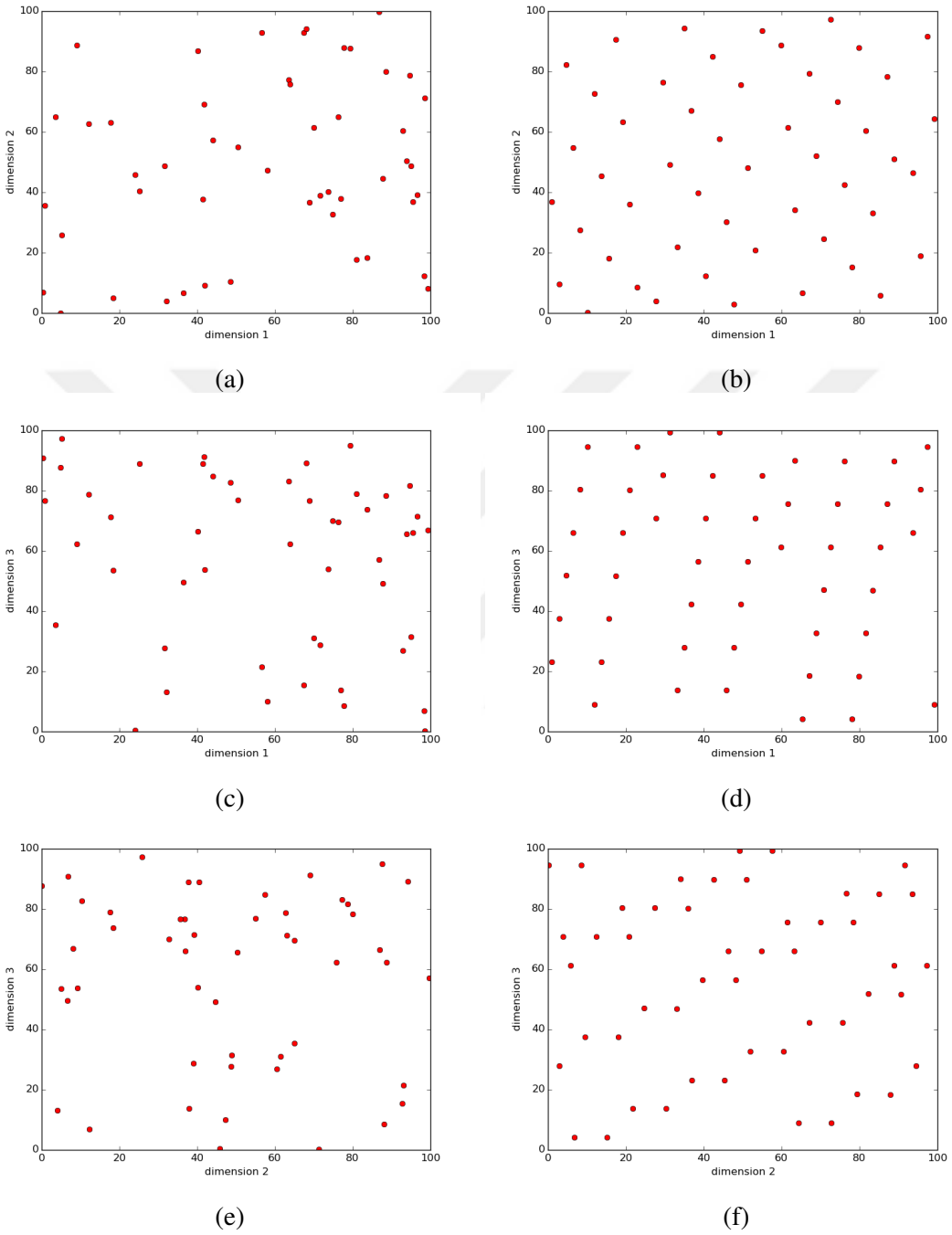

Figure 3.2 : Fifty randomly placed agents: (a) dimension 1,2. (c) dimension 1,3. (e) dimension 2,3. versus fifty agents placed by the sentinel placement method: (b) dimension 1,2. (d) dimension 1,3. (f) dimension 2,3.

Here s_{before} is a set of fitness values of sentinels before a change, s_{after} is a set of fitness values of sentinels after a change and *count* equals sentinel count.

- Change Severity: The distance between the optimum before and after the environmental change is called change severity [6]. For most of the cases, finding the optimal solution at each environmental state won't be possible. Hence, the estimated change severity measure is used. Without knowing the real optimum, the best solution is provided by performing local hill-climbing (LHC) to the sentinel points is used as an optimum. Euclidean distance is used for distance calculation, and the change severity is normalized according to the size of the search space.

- LHC Fitness Correlation: The correlation of fitness after LHC before and after landscape chance is also measured [6]. The measure is formulated as in Equation 3.3 where s_{before} and s_{after} are sets of fitness values that obtained by LHC.

- Fitness Difference: The fitness differences of sentinels before and after the change are calculated. Then normalized fitness differences are averaged to use as a fitness difference measure. The fitness difference is calculated as in Equation 3.4:

$$FD = \frac{\sum_{i=1}^{count} \left| s_{i_{before}} - s_{i_{after}} \right|}{count} \tag{3.4}$$

  where s is fitness of sentinel, $before$ and $after$ indicate if it is from before or after a change and *count* equals sentinel count.

The proposed algorithm is evaluated on the MPB. The MPB generator provides dynamic environments with a variety of different change characteristics.

Euclidean distance is used if a measure requires distance calculations between points. Where LHC is needed, a stochastic hill climbing algorithm is applied for 100 iterations, and each LHC process starts from coordinates of the sentinels. The coordinates and fitness values obtained at the end of the hill climbing processes are stored for measure calculations. However, this process does not change the initial coordinates of the sentinels. In this way, the sentinels start from the same coordinate points for every calculation processes. The pseudo code of the stochastic hill climbing algorithm is given in Figure 3.3. In the algorithm, the random neighbor is produced with using

Algorithm 2 Stochastic hill climbing algorithm

```
1: while numberOfIterations do
2:     candidateSolution ← createRandomNeighbor(currentSolution)
3:     if candidateSolution.fitness is better currentSolution.fitness then
4:         currentSolution ← candidateSolution
5:     end if
6: end while
7: return currentSolution
```

Figure 3.3 : Stochastic hill climbing algorithm.

gaussian mutation with a distribution of $\mathcal{N}(\mu, \sigma^2)$ where μ (mean) is set to 0, and sigma (standard deviation) is 1. The gaussian mutation operation is performed for each dimension of the solution candidate and described in section 3.2.

Measure computing after each change causes some delay and more fitness evaluations. However, the hill climbing process can be parallelized to deal with the overhead of extra evaluations. Therefore we count that each LHC operation costs one evaluation, losing one evaluation for each sentinel at LHC step is given up for employing classification.

3.1.3 Dataset creation

A dataset consists of calculated measures as features and class information of changes, collected using the MPB with three different severity levels. In the experimental settings section 4.1.2, three different severity settings are listed in detail. These levels are labeled as class information for each sample. (Frequency setting of the MPB makes no difference for dataset creation. Since we classify dataset according to the severity properties.)

Firstly, the severity settings of the dynamic environment are initialized for each specific level. Then for each environmental setup, before the MPB runs, sentinel placement method is employed to the environment. In the first stationary state of the environment, sentinels prepare data from the landscape. This data will be used as information of the before the change since the calculation of measures requires information from the landscape before and after the change in the environment. During the run when a change occurs in the landscape, sentinels prepare data from the altered landscape.

(MPB provides information whenever a change occurs.) This data is also stored to use as a previous state of the environment if a new change happens.

Finally, measures are calculated by using both before and after change data as explained in measure calculations section 3.1.2. The results are recorded as features of a sample. This process repeatedly continues until a sufficient amount of samples is gathered. Finally, the four-featured dataset with 407163 samples has been obtained.

A small part of the dataset is shown in Table 3.1 to help with visualization.

Table 3.1 : A sample of the dataset with four features and class information.

FITNESS DIF	CHANGE SEV	FITNESS CORR	LHC FITNESS CORR	CLASS
0.4124	0.0092453	0.995381	0.915427	LS
3.395	0.214257	0.603657	0.566644	HS
3.7838	0.186679	0.676397	0.659543	HS
2.3847	0.0640422	0.831933	0.763744	MS
...	...	...	...	...

The dataset creation process is also illustrated with a flowchart in Figure 3.4 Here, *Stop Criteria* is the number of iterations to collect sufficient data.

3.1.4 Classifier model

The model building process has two stages, the training stage, and the model evaluation stage. First, a classifier is trained with the given dataset, the output of the dataset creation. Then, the performance of the built classifier is tested with different performance measures such as accuracy, recall, and precision. In Figure 3.5, a diagram of this process is presented.

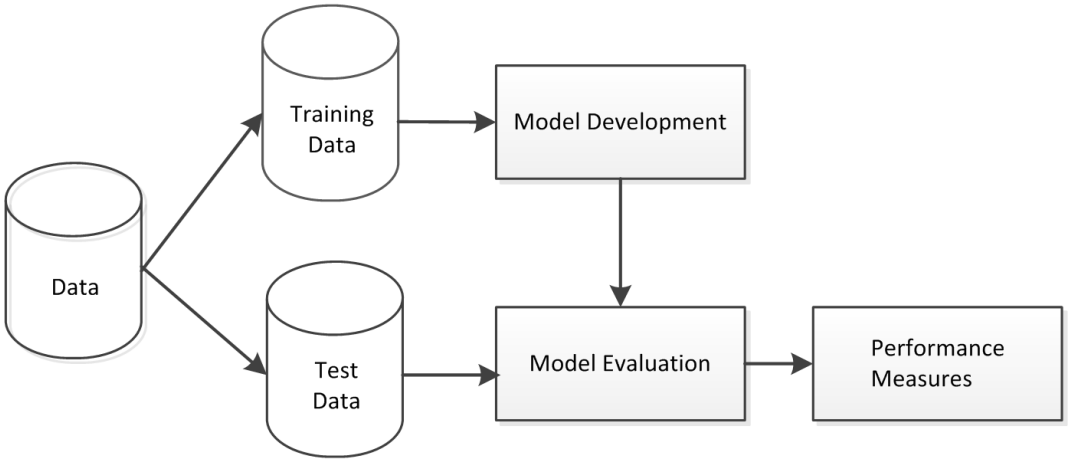

Figure 3.5 : Model building process.

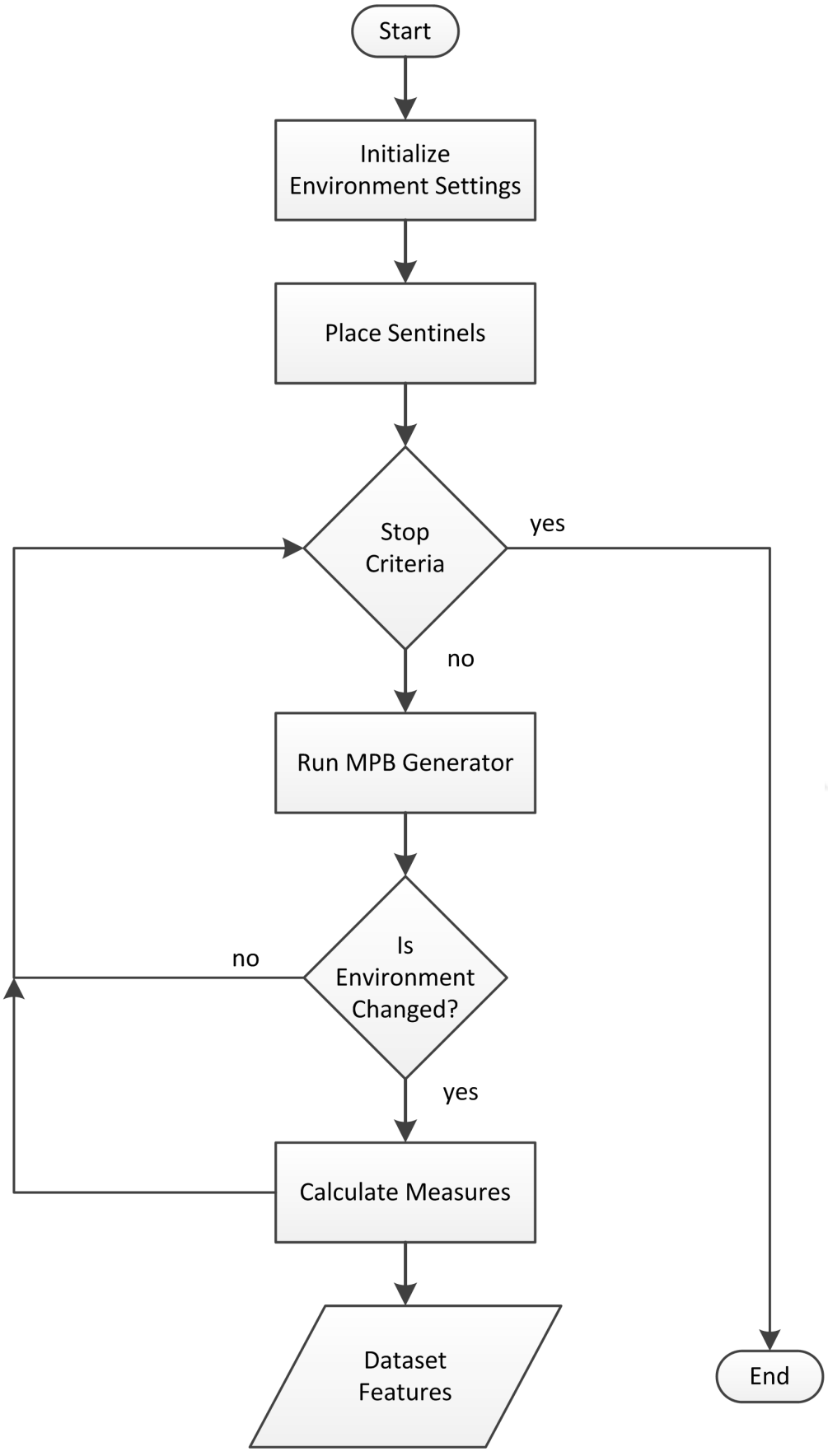

Figure 3.4 : Flowchart of the dataset creation process.

In this study, the Random Forest Algorithm [47] is used for dynamic environment classification. We trained the Random Forest model using the scikit-learn [48, 49] with the collected dataset. The number of decision trees in the forest is 10, and the maximum depth of the tree of the classifier is selected as 8. In our experiments with the collected data, the classifier gives excellent results in terms of accuracy and speed compared to other classification methods (K-Nearest Neighbor (KNN), Gradient Boosting, Multi-layer Perceptron (MLP), and Support Vector Machine (SVM) Classifiers) as explained in the section 4.1.3.

3.2 Mutation Rate Selection

The sentinel placement is performed at the beginning of the proposed method. The algorithm starts with a single feasible solution which is created randomly. During stationary states, the individual goes through gaussian mutation with a distribution of $\mathcal{N}(\mu, \sigma^2)$ where μ (mean) is set to 0, and sigma (standard deviation) is the default value of 10 and creates a mutant. It is assumed that the algorithm is aware of the time when a change in the environment happens. Therefore, when there is a change, measure calculations are performed before the classification step as shown in line 6 of the proposed algorithm in Figure 3.1. Then, four calculated features which present last landscape change go through classification with our model to determine its class as simplified in Figure 3.6.

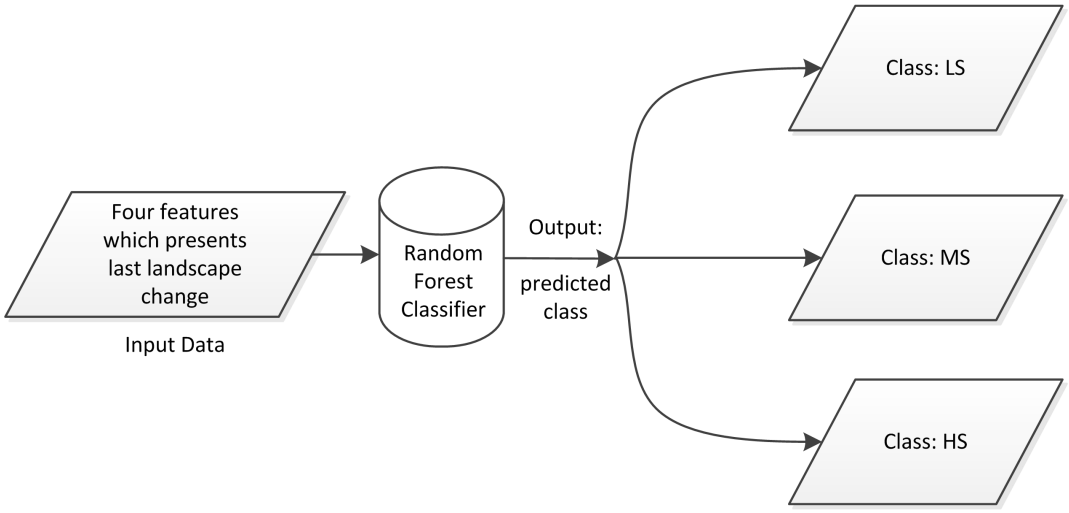

Figure 3.6 : The landscape change classification process.

At the gaussian mutation step, instead of the default standard deviation, the algorithm selects a standard deviation randomly and uniformly in the determined value ranges

for the specific class as shown in line 15 of the proposed algorithm in Figure 3.1. The intervals are listed in the parameter tuning section. This standard deviation selection according to the class of a change is used during defined sequential mutation steps ($maxNumOfMutation$) after every landscape changes. Then the algorithm goes back to applying the default standard deviation. The gaussian mutation operation is performed as provided in Equation 3.5.

$$\vec{m}_i = \vec{x}_i + N(\mu, \sigma^2) \tag{3.5}$$

Here, $\vec{m}$ is the mutant and $\vec{x}$ is the current individual respectively. The mechanisms that the method uses to respond to changes resemble the previously proposed hyper-heuristic approaches for dynamic environments. It chooses appropriate mutation rates for specific change characteristics like a hyper-heuristic chooses a low-level heuristic.

3.3 Using Sentinels as Solutions

In the measure calculation step, sentinel points go through an LHC process right after each change. In LHC process, every sentinel points attempt to reach optimum value like a single point doing a local search. End of the hill climbing, sentinels will have information on better solution points in the search space. Therefore, those improved points can be considered as decent solutions.

As mentioned before, hill climbing costs several evaluations directly proportional to the sentinel count. However, the proposed approach turns that into an advantage by using the best sentinel information. After the LHC process, the algorithm selects the best solution among the sentinels and replaces the current solution with the best sentinel if the sentinel is better than the current solution as shown in line 9 of the proposed algorithm in Figure 3.1.

4. EXPERIMENTS AND RESULTS

A three-phase experiment is designed to evaluate the performance of the proposed method. First phase is for parameter tuning, second phase for analyzing the components of the proposed method and the last phase for comparison with other methods in literature.

4.1 Experimental Design

4.1.1 The components of the proposed method

Some of the experiments focus on the component analysis of the proposed method. For these experiments, the proposed algorithm is divided into separate algorithms to have better comprehension.

The 'Snt' approach operates as a single point algorithm and uses best sentinel information as a solution after making sentinels LHC. Snt does not classify changes in an environment. Snt method mutates the single point with fixed default standard deviation without using the classification information of changes.

On the other hand, 'Cls' method classifies changes and uses the classification results of changes when deciding a mutation rate after every change in an environment. Cls make use of sentinels for change classification. However, it does not use the best sentinel as a solution.

Moreover, to test the classification mechanism (Cls), the Cls method is altered as a typical hyper-mutation method. Therefore at the mutation step, the altered method, 'RandHM' method, selects a random standard deviation from the complete set of standard deviations without knowing the class information.

Furthermore, the combination of Snt and RandHM methods is included as the 'Snt+RandHM' method. Snt+RandHM approach operates as Snt method; it uses the best sentinel information as solutions. But whenever a mutation operation is

performed instead of using fixed default standard deviation, Snt+RandHM chooses a random standard deviation from the complete set of standard deviations like RandHM approach.

Lastly, our complete proposed method is called the 'Cls+Snt' approach, which is a combination of both Snt and Cls methods. Cls+Snt method applies the class information of changes for mutation rate selection and also after each LHC of sentinels, it replaces the current solution with the best sentinel if the sentinel is better than the current as explained in detail in the proposed method section.

Figure 4.1 shows the components of the proposed method and their different combinations. In the following sections, these short forms are used when referring to the components.

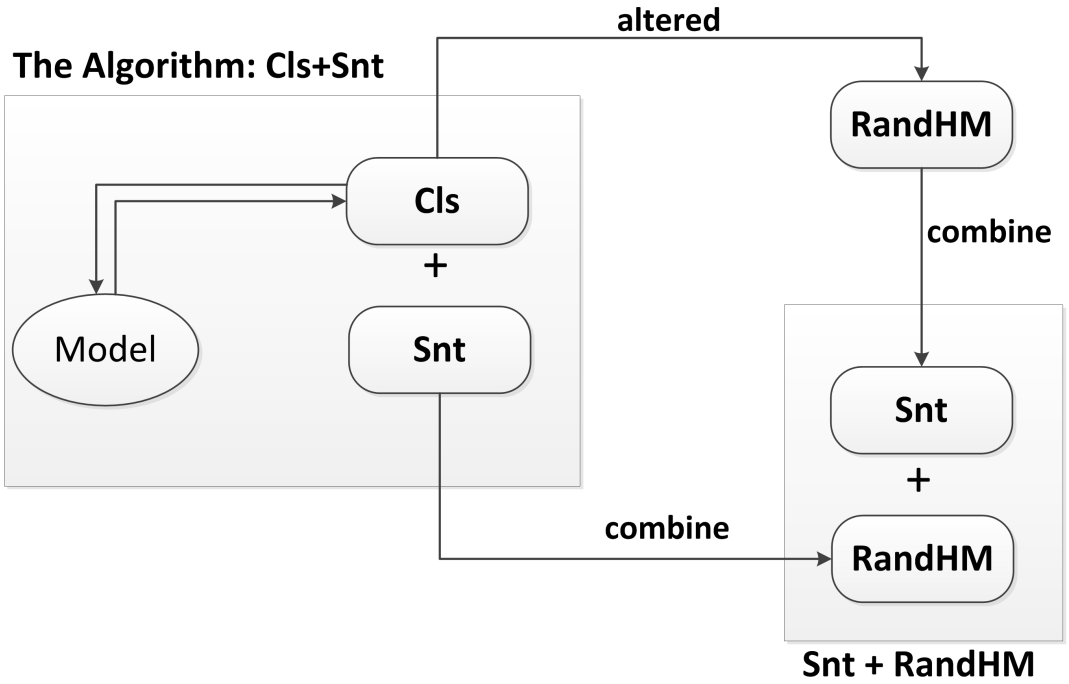

Figure 4.1 : The components of the proposed method and their different combinations.

4.1.2 Experimental settings

The performances of the compared approaches are examined under three separate severity level classes with three different change frequency pairs. The performances of the methods are compared using offline error [5]. One-way ANOVA test is used for detecting the significance of the results.

Each experiment is run 100 times for the given settings with 20 changes in the environment. Thus, there are 21 following stationary stages for each run. The number of iterations of a run is calculated as in Equation 4.1. Also, the first stationary state of the runs is not included in the offline error calculation.

$$numberOfIterations = (numberOfChanges + 1) * changePeriod \tag{4.1}$$

Table 4.1 gives the parameters settings of the MPB and the severity levels are listed in Table 4.2.

- Classes: low severity (LS), medium severity (MS) and high severity (HS).
- Change periods: low frequency (LF), medium frequency (MF), high frequency (HF). The landscape changes every 6006 fitness evaluations for LF, 1001 for MF and 126 for HF. The change periods are determined according to Kiraz's study [44].

Table 4.1 : Parameter settings for the MPB.

Parameter	Setting
Number of peaks	5
Peak heights	[30, 70]
Peak widths	[0.8, 7]
Uniform-height	50
Uniform-width	3
Peak function	Cone
Range in each dimension	[0, 100]
Number of dimensions	5
Basis function	Not used
Correlation coefficient	0

Table 4.2 : Parameter settings for each severity level.

Setting	LS	MS	HS
move severity	1	25	50
height severity	1	5	10
width severity	0.05	0.05	0.05

In this study, some of the experiments are conducted with different severity settings which have more closely categorized classes as listed in Table 4.3. These are also the MPB severity settings of Kiraz's study [44].

In this thesis, statistical results of compared algorithms are expressed with symbols that are S+', 'S-', '>' and '<'. The explanation of these symbols are listed in Table 4.4.

Table 4.3 : Different parameter settings for each severity level.

Setting	LS	MS	HS
move severity	1	5	10
height severity	1	5	10
width severity	0.1	0.5	1.0

Table 4.4 : The meanings of the result symbols.

	Symbol Meanings
S+	Indicates that the first approach is significantly better than the second approach.
S-	Indicates that the first approach is significantly worse than the second approach.
>	Indicates that the first approach is better than the second approach.
<	Indicates that the first approach is worse than the second approach.

4.1.3 Evaluation of classifier model

The experimental settings for evaluating classifier model are listed as follows:

- 80% of the dataset instance is used as training data.
- 20% of the dataset instance is used as testing data.
- 10 fold cross-validation is applied.

For choosing a suitable classifier model for our classification task, we examined a number of algorithms with our data. The algorithms score very close results with the severity settings used in this study from the point of classification accuracy. The results can be observed in Table 4.5.

Table 4.5 : Classifier accuracy for sparse severity settings.

Classifier	Classification Accuracy
Random Forest	0.978
KNN (N=3)	0.968
SVM	0.966
MLP Classifier	0.962
Gradient Boosting Classifier	0.976

Also, the robustness of the algorithms are tested with different severity settings taken from the Kiraz's study [44] (see Table 4.3). Table 4.6 provides the result of the second experiment. Since the different sets of settings for the three classes have closer ranges compared to this study, the classification task becomes more challenging. In this aspect, the Random Forest and the Gradient Boosting algorithms are more robust than

the other techniques. In the end, the Random Forest is chosen because it gives faster results than the Gradient Boosting.

Table 4.6 : Classifier accuracy for close severity settings.

Classifier	Classification Accuracy
Random Forest	0.871
KNN (N=3)	0.755
SVM	0.696
MLP Classifier	0.693
Gradient Boosting Classifier	0.874

Table 4.7 provides ranked features of the model according to the variable importance metric. The variable importance metric illustrates the statistical significance of each feature in the dataset. The feature that ranked first is also the most important feature that affects the model. The variable importance metric calculation can be found in the H2O Flow framework [50].

Table 4.7 : The ranked features of the model according to the variable importance metric.

Rank	Feature
1	Fitness Correlation
2	Fitness Difference
3	Change Severity
4	LHC Fitness Correlation

Here a confusion matrix (CM) which represents the performance of the classifier on the test data is given for further analysis on the model. The CM in Table 4.8 shows the predicted classes versus the actual classes.

Table 4.8 : The confusion matrix of the classifier model.

		Predicted		
		LS	MS	HS
Actual	LS	21812	2	0
	MS	4	29094	1198
	HS	0	706	28713

The numbers in the CM(3x3) can be interpreted as follows:

- CM(1,1): 21812 instances are correctly predicted as LS.
- CM(1,2): 2 instances are predicted as MS but their actual label is LS.

- CM(1,3): No LS instances that is predicted as HS.
- CM(2,1): 4 instances are predicted as LS but their actual label is MS.
- CM(2,2): 29094 instances are correctly predicted as MS.
- CM(2,3): 1198 instances are predicted as HS but their actual label is MS.
- CM(3,1): No HS instances that is predicted as LS.
- CM(3,2): 706 instances are predicted as MS but their actual label is HS.
- CM(3,3): 28713 instances are correctly predicted as HS.

It can be noticed from the observations the model almost perfectly predicts low severity class (LS). Some of the cases it confused medium severity class (MS) with high severity class (HS).

Moreover, precision and recall values are calculated for each class with Equation 4.2 and Equation 4.3 respectively. Precision indicates the proportion of the predicted samples that are correctly predicted. On the other hand, recall indicates the amount of the correctly predicted samples from the samples that should have been predicted in the actual class.

$$precision = \frac{tp}{(tp + fp)} \tag{4.2}$$

$$recall = \frac{tp}{(tp + fn)} \tag{4.3}$$

For example, the false positive (fp) for class LS is the instances that have predicted as LS; however, should have predicted as another class. On the other hand, the false negative (fn) for class LS is the instances that have predicted as another class; but, should have predicted as LS. Finally, the true positive (tp) for class LS is the instances that are labeled as LS and have predicted as LS.

The precision and recall values of each class are provided in Table 4.9

Table 4.9 : Precision and recall values of each class.

Class	Precision	Recall
LS	0.99	0.99
MS	0.97	0.96
HS	0.95	0.97

4.1.4 Parameter tuning

Different components of the proposed method have parameters that require tuning for better performance. These parameters are sentinel count, default standard deviation and standard deviation value ranges which are used in the mutation operation for each class. The parameters are determined experimentally with several preliminary tests and optimized according to their effect on the performance.

Also, the effect of the number of peaks in an environment is investigated in terms of the accuracy of the classifier and performance of the proposed method.

4.1.4.1 Effect of number of peaks

Since the classification model plays an essential role in our approach, we conducted experiments for discovering the need for rebuilding the model if the number of peaks of the dynamic environment changes. Therefore, the effect of using a different number of peaks than the actual environment has, in the classifier training process is investigated.

The experiments are run with the same environment that has ten peaks. In the first test, the classifier model trained with five peaks. On the other hand, in the second test the classifier model, trained with ten peaks, is used. For each analysis, offline errors of the proposed method for each frequency-severity pairs and classification accuracy of the model are measured. In Table 4.10 the results of two experiments are listed.

In Table 4.10, it can be observed that if training data and input data of the classifier have different peak counts, the accuracy of the classifier decreases. Despite the drop in accuracy, the overall performance of the proposed method is not significantly affected. If the dataset is expanded with more data from environments that have various peak counts, the accuracy of the classifier would be higher.

Table 4.10 : Classification accuracy of classifier models and offline error values of the method with standard deviations in the environment that has ten peaks. One is trained with five peaks, and the other is trained with ten peaks.

		Environment has 10 peaks. Model trained with 10 peaks.		Environment has 10 peaks. Model trained with 5 peaks.	
		Offline Error	Classification Accuracy	Offline Error	Classification Accuracy
LF	LS	1.89 ± 0.29		1.91 ± 0.3	
	MS	4.29 ± 0.57		4.4 ± 0.63	
	HS	4.57 ± 0.58		4.56 ± 0.57	
MF	LS	2.09 ± 0.33		2.06 ± 0.29	
	MS	4.8 ± 0.68	98.46	4.85 ± 0.67	96.13
	HS	5.03 ± 0.62		4.98 ± 0.62	
HF	LS	3.54 ± 0.52		3.53 ± 0.46	
	MS	8.62 ± 1.12		8.53 ± 1.08	
	HS	8.96 ± 0.91		8.97 ± 0.88	

Therefore, we train the classifier model with data collected from both five peaks and ten peaks environments, and we test the model in the environment that has ten peaks. Table 4.11 gives the corresponding results of this test. This time, classification accuracy and offline error values are really close as the model that is trained with data obtained from ten peaks environment. This experiment shows us that a model which is design for change classification can cope with different landscapes with various peak numbers.

4.1.4.2 Effect of default standard deviation parameter

In this study, some of the algorithms that are compared, use the default standard deviation parameter. The default standard deviation is a free parameter that affects performance. We analyze this parameter with four different settings. Each algorithm is run for the four different values of the default standard deviation. Statistical significance tests are conducted to decide the best setting of default standard deviation parameter for our proposed method (Cls+Snt). Cls+Snt, Snt, RandHM, and RandHM+Snt algorithms are pairwise compared for four different default standard deviation settings. The ANOVA test results are provided in Table 4.12.

Table 4.11 : Classification accuracy of the model and offline error values of the method with standard deviations in the environment that has ten peaks. Training data obtained from both five peaks and ten peaks environments.

		Environment has 10 peaks. Model trained with both 5 peaks and 10 peaks data.	
		Offline Error	Classification Accuracy
LF	LS	1.87 ± 0.30	
	MS	4.28 ± 0.58	
	HS	4.54 ± 0.53	
MF	LS	2.13 ± 0.36	
	MS	4.76 ± 0.63	97.53
	HS	5.03 ± 0.58	
HF	LS	3.52 ± 0.45	
	MS	8.52 ± 1.07	
	HS	8.93 ± 0.92	

Table 4.12 : The ANOVA test results of methods for different default standard deviation values with different frequency and severity combinations.

Algorithm	Default Stdev	LF			MF			HF		
		LS	MS	HS	LS	MS	HS	LS	MS	HS
Cls+Snt vs Snt	0.3	<	<	<	<	<	S-	S-	<	<
	2.0	S+	>	<	S+	<	<	S+	<	>
	5.0	S+	>	>	S+	S+	>	S+	<	>
	10.0	S+	S+	<	S+	S+	<	S+	>	<
Cls+Snt vs RandHM	0.3	S+	S+	S+	S+	S+	S+	S+	S+	S+
	2.0	S+	S+	S+	S+	S+	S+	S+	S+	S+
	5.0	S+	S+	S+	S+	S+	S+	S+	S+	S+
	10.0	S+	S+	S+	S+	S+	S+	S+	S+	S+
Cls+Snt vs RandHM+Snt	0.3	<	<	<	>	>	>	S+	<	>
	2.0	S+	<	<	S+	<	<	S+	<	<
	5.0	S+	<	<	S+	<	S-	S+	>	<
	10.0	S+	<	S-	S+	>	S-	S+	<	>
Snt vs RandHM	0.3	S+	S+	S+	S+	S+	S+	S+	S+	S+
	2.0	S+	S+	S+	S+	S+	S+	S+	S+	S+
	5.0	S+	S+	S+	S+	S+	S+	S+	S+	S+
	10.0	S+	S+	S+	S+	S+	S+	S+	S+	S+
Snt vs RandHM+Snt	0.3	<	>	>	>	>	S+	S+	>	>
	2.0	<	<	>	<	<	<	<	<	<
	5.0	S-	S-	<	S-	S-	S-	<	>	<
	10.0	S-	S-	S-	S-	S-	S-	<	<	<
RandHM vs RandHM+Snt	0.3	S-	S-	S-	S-	S-	S-	S-	S-	S-
	2.0	S-	S-	S-	S-	S-	S-	S-	S-	S-
	5.0	S-	S-	S-	S-	S-	S-	S-	S-	S-
	10.0	S-	S-	S-	S-	S-	S-	S-	S-	S-

According to the S+ results, the Cls+Snt method performs better when the default standard deviation parameter is set to 10. The Cls+Snt method gives better results than

the Snt method at low severity settings regardless of frequency. Also, the RandHM method falls behind of the Cls+Snt and the Snt methods for all settings.

Table 4.13 : The overall ('S+', 'S-', '>' and '<') counts for different default standard deviation values.

Algorithm	Default Stdev	S+	S-	>	<
Cls+Snt	0.3	10	2	4	11
	2.0	15	0	2	10
	5.0	16	1	5	5
	10.0	17	2	3	5
Snt	0.3	13	0	13	1
	2.0	9	3	5	10
	5.0	9	9	2	7
	10.0	9	11	3	4
RandHM	0.3	0	27	0	0
	2.0	0	27	0	0
	5.0	0	27	0	0
	10.0	0	27	0	0
RandHM+Snt	0.3	9	3	8	7
	2.0	9	3	14	1
	5.0	15	3	7	2
	10.0	17	3	5	2

The overall S+ counts in Table 4.13 confirm that the proposed method, Cls+Snt, performs best when the default standard deviation is set to 10. In this study, the default standard deviation parameter is set to 10 for all approaches that employ the parameter.

4.1.4.3 Effect of sentinel count

The accuracy of the model depends on a set of parameters such as sentinel count, number of peaks and the severity settings of each class. In this section, the sentinel count parameter is investigated for building a better classifier model. The sentinel count effect is shown in terms of the classification accuracy and the offline error that represents the algorithm performance for each frequency-severity pairs in Table 4.14.

The results are indicated that the accuracy values of the classifier are close when the sentinel size is 50 and 100. However, considering the offline error values, when the

Table 4.14 : The effect of the sentinel size for different frequency and severity combinations.

# of sentinel	Frequency	Severity	Classification Accuracy of Random Forest Classifier	Offline Error
		LS		4.1 ± 2.29
	LF	MS		12.58 ± 3.84
		HS		14.65 ± 2.86
		LS		4.18 ± 2.15
10	MF	MS	0.9181	13 ± 3.61
		HS		16.34 ± 2.86
		LS		6.04 ± 1.97
	HF	MS		23.05 ± 4.84
		HS		27.84 ± 4.14
		LS		2.87 ± 1.48
	LF	MS		7.71 ± 1.65
		HS		8.82 ± 1.39
		LS		2.87 ± 1.26
30	MF	MS	0.9584	8.53 ± 2.09
		HS		9.95 ± 1.7
		LS		3.59 ± 1.25
	HS	MS		11.37 ± 2.05
		HS		13.39 ± 1.95
		LS		2.24 ± 0.74
	LF	MS		6.05 ± 1.28
		HS		7.01 ± 1.12
		LS		2.4 ± 1.09
50	MF	MS	0.9662	6.69 ± 1.61
		HS		7.64 ± 1.12
		LS		3.43 ± 1.05
	HS	MS		10.52 ± 1.93
		HS		11.81 ± 1.4
		LS		1.83 ± 0.44
	LF	MS		4.54 ± 0.8
		HS		4.78 ± 0.84
		LS		2.06 ± 0.4
100	MF	MS	0.9785	5 ± 0.71
		HS		5.59 ± 0.85
		LS		3.54 ± 0.53
	HF	MS		9.73 ± 1.56
		HS		10.8 ± 1.32

sentinel size is 100, the method gives better performance. Therefore, as an outcome of the experiments, we selected sentinel count as 100.

4.1.4.4 Standard deviation value ranges for each class

After every landscape changes at the gaussian mutation step that detailed in the proposed method, the proposed algorithm picks a standard deviation value randomly and uniformly from the determined value ranges for each specific class. The specified intervals are defined experimentally. Table 4.15 gives the standard deviation groups for the classes that are used in this study.

Table 4.15 : The standard deviation ranges for each class.

Class	Standard Deviation Range
LS	[0.5, 0.7]
MS	[2.0, 3.0]
HS	[7.0, 9.0]

4.2 The Experiments for Component Analysis of the Proposed Method

The suggested method is analyzed to have a better comprehension by simply dividing it into two separate algorithms. The performances of the separated algorithms are compared. Also, different combinations and variations of these algorithms have been described and analyzed. These experiments provide insight into the strengths and weaknesses of the method. Algorithms are pairwise compared for different frequency and severity combinations. The results are indicated in Table 4.16. The determined default standard deviation is 10 and the maximum number of mutation step is 70. These are the same for all compared methods that are using the mechanisms.

Firstly, the Cls method is compared with the Snt method. Cls method uses class information while Snt method uses best sentinel information. The Snt method clearly has better performance than the Cls method, as seen from the results. However, when the Cls method combines with the Snt, it outperforms both the Cls and the Snt method alone. RandHM, which is the modified version of Cls, uses hyper-mutation mechanism without knowing the class of a change in an environment. It chooses random standard deviations, unlike the Cls method that selects standard deviations according to a change class. The comparison between Cls and RandHM shows us the classification

Table 4.16 : The ANOVA test results of the component analysis of the proposed method for different frequency and severity combinations.

Algorithm	LF			MF			HF		
	LS	MS	HS	LS	MS	HS	LS	MS	HS
Cls+Snt vs Snt	S+	S+	>	S+	S+	<	S+	>	<
Cls+Snt vs Cls	S+	S+	S+	S+	S+	S+	S+	S+	S+
Cls+Snt vs RandHM	S+	S+	S+	S+	S+	S+	S+	S+	S+
Cls+Snt vs RandHM+Snt	S+	>	S-	S+	>	S-	S+	<	>
Snt vs Cls	S+	S+	S+	S+	S+	S+	S+	S+	S+
Snt vs RandHM	S+	S+	S+	S+	S+	S+	S+	S+	S+
Snt vs RandHM+Snt	S-	<	S-	S-	S-	S-	<	<	>
Cls vs RandHM	S+	S+	S-	S+	S+	>	S+	S+	S+
Cls vs RandHM+Snt	S-	S-	S-	S-	S-	S-	S-	S-	S-
RandHM vs RandHM+Snt	S-	S-	S-	S-	S-	S-	S-	S-	S-

approach is useful for dynamic environments. Cls gives better performance than RandHM. On the other hand, the results show that the RandHM+Snt performs better than the Snt method yet, the Cls+Snt still has better results. The RandHM has the weakest performance among all algorithms. The Cls+Snt algorithm is significantly better than the RandHM+Snt for low severity settings of all frequency levels. Also, Table 4.17 shows that the Cls+Snt performs well according to total scores. Overall, the experimental observations indicate that the Cls+Snt combination is the best choice for the solver approach.

Table 4.17 : The overall ('S+', 'S-', '>' and '<') counts for component analysis of the proposed method.

Algorithm	S+	S-	$\geq$	$\leq$
Cls+Snt	26	2	5	3
Snt	18	10	3	5
Cls	7	28	1	0
RandHM	1	34	0	1
RandHM+Snt	25	3	4	4

4.3 The Experiments for Comparison with Similar Methods in Literature

The proposed method is compared with other single-point search based hyper-heuristic algorithms proposed for dynamic environments since the Cls part of the method exhibits similar behaviors of a hyper-heuristic. One of the previous studies on hyper-heuristics in dynamic environments, Kiraz et al. [44] shows the appropriateness of selection hyper-heuristics as solvers in dynamic optimization problems. The study carried on hyper-heuristics based on single-point search framework with using the MPB. The empirical results of the study demonstrate that selection hyper-heuristics are

capable of reacting and tracking well in different types of changes. In Kiraz's study, the Choice Function (CF) heuristic selection [51] method combined with the Improving and Equal (IE) move acceptance method [51, 52] (CF-IE) has the best performance compared to the other approaches.[1] In this paper, we compared our classification-based method with the CF-IE approach, along with the Hyper-mutation [13] Improving and Equal (HM-IE) and a basic single-point search method (NoHM).

The experimental settings of CF-IE and HM-IE are directly taken from Kiraz's study [44]. The CF-IE uses seven mutation operators that have seven different standard deviations $\{0.5, 2, 7, 15, 20, 25, 30\}$ are used as low-level heuristics as described in Kiraz's study [44].

The HM-IE method operates gaussian mutation using 2 as a default standard deviation. The method changes its standard deviation of 2 to 7 for 70 sequential steps if a change occurs which is also defined in Kiraz's study [44].

The NoHM method runs with using the default standard deviation of 10 in gaussian mutation operation without reacting the changes.

Cls+Snt, CF-IE, HM-IE, and NoHM algorithms are pairwise compared for different frequency and severity combinations. Table 4.18 summarizes the results of the experiments and Figure 4.2 illustrates the box-plots of offline errors for the statistical comparison of the experiments. Also, in Table 4.19, shows the overall counts of each result type of significance tests.

Table 4.18 : The comparison of ANOVA test results of the proposed method with similar approaches in the literature for different frequency and severity combinations.

Algorithm	LF			MF			HF		
	LS	MS	HS	LS	MS	HS	LS	MS	HS
Cls+Snt vs CF-IE	S+	S+	S+	S+	S+	S+	S+	S+	S+
Cls+Snt vs HM-IE	S+	S+	S+	S+	S+	S+	S+	S+	S+
Cls+Snt vs NoHM	S+	S+	S+	S+	S+	S+	S+	S+	S+
CF-IE vs HM-IE	S+	>	S+	S+	>	S-	S-	S-	S-
CF-IE vs NoHM	S+	S+	S+	S+	S+	S+	S+	S+	S-
HM-IE vs NoHM	S+	S+	S+	S+	S+	S+	S+	S+	S+

[1] In this study, the MPB settings for three different severity classes are determined more widely sparsed compared to the Kiraz's study for capturing the behavior of the changing environment. Since the MPB severity settings differ from the Kiraz's study, the experimental results are also different.

Table 4.19 : The overall ('S+', 'S-', '>' and '<') counts for the proposed method and other approaches in the literature.

Algorithm	S+	S-	>	<
Cls+Snt	27	0	0	0
CF-IE	11	14	2	0
HM-IE	13	12	0	2
NoHM	1	26	0	0

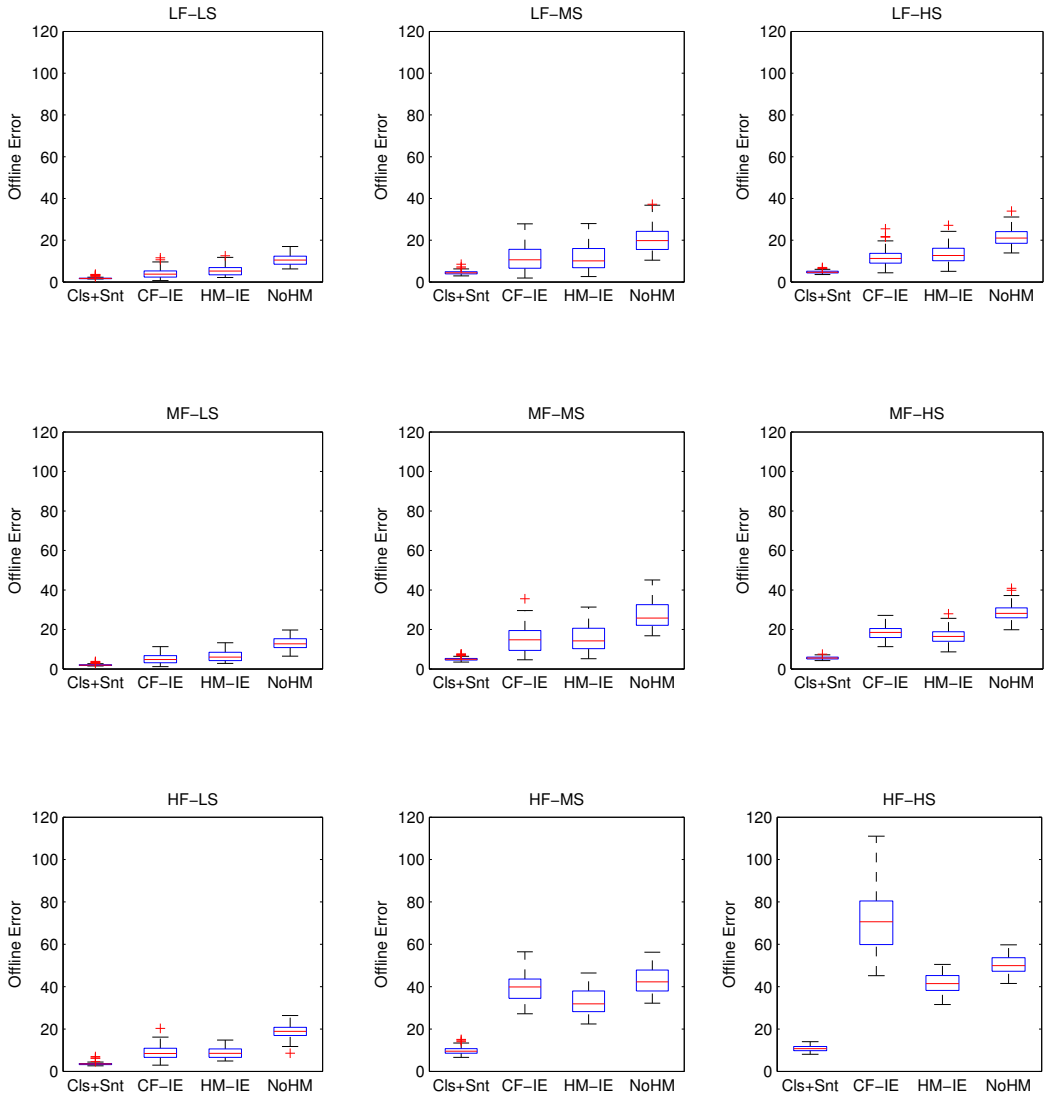

Figure 4.2 : Box-plots of offline errors for compared approaches (Cls+Snt, CF-IE, HM-IE and NoHM) for different frequency and severity combinations.

The experiment shows us the Cls+Snt is significantly better than other compared methods for all frequency-severity pairs (see Table 4.19). The HM-IE is scored closest to the Cls+Snt approach, and it is followed by the CF-IE. The CF-IE gives better than

average outputs, and it is more efficient with lower frequency-severity settings. The NoHM has poorer scores among the group.

4.4 The Random-run Experiments

For further analysis, the change adaptability of the compared approaches is tested with an experiment. The test is carried out by ensuring that each change step, a severity setting is randomly chosen from the determined three classes in Table 4.2 while the frequency is kept fixed. Table 4.20 gives the results of the experiment for each frequency setting in terms of offline error with the standard deviations of the offline errors for 100 runs. Figure 4.3, Figure 4.4 and Figure 4.5 provide the corresponding results as box-plots of offline errors for random-run experiments.

Table 4.20 : Offline errors and standard deviations of the random-run experiments of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) averaged over 100 runs for each frequency settings.

Algorithm	LF	MF	HF
Cls+Snt	3.69 ± 0.77	4.25 ± 0.97	8.12 ± 1.67
CF-IE	10.70 ± 5.20	14.20 ± 5.53	40.01 ± 8.53
HM-IE	11.40 ± 5.32	14.73 ± 5.60	29.25 ± 6.29
NoHM	18.50 ± 4.88	24.92 ± 5.70	39.90 ± 6.20

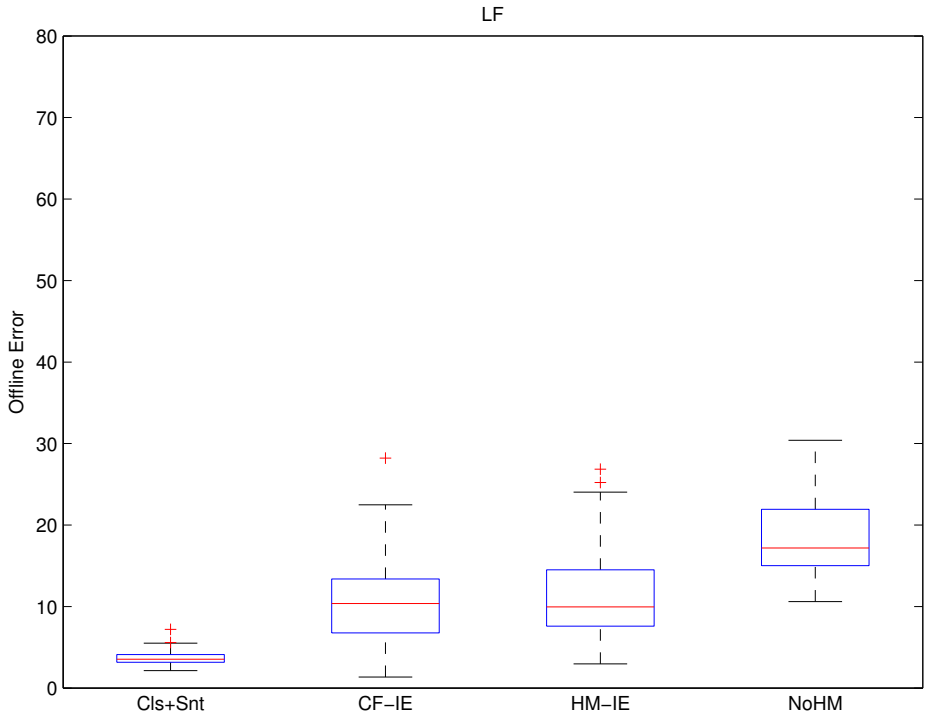

Figure 4.3 : Box-plots of offline errors for random-run experiments of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for low frequency setting.

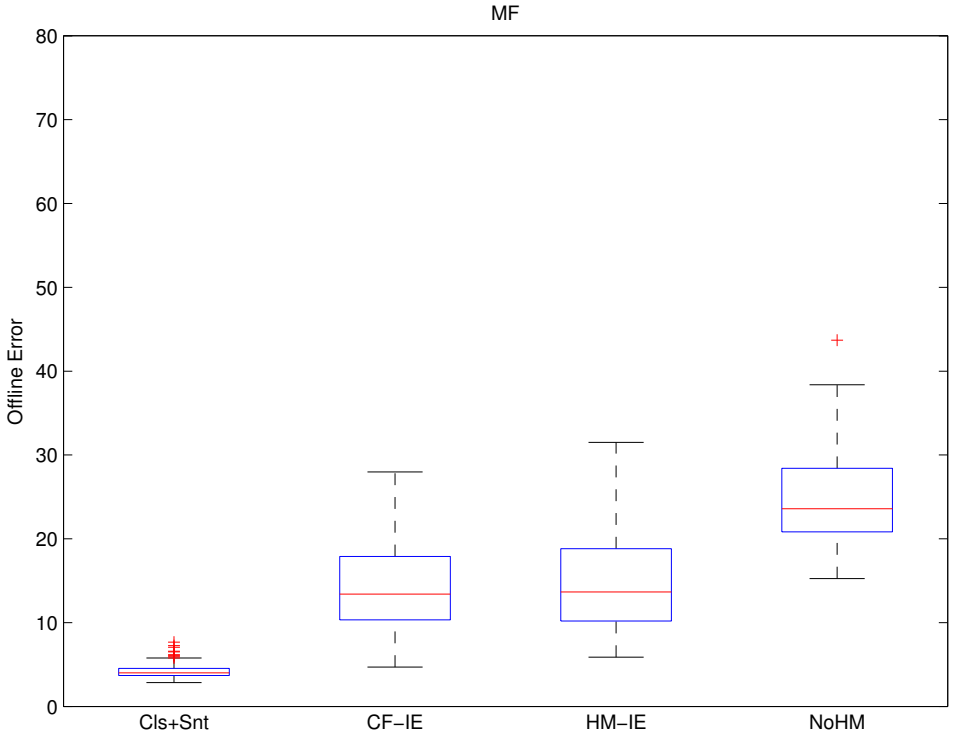

Figure 4.4 : Box-plots of offline errors for random-run experiments of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for medium frequency setting.

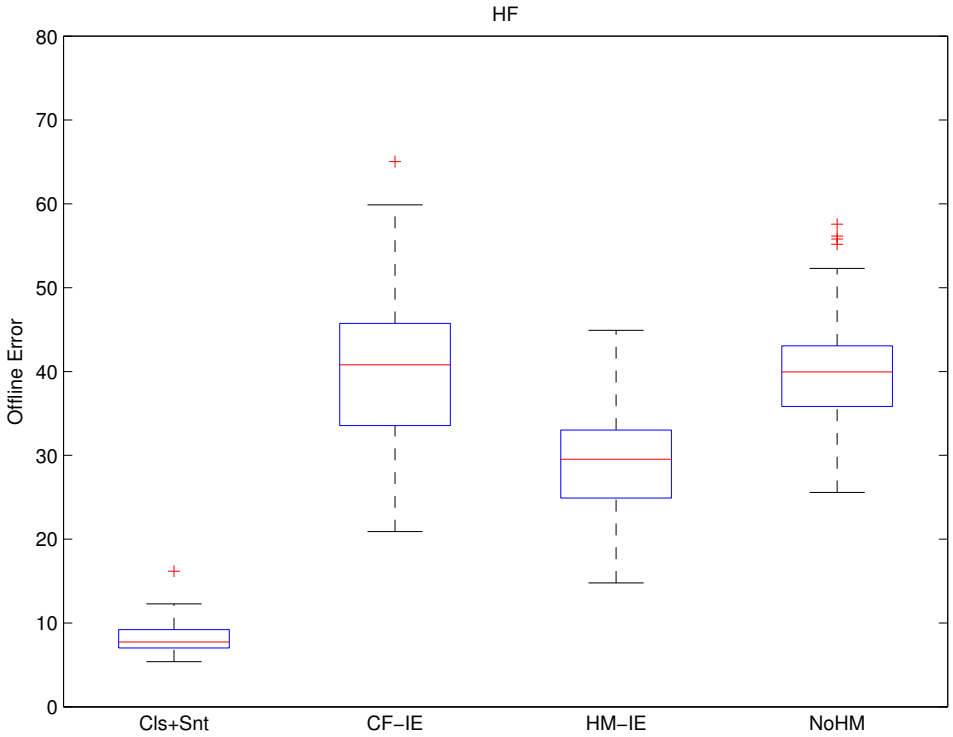

Figure 4.5 : Box-plots of offline errors for random-run experiments of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for high frequency setting.

The offline error of all approaches rises as the frequency increases. Looking at the random-run experiment results, it can be deduced that the Cls+Snt has the best performance. Since the approach reacts to changes in a dynamic environment immediately it is capable of following the random sequential changes. CF-IE gives close results compare to HM-IE for low (LF) and medium (MF) frequency settings. However, when the frequency increase to high (HF) setting, the performance of HM-IE is better than CF-IE. NoHM has poorest scores for all frequency settings.

4.5 The Experiments for Different Settings

In this part of the experiment, we examine the influence of the severity settings that can affect the performance of all compared approaches. We aim to test the limits of our method by using different severity settings. Cls+Snt, CF-IE, HM-IE, and NoHM are also run with the MPB severity settings of Kiraz's study for the random-run experiment (see Table 4.3).

The offline errors of this version for LF, MF and HF are listed in Table 4.21.

Table 4.21 : Offline errors and standard deviations of the random-run experiments of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for each frequency settings using different severity settings.

Algorithm	LF	MF	HF
Cls+Snt	4.00 ± 2.07	4.78 ± 1.91	7.33 ± 2.41
CF-IE	10.36 ± 5.49	11.07 ± 5.96	19.32 ± 6.12
HM-IE	12.87 ± 6.47	15.43 ± 6.57	23.10 ± 7.36
NoHM	19.36 ± 6.75	23.31 ± 6.72	31.44 ± 8.91

In this experimental setup when frequency level is high, the CF-IE, HM-IE and NoHM show improvement in terms of performance compare to previous random-run experiment with initial settings. This means CF-IE, HM-IE and NoHM approaches adapt better in low severity setting. The most significant performance improvement can be observed in CF-IE approach.

On the other hand, Cls+Snt gives similar results compared to the previous random-run experiment. The change in the severity settings does not have much effect on Cls+Snt method. NoHM has the poorest performance for all frequency settings of this experiment.

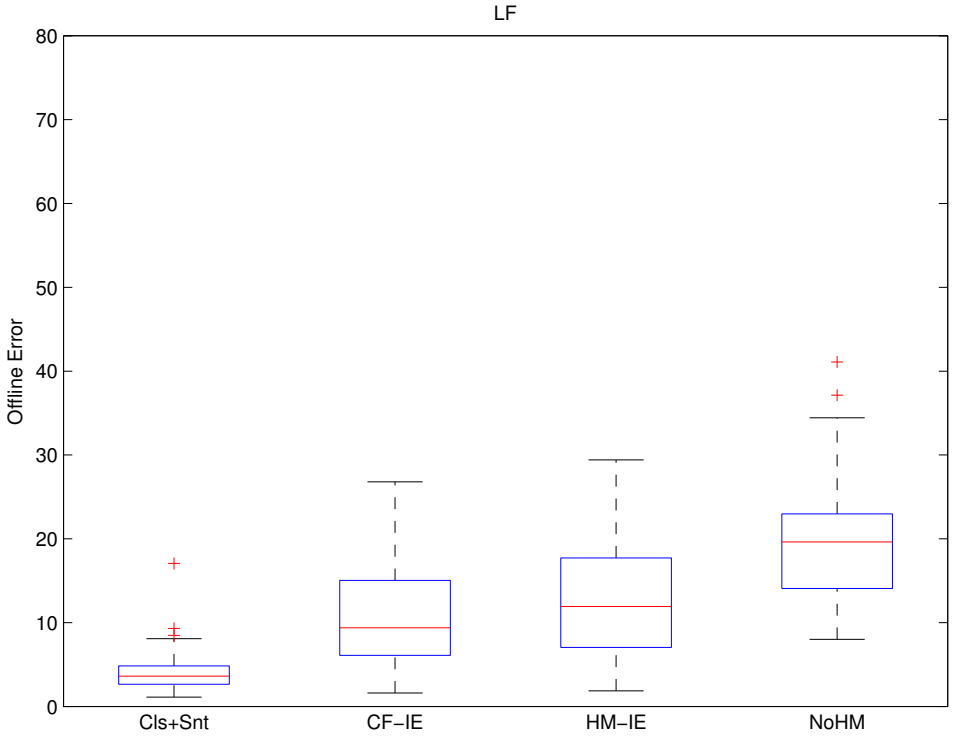

Figure 4.6 : Box-plots of offline errors random-run experiments (using different severity settings) of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for low frequency setting.

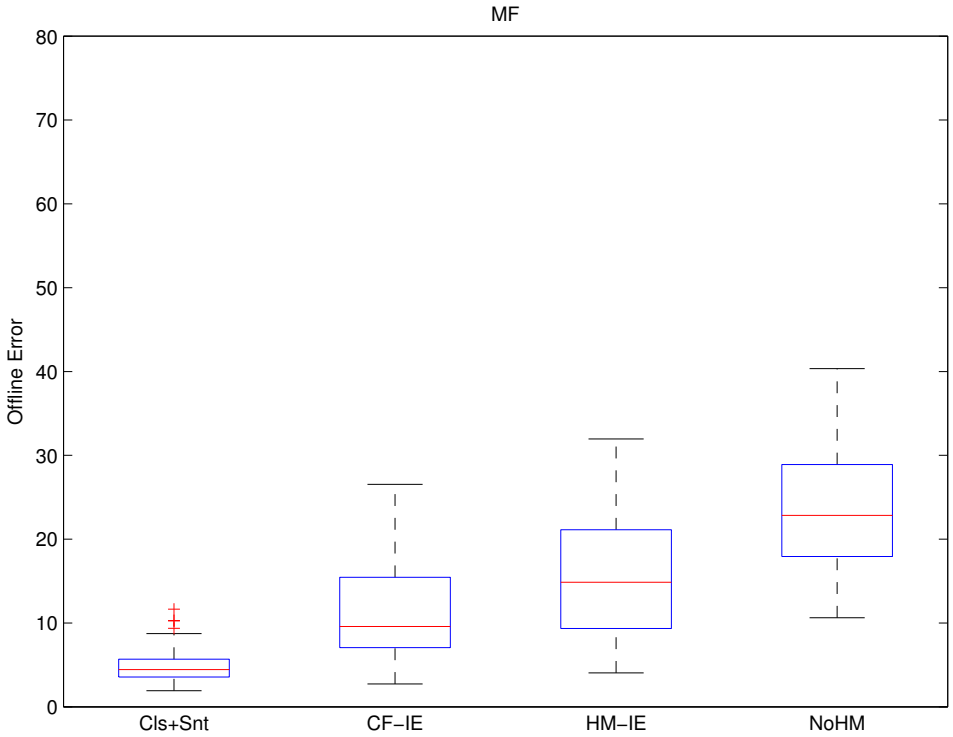

Figure 4.7 : Box-plots of offline errors random-run experiments (using different severity settings) of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for medium frequency setting.

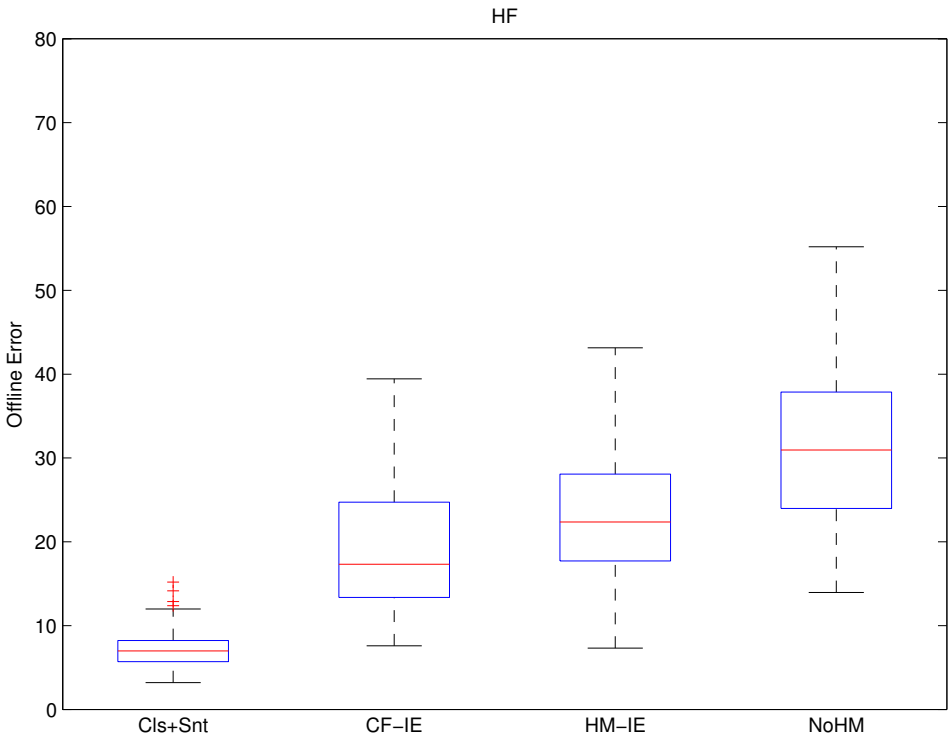

Figure 4.8 : Box-plots of offline errors random-run experiments (using different severity settings) of each compared method (Cls+Snt, CF-IE, HM-IE, and NoHM) for high frequency setting.

Figure 4.6, Figure 4.7 and Figure 4.8 show the corresponding results as box-plots of offline errors for this experiment.

5. CONCLUSION

In this thesis, a classification-based single point search algorithm, which makes use of change characterization information to react differently under different change characteristics in dynamic environments, is introduced. The proposed algorithm reacts to changes like hyper-heuristic approaches previously proposed for dynamic environments. The method uses the representation of change to track the optimum smartly. The novelty of this work lies in making use of the properties of the change for adapting to a changing environment continuously.

The proposed algorithm is tested using the MPB, a test benchmark, that generates dynamic landscapes with a number of peaks; every change in the environment creates differences in heights, widths, and locations of each peak. The benchmark is suitable for this study since it presents similar characteristics like in real-world.

In the first stage of this thesis, we applied several measures to extract features from the dynamic landscape. These features provide information for the classification process of dynamic environments. At the classification process, several classification algorithms are used with the extracted data for building a classifier model. The Random Forest Algorithm is selected since it gives excellent results in terms of accuracy, speed and robustness compared to other classification methods (K-Nearest Neighbor, Gradient Boosting, Multi-layer Perceptron, and Support Vector Machine Classifiers). The influence of the parameters on the accuracy of the classification mechanism is also tested with several experiments and proper parameter settings are adopted.

In the second stage of this thesis, experiments are performed to understand the underlying components of the proposed method. The method is analyzed to have a better comprehension by simply dividing it into separate algorithms. These experiments provide insight into the strengths and weaknesses of the proposed method.

In the third stage of this thesis, we compare the performance of our suggested approach with similar single point search-based hyper-heuristic approaches for dynamic environments in literature. The experimental results are promising and show the strength of the proposed heuristic approach as a dynamic optimization solver. In addition to this experiment, for maintaining dynamism as close as to real life, a test setting with random changes in the dynamic environment is established, and compared methods are tested in this experimental setup. As a result of this experiment, it has been shown that the adaptation ability of our proposed approach is better than other methods. All approaches are compared under the assumption that the occurrence of a landscape change is known and thus change detection step is ignored. If change detection is considered as an issue, the proposed method will have a clear advantage with the sentinels, even the occurrence of changes cannot be easily detected.

In the final stage of this thesis, we run the experiments for different settings to prove the capability of our mechanism. The experimental results indicate the proposed approach is able to work with various types of dynamism. The classification model plays an essential role in our approach. The model used in this work has been trained with the extracted data from environments that have specified settings. The capability of the model is explored for different environment settings with this experiment. A comprehensive analysis can be conducted for building a more generic model. Therefore, the research of a more generic model can be a separate study topic. The accuracy of the model depends on a set of parameters such as sentinel count, number of peaks and the severity settings of each class. These parameters except the sentinel count also have an impact on the difficulty level of a dynamic optimization problem and will be investigated more for the overall performance of the proposed method. All in all, as a future work the parameter dependency of the proposed algorithm can be investigated more extensively.

We are aware that the process of generating input data for classifier costs several evaluations directly proportional to the sentinel count. However, the method turns that into an advantage by using the best sentinel information as explained in the Snt method. The trade-off between the sentinel count and the accuracy of the classifier can be examined further to be optimized. Improvements in this aspect will decrease computational complexity and save time to the approach.

Furthermore, introducing different types of dynamism, like dimensionality change to our optimization problem could be interesting as another future work.

Finally, the main idea explored in this study can be applied to population-based search algorithms and thus can be compared with other state-of-art population-based heuristic methods for dynamic environments.

REFERENCES

[1] **Cruz, C.**, **González, J.R. and Pelta, D.A.** (2011). Optimization in dynamic environments: a survey on problems, methods and measures, *Soft Computing*, *15*(7), 1427–1448.

[2] **Jin, Y. and Branke, J.** (2005). Evolutionary optimization in uncertain environments-a survey, *IEEE Transactions on evolutionary computation*, *9*(3), 303–317.

[3] **De Jong, K.** (1999). Evolving in a changing world, *International Symposium on Methodologies for Intelligent Systems*, Springer, pp.512–519.

[4] **Branke, J.** (1999). *Evolutionary algorithms for dynamic optimization problems: A survey*, AIFB.

[5] **Branke, J.** (2001). Evolutionary Optimization in Dynamic Environments.

[6] **Branke, J.**, **Salihoğlu, E. and Uyar, Ş.** (2005). Towards an analysis of dynamic environments, *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, ACM, pp.1433–1440.

[7] **Weicker, K.** (2003). *Evolutionary algorithms and dynamic optimization problems*, Der Andere Verlag Berlin.

[8] **Morrison, R.W.** (2004). *Designing Evolutionary Algorithms for Dynamic Environments*, Springer Science & Business Media.

[9] **Yang, S.**, **Ong, Y.S. and Jin, Y.** (2007). *Evolutionary computation in dynamic and uncertain environments*, volume 51, Springer Science & Business Media.

[10] **Grefenstette, J.J.** *et al.* (1992). Genetic algorithms for changing environments, *PPSN*, volume 2, pp.137–144.

[11] **Tinós, R. and Yang, S.** (2007). A self-organizing random immigrants genetic algorithm for dynamic optimization problems, *Genetic Programming and Evolvable Machines*, *8*(3), 255–286.

[12] **Yang, S.** (2008). Genetic algorithms with memory-and elitism-based immigrants in dynamic environments, *Evolutionary Computation*, *16*(3), 385–416.

[13] **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**, NAVAL RESEARCH LAB WASHINGTON DC.

[14] **Abbass, H.**, **Sastry, K. and Goldberg, D.** (2004). Oiling the wheels of change: The role of adaptive automatic problem decomposition in non-stationary environments (IlliGAL Report No. 2004029), *Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory*.

[15] **Tinós, R. and Yang, S.** (2008). Evolutionary programming with q-Gaussian mutation for dynamic optimization problems, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, IEEE, pp.1823–1830.

[16] **Branke, J.** (1999). Memory enhanced evolutionary algorithms for changing optimization problems, *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, IEEE, pp.1875–1882.

[17] **Karaman, A.**, **Uyar, Ş. and Eryiğit, G.** (2005). The memory indexing evolutionary algorithm for dynamic environments, *Workshops on Applications of Evolutionary Computation*, Springer, pp.563–573.

[18] **Yang, S.** (2006). Associative memory scheme for genetic algorithms in dynamic environments, *Workshops on Applications of Evolutionary Computation*, Springer, pp.788–799.

[19] **Richter, H. and Yang, S.** (2009). Learning behavior in abstract memory schemes for dynamic optimization problems, *Soft Computing*, *13*(12), 1163–1173.

[20] **Pelta, D.**, **Cruz, C. and Verdegay, J.L.** (2009). Simple control rules in a cooperative system for dynamic optimisation problems, *International Journal of General Systems*, *38*(7), 701–717.

[21] **Yang, S.**, **Cheng, H. and Wang, F.** (2010). Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(1), 52–63.

[22] **Branke, J.**, **Kaußler, T.**, **Smidt, C. and Schmeck, H.**, (2000). A multi-population approach to dynamic optimization problems, Evolutionary Design and Manufacture, Springer, pp.299–307.

[23] **Blackwell, T. and Branke, J.** (2004). Multi-swarm optimization in dynamic environments, *Workshops on Applications of Evolutionary Computation*, Springer, pp.489–500.

[24] **Mendes, R. and Mohais, A.S.** (2005). DynDE: a differential evolution for dynamic optimization problems, *2005 IEEE Congress on Evolutionary Computation*, volume 3, IEEE, pp.2808–2815.

[25] **Blackwell, T. and Branke, J.** (2006). Multiswarms, exclusion, and anti-convergence in dynamic environments, *IEEE transactions on evolutionary computation*, *10*(4), 459–472.

[26] **Moser, I. and Hendtlass, T.** (2007). A simple and efficient multi-component algorithm for solving dynamic function optimisation problems, *2007 IEEE Congress on Evolutionary Computation*, IEEE, pp.252–259.

[27] **Novoa, P.**, **Pelta, D.A.**, **Cruz, C. and del Amo, I.G.** (2009). Controlling particle trajectories in a multi-swarm approach for dynamic optimization problems, *International Work-Conference on the Interplay Between Natural and Artificial Computation*, Springer, pp.285–294.

[28] **Morrison, R.W. and De Jong, K.A.** (1999). A test problem generator for non-stationary environments, *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, IEEE, pp.2047–2053.

[29] **Aydin, M.E. and Öztemel, E.** (2000). Dynamic job-shop scheduling using reinforcement learning agents, *Robotics and Autonomous Systems*, *33*(2-3), 169–178.

[30] **Younes, A.**, **Calamai, P. and Basir, O.** (2005). Generalized benchmark generation for dynamic combinatorial problems, *Proceedings of the 7th annual workshop on Genetic and evolutionary computation*, ACM, pp.25–31.

[31] **Hanshar, F.T. and Ombuki-Berman, B.M.** (2007). Dynamic vehicle routing using genetic algorithms, *Applied Intelligence*, *27*(1), 89–99.

[32] **Lin, S.C.**, **Goodman, E.D. and Punch III, W.F.** (1997). A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problem., *ICGA*, pp.481–488.

[33] **Branke, J. and Mattfeld, D.C.** (2000). Anticipation in dynamic optimization: The scheduling case, *International Conference on Parallel Problem Solving from Nature*, Springer, pp.253–262.

[34] **Mack, Y.**, **Goel, T.**, **Shyy, W. and Haftka, R.**, (2007). Surrogate model-based optimization framework: a case study in aerospace design, Evolutionary computation in dynamic and uncertain environments, Springer, pp.323–342.

[35] **Michalewicz, Z.**, **Schmidt, M.**, **Michalewicz, M. and Chiriac, C.**, (2007). Adaptive business intelligence: three case studies, Evolutionary computation in dynamic and uncertain environments, Springer, pp.179–196.

[36] **De Jong, K.A.** (1975). Analysis of the behavior of a class of genetic adaptive systems.

[37] **Morrison, R.W.** (2003). Performance measurement in dynamic environments, *GECCO workshop on evolutionary algorithms for dynamic optimization problems*, 5-8, Citeseer.

[38] **Yang, S.**, (2007). Explicit memory schemes for evolutionary algorithms in dynamic environments, Evolutionary computation in dynamic and uncertain environments, Springer, pp.3–28.

[39] **Yang, S. and Yao, X.** (2008). Population-based incremental learning with associative memory for dynamic environments, *IEEE Transactions on Evolutionary Computation*, *12*(5), 542–561.

[40] **Mühlenbein, H. and Paass, G.** (1996). From recombination of genes to the estimation of distributions I. Binary parameters, *International conference on parallel problem solving from nature*, Springer, pp.178–187.

[41] **Peng, X.**, **Gao, X. and Yang, S.** (2011). Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments, *Soft Computing*, *15*(2), 311–326.

[42] **Duhain, J.G. and Engelbrecht, A.P.** (2012). Towards a more complete classification system for dynamically changing environments, *2012 IEEE Congress on Evolutionary Computation*, IEEE, pp.1–8.

[43] **Burke, E.K.**, **Gendreau, M.**, **Hyde, M.**, **Kendall, G.**, **Ochoa, G.**, **Özcan, E. and Qu, R.** (2013). Hyper-heuristics: A survey of the state of the art, *Journal of the Operational Research Society*, *64*(12), 1695–1724.

[44] **Kiraz, B.**, **Etaner-Uyar, A.Ş. and Özcan, E.** (2013). Selection hyper-heuristics in dynamic environments, *Journal of the Operational Research Society*, *64*(12), 1753–1769.

[45] **Morrison, R.W.**, (2004). A New EA for Dynamic Problems, Designing Evolutionary Algorithms for Dynamic Environments, Springer, pp.53–68.

[46] **Nguyen, T.T.**, **Yang, S. and Branke, J.** (2012). Evolutionary dynamic optimization: A survey of the state of the art, *Swarm and Evolutionary Computation*, *6*, 1–24.

[47] **Breiman, L.** (2001). Random forests, *Machine learning*, *45*(1), 5–32.

[48] **Pedregosa, F.**, **Varoquaux, G.**, **Gramfort, A.**, **Michel, V.**, **Thirion, B.**, **Grisel, O.**, **Blondel, M.**, **Prettenhofer, P.**, **Weiss, R.**, **Dubourg, V.** *et al.* (2011). Scikit-learn: Machine learning in Python, *Journal of machine learning research*, *12*(Oct), 2825–2830.

[49] **Buitinck, L.**, **Louppe, G.**, **Blondel, M.**, **Pedregosa, F.**, **Mueller, A.**, **Grisel, O.**, **Niculae, V.**, **Prettenhofer, P.**, **Gramfort, A.**, **Grobler, J.** *et al.* (2013). API design for machine learning software: experiences from the scikit-learn project, *arXiv preprint arXiv:1309.0238*.

[50] **Variable Importance - H2O 3.24.0.2 documentation**, Retrieved January 18, 2019 from `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/variable-importance.html#variable-importance`.

[51] **Cowling, P.**, **Kendall, G. and Soubeiga, E.** (2000). A hyperheuristic approach to scheduling a sales summit, *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp.176–190.

[52] **Cowling, P.**, **Kendall, G. and Soubeiga, E.** (2002). Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation, *Workshops on Applications of Evolutionary Computation*, Springer, pp.1–10.

CURRICULUM VITAE

Name Surname: Şeyda Yıldırım Bilgiç

Place and Date of Birth: Erzurum, 18.04.1990

E-Mail: sydyildirim@gmail.com

EDUCATION:

- **B.Sc.:** 2014, Yildiz Technical University, Faculty of Electrical and Electronic, Computer Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2015-..., TUBITAK Bilgem BTE, Researcher

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Yıldırım Bilgiç Ş., Etaner-Uyar A.Ş., 2019: A Classification-based Heuristic Approach for Dynamic Environments, MENDEL 2019: International Conference on Soft Computing-MENDEL, July 10-12, 2019 Brno, Czech Republic.