Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost

Danial Yazdani

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

August 2018

Declaration

The work presented in this thesis was carried out at the Liverpool Logistics, Offshore and Marine Research Institute, Liverpool John Moores University. Unless otherwise stated, it is the original work of the author.
While registered as a candidate for the degree of Doctor of Philosophy, for which submission is now made, the author has not been registered as a candidate for any other award. This thesis has not been submitted in whole, or in part, for any other degree.

Danial Yazdani
Liverpool Logistics, Offshore and Marine Research Institute
Faculty of Engineering and Technology
Liverpool John Moores University
Byrom Street Campus
Liverpool
L3 3AF
UK

October 2018

Abstract

Change is an inescapable aspect of natural and artificial systems, and adaptation is central to their resilience. Optimization problems are no exception to this maxim. Indeed, viability of businesses depends heavily on their effectiveness in responding to a change in the myriad of optimization problems they entail. Changes in optimization problems usually are result of change in the objective function and/or number of variables and/or constraints. Such optimization problems are denoted as dynamic optimization problems (DOPs) in the literature. Despite the large body of literature on DOPs and algorithms in this domain, there are still noticeable gaps between real-world DOPs and academic research. The first objective of this thesis is investigating DOPs to identify any class of DOPs or any DOPs' characteristics that are common in practical situation but have not been studied by the researchers.

In this thesis, two important gaps are identified, namely considering switching cost in DOPs and large-scale DOPs. Both are common in many real-world dynamic problem but a few research investigated them in the past. In an attempt to bridge these gaps, this thesis makes the following contributions:

First, this thesis considers the impact of cost for changing solutions after environmental changes. In fact, changing solutions in real-world problems is costly. Furthermore, larger changes have higher cost and need more resources such as time, human resources and energy. Thus, lack of switching cost consideration in most previous algorithms makes them unsuitable for many of real-world DOPs. In this thesis, different scenarios of DOPs with switching cost are investigated, their challenges are identified, and the performance of the state-of-the-art methods are investigated for solving them. Contributions include developing a novel robust optimization over time (ROOT) framework, a novel adaptive method for maximizing efficiency by changing or keeping solutions after environmental changes, and a novel multi-objective and time-linkage based method for minimizing switching cost.

Second, this thesis investigates large-scale DOPs. Up to now, little attention has been given to the scalability of DOPs. Indeed, the dimension of typical DOPs studied in the literature hardly exceeds twenty. In this thesis, the challenges of large-scale DOPs are studied, then the efficiency of the current methods are investigated for solving them. Moreover, this thesis proposes a novel cooperative coevolution algorithm based on a multi-population approach which benefits from a new resource allocation method for DOPs with high-dimensional search space.

All the proposed methods in this thesis use particle swarm optimization as the core optimizer embedded in a multi-population framework. The performance of the proposed methods are compared with state-of-the-art methods on a wide range of problem instances generated by the state-of-the-art and the proposed DOP benchmarks. The comparison results indicate the superiority of the proposed methods.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my Supervisors Dr. Trung Thanh Nguyen, Prof. Juergen Branke and Prof. Jin Wang for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Besides my supervisors, I would like to thank Dr. Mohammad Nabi Omidvar for his insightful comments and encouragement. Last but not the least, I would like to thank my family: my parents and to my wife for supporting me spiritually throughout writing this thesis.

This research was supported in part by a Dean's Scholarship by Faculty of Engineering and Technology, LJMU, a Newton Institutional Links grant no. 172734213, funded by the UK BEIS and delivered by the British Council, and a NRCP grant no. NRCP1617-6-125 delivered by Royal Academy of Engineering.

Danial Yazdani   October 2018

Declaration of Authorship

“If you can dream it, you can do it.”

Walt Disney

Contents

List of Figures

List of Tables

Abbreviations

ASC	**A**daptive **S**olution **C**hooser
CBCC	**C**ontribution-**B**ased **C**ooperative **C**oevolution
CC	**C**ooperative **C**oevolutionary
CCMPSO	**C**ooperative **C**oevolutionary **M**ulti **P**article **S**warm **O**ptimization
CMPB	**C**omposite **M**oving **P**eaks **B**enchmark
DG	**D**ifferential **G**rouping
DG2	Second version of **D**ifferential **G**rouping
DMOOP	**D**ynamic **M**ulti-**O**bjective **O**ptimization **P**roblems
DOP	**D**ynamic **O**ptimization **P**roblems
DTP	**D**ynamic **T**ime-linkage **P**roblem
EA	**E**volutionary **A**lgorithm
EDO	**E**volutionary **D**ynamic **O**ptimization
FTmPSO	**F**inder-**T**racker **m**ulti-swarm **P**article **S**warm **O**ptimization
GDG	**G**lobal **D**ifferential **G**rouping
mMPBR	**m**odified **M**oving **P**eaks **B**enchmark for **R**OOT
MMsPSO	**M**ulti **M**ulti **s**warm **P**article **S**warm **O**ptimization
MOOP	**M**ulti-**O**bjective **O**ptimization **P**roblems
MPB	**M**oving **P**eaks **B**enchmark
mPSO	**M**ulti-swarm **P**article **S**warm **O**ptimization
MSsPSO	**M**ulti **S**ingle **s**warm **P**article **S**warm **O**ptimization
NRS	**N**ext **R**obust **S**olution
POF	**P**areto-**O**ptimal **f**ront
POS	**P**areto-**O**ptimal **S**et
PSDR	**P**revious-**S**olution **D**isplacement **R**estriction
PSO	**P**article **S**warm **O**ptimization

ROOT	**R**obust **O**ptimization **O**ver **T**ime
SC	**S**witching **C**ost
SI	**S**warm **I**ntelligence
SMsPSO	**S**ingle **M**ulti **s**warm **P**article **S**warm **O**ptimization
sPSO	**S**ingle-swarm **P**article **S**warm **O**ptimization
SRA	**S**cheduling and **R**esource **A**llocation
TMO	**T**racking **M**oving **O**ptima
XDG	E**x**tended **D**ifferential **G**rouping

To my family . . .

Chapter 1

Introduction

Optimization is the heart of many processes that take place in nature. Moreover, it plays a major role in business and engineering domains. In order to solve optimization problems, at least an objective function is normally designed based on the problem parameters first. Then, the goal is to find the input parameters that optimize objective function(s) under necessary constraints by use of mathematical or intelligent methods. When optimization problems are too complex, using mathematical methods becomes extremely difficult or even impossible. For such problems, intelligent methods adopted from nature can be used.

Optimization problems can be categorized according to different criteria. One criterion is whether the optimization problem is static or dynamic. In problems with static environments, the problem remains unchanged in the course of time, whereas in dynamic problems, the problem changes over time. Since many real-world problems have parameters that are time-variant, it can be concluded that dynamic optimization problems (DOPs) is of paramount importance (Nguyen, 2011).

In view of the importance of DOPs, numerous researchers and scientists have attempted to design algorithms for solving these problems (Yang and Yao, 2013). In the recent decade, using evolutionary algorithms and swarm intelligence methods for optimization in dynamic environments has become the focus of attention of many researchers in the field of computational intelligence (Nguyen et al., 2012a; Mavrovouniotis et al., 2017). The reason behind this attention may be found in the nature of these methods, their path of evolution, and their adaptability when facing changes in the environment. However, designing optimization algorithms in dynamic environments faces many challenges that require new algorithms. As a matter of fact, not only that the algorithm should be able to find optimum position(s) in dynamic environments, it should also be able to track them. Resolving existing challenges in DOPs, improvement of results, reduction of computational costs, and especially closing gaps between academic research and real-world problems in this field are the most important goals in designing optimization algorithms for dynamic environments.

Despite the large body of literature on DOPs and algorithms, this field still has a lot of open areas with open research questions, of which perhaps one of the most important questions is about how well academic research in this area reflects the common characteristics of real-world DOPs and if there are any types of DOPs that have not been covered by current academic research. The main purpose of this thesis is to investigate this important question and to propose solutions to close some of the gaps in this issue.

1.1 Scope of the thesis

DOPs' range is very large and they can be categorized from different aspects. Investigated DOPs in this thesis have the following characteristics

- The search space is continuous.
- The change happens in the objective function. Therefore, the shape of search space change over time.
- The landscape is multi-modal i.e. there are several peaks whose height, width and location change over time and the optimizer has to cover several peaks in order to increase the performance.
- The change is unpredictable i.e. the generated changes do not follow any regular pattern such as circular relocations and moving on a line.
- They are unconstrained and any solution inside the boundaries is a feasible solution.
- The boundaries or variable domains do not change over time.
- Variable interactions remain unchanged over time.
- Number of variables is constant over time.
- Number of objective is stationary over time.
- Environmental changes are detectable.
- The environmental changes happen discrete in time.

Investigated DOPs in this thesis are time-dependent problems in which the optimizer has to cope with environmental changes (Branke, 2002). This type of DOPs can be defined as:

$$F(\mathbf{x}) = f(\mathbf{x}, \theta^{(t)}), \tag{1.1}$$

where f is the objective function, $\mathbf{x}$ is a design vector, $\theta^{(t)}$ is environmental parameters which change over time and t is the time index with $t \in [0, T]$ where T is the problem life cycle or number of environments. $\theta^{(t)}$ in the investigated DOPs in this thesis changes discretely. In this type of DOPs, the environmental parameters change over time with stationary periods between changes. As a result, for a DOP with $T-1$ environmental changes, there is a sequence of T static environments that can be described as:

$$F(\mathbf{x}) = \left[f(\mathbf{x}, \theta^{(1)}), f(\mathbf{x}, \theta^{(2)}), \ldots, f(\mathbf{x}, \theta^{(T)}) \right], \tag{1.2}$$

where $\theta^{(i)}$ represents the environmental parameters in the i^{th} environment.

1.2 Research questions

The approach of this thesis is to start from a general question to get an overview of the important gaps in the field of DOPs:

- *Is there any type of DOP, or any type of problem characteristics that are common in real-world dynamic optimization problems but have not been studied in the DOP field by researchers?*

Answering the question above requires a literature review on DOPs. After determining current gaps between academic research and practical situations in DOPs and find problems with specific characteristics that have been rarely investigated in the past, there will be more specific questions such as:

- *How these problems and characteristics can be captured in academic benchmark problems?*
- *How new benchmarks can be designed to generate test instances whose characteristics are similar to those specific real-world problems?*
- *What would be the performance of existing methods on these problems?*
- *How the performance of existing methods can be evaluated?*

- *What can be done to improve the performance?*
- *How these problems can be effectively solved, which have not been solved before?*

These specific questions show us the research directions to be done in the rest of the thesis.

1.3 Contribution of the thesis

Contributions in this thesis can be classified into two main groups:

1.3.1 Considering displacement between successive solutions

Most previous research on DOPs focuses on tracking moving optima (TMO) (Nguyen et al., 2012a) without considering any limitation or cost for changing solutions. In fact, changing solutions in real-world problems is costly. Furthermore, larger changes have more cost and need more resources such as time, human resources and energy. Thus, lack of switching cost (SC) consideration in TMO algorithms makes them unsuitable for many real-world problems. Moreover, in many real-world DOPs, changing solutions frequently is not desirable or it may be very costly. For example, in scheduling, changing the schedule may have significant impact on suppliers and customers, or, in the design of telephone networks, sending out engineers to change the physical infrastructure can be very expensive. In the taking-off/landing scheduling problem, it is desirable to keep the current implemented solution/schedule after an environmental change (E.Wilkins et al., 2008; Atkin et al., 2008) to avoid unfavorable disruptions in airport operations.

This thesis investigates DOPs with SC under three different conditions:

1.3.1.1 DOPs with previous-solution displacement restriction

In DOPs with previous-solution displacement restriction (PSDR) (Nguyen, 2011), the average displacement between successive solutions is an objective which needs to be minimized. In DOPs with PSDR, the algorithm needs to find a new solution after an environmental change that is not much different from the previous one (Nguyen, 2011). Moreover, displacement between consecutive solutions can be seen as the SC in many problems (Huang et al., 2017) which needs to be minimized as the second objective. In the field of DOPs, little attention has been given to this type of problems. Investigating DOPs with PSDR, studying the performance of the current algorithms on them, and designing a new method for solving them are among the goals of this thesis.

1.3.1.2 DOPs with very large switching cost

When SC is very large, or frequently changing solutions is undesirable, the algorithm needs to keep solutions as long as they remain acceptable. Additionally, these circumstances happens when there are limitations in resources for changing solutions. For addressing this type of DOPs, an approach called robust optimization over time (ROOT) (Yu et al., 2010) was proposed in which algorithms search for solutions that are not necessarily the best but they are acceptable and can remain acceptable after environmental changes. Therefore, in ROOT, algorithms try to find solutions which are robust to environmental changes and the main objective is maximizing survival time of robust solutions over time (Fu et al., 2013). Several state-of-the-art ROOT methods have been proposed until now, of which the one based on survival time metric (Fu et al., 2013) is the most successful. However, this method and all other methods such as (Huang et al., 2017; Guo et al., 2014), which were designed based on survival time metric, are not capable of finding robust solutions in problems with higher dimensions. According to (Yazdani et al., 2018a; Huang et al., 2017), these methods completely lose their efficiency on 10 dimensional problems, which makes them unsuitable for many real-world problems. As a result, proposing a new framework for addressing the above mentioned challenges of the current methods is among the goals of this thesis.

This thesis proposes a new framework for ROOT. Its novelty and contribution are as follows. First, this framework uses multi-population (Mavrovouniotis et al., 2017; Nguyen et al., 2012a) methods to track and monitor peaks and learn about their characteristics. Second, in contrast to previous state-of-the-art frameworks which are based on predicting future fitness values of solutions, the proposed framework tries to predict future behaviors of peaks and then selects the next robust solution based on this information. Third, this thesis proposes four new strategies to select the next robust solution. Experiments show that the proposed method outperforms previous methods significantly and can perform very well in problems with higher dimensions.

1.3.1.3 DOPs with varying switching cost

In the presence of SC in DOPs, one important technical question is "When should a solution be changed?". TMO and ROOT address two extreme cases. TMO is suitable for circumstances in which there is no SC or it is very low. On the other hand, ROOT is suitable for situations in which the SC is very high so the algorithm tries to keep each solution as long as it remains acceptable after environmental changes.

In this thesis, a new adaptive solution chooser (ASC) algorithm is proposed which acts in a similar way to TMO algorithms where SC is low and acts like ROOT algorithms when the SC is high. However, the main contribution of ASC is where the algorithm can decide about changing or keeping solutions based on their current fitness values, the fitness of other found solutions with better quality and their SC from the current solution. Indeed, although changing solutions in real-world problems is costly, there are situations in which the algorithm has found a solution whose quality is so high that the benefit of switching largely outweighs the cost.

1.3.2 Scaling up DOPs

Despite the large body of literature on DOPs and algorithms, little attention has been given to their scalability. Indeed, the dimension of a typical DOP studied in the literature hardly exceeds twenty. Motivated by rapid technological advancements, large-scale optimization has gained popularity in recent years. However, the exponential growth in the size of the search space, with respect to an increase in the number of the decision variables, has made large-scale optimization a challenging task. For DOPs, however, the challenge is twofold. For such problems, not only should an algorithm be capable of finding the global optimum in the vastness of the search space but should also be able to track it over time. In this thesis, advances in large-scale global optimization are investigated and a novel decomposition-based algorithm is proposed for large-scale dynamic optimization problems.

The Moving Peaks Benchmark (MPB) (Branke, 1999) is the most popular benchmark in the field of DOPs. In this part of this thesis, first, the standard MPB is formally analyzed and it is shown that its lack of modularity limits its applicability for the study of large-scale DOPs. Then a new benchmark is proposed by composing several MPBs. The proposed benchmark is suitable for generating problem instances in which the components are heterogeneous in terms of dimension and their contribution to the fitness function value. More specifically, this part of this thesis has the following major contributions:

1. A mathematical variable interaction analysis on the MPB benchmark to determine its interaction structure.
2. A large-scale benchmark suite with a modular heterogeneous structure allowing for imbalance among its components.
3. A decomposition-based algorithm for solving large-scale DOPs with a novel resource allocation mechanism.

1.4 Outline of the thesis

This Thesis is organized as follows:

Chapter 2 reviews existing research related to the proposed approaches in this thesis.

In Chapter 3, a novel algorithm is proposed for DOPs with PSDR. This algorithm utilizes a multi-swarm particle swarm optimization (PSO) that is responsible for finding and tracking peaks based on the optimality objective and a single-swarm PSO (sPSO) whose task is to find the optimum solution according to both optimality and displacement/SC objectives. After each environmental change, a new decision maker chooses a peak according to the fitness values of peaks in the present and some of their characteristics which can be used to anticipate the future displacement/SC value. Then, sPSO uses the location information from the decision maker in order to accelerate the optimization process and improve the performance.

Chapter 4 proposes a new framework for ROOT. In the proposed framework, a multi-population method is utilized to track and monitor peaks and learn about their behavior. Then, if the current solution is needed to be changed due to its unacceptable fitness value, a new decision maker chooses the next solution based on the learned behavior of peaks. Four different strategies are proposed which choose solutions based on different characteristics (Yazdani et al., 2018a).

In Chapter 5, a new adaptive solution chooser (ASC) algorithm is proposed which acts in a similar way to TMO algorithms where SC is low and acts as ROOT algorithms when the SC is high. ASC is able to decide about changing or keeping solutions based on their current fitness values, the fitness of other found solutions with better quality and their SC from the current solution. The main purpose of designing ASC is to minimize the the cost by maximizing average profit minus switching cost.

In Chapter 6, large-scale DOPs are investigated and a new benchmark and decomposition-based algorithm for this type of DOPs are proposed. The idea is to first discover and exploit the underlying structure of a given problem by decomposing it into several components of smaller size, and then to tackle the sub-problems simultaneously. The former can be achieved by a wide range of variable interaction analysis algorithms capable of identifying the underlying structure of a black-box problem with high efficiency and accuracy (Omidvar et al., 2014a, 2017; Mei et al., 2016; Sun et al., 2017), and the latter can be achieved by means of cooperative coevolution (Potter and Jong, 2000; Yang et al., 2008a; Li and Yao, 2012).

In this chapter, a new benchmark based on MPB is proposed for large-scale DOPs. In addition to the new benchmark, a new algorithm for large-scale DOPs is proposed.

The proposed algorithm utilizes the state-of-the-art DG2 (Omidvar et al., 2017) as the decomposition method. After determining variable interactions and components by DG2, the proposed algorithm uses a swarm for each detected component that works in a cooperative coevolutionary (CC) manner with other components' swarms. Each swarm consists of several sub-swarms to track multiple moving optima in the component's search space. Finally, the proposed method benefits from a new resource allocation approach in which it tries to prevent over exploitation by sub-swarms, and allocates more computational resource to the best sub-swarm of each component and the swarm of the component with the highest progress. Four algorithms are empirically evaluated on a wide range of problem settings to show the individual impact of approaches such as CC, tracking multiple moving optima and resource allocation on improving performance.

Chapter 7 concludes the thesis. Contributions of the thesis are summarized and future research directions are also suggested.

Chapter 2

Related work

Using evolutionary algorithms (EA) and swarm intelligence (SI) methods for optimizing DOPs is a popular and active research area and has increasingly attracted interest from the community of evolutionary computation (EC). Since the topics in this domain are very broad and diverse, it is impossible to cover everything in one chapter. Nevertheless, the topics related to this thesis are covered in this chapter.

A number of studies have been made in the past to review the literature in the field of DOPs and algorithms. Some first attempts were made by Branke (2002, 1999). The topic, as a part of the broader area of uncertainty and dynamic environments, was briefly surveyed and classified by Jin and Branke (2005). Cruz et al. (2011) have made a detailed review of DOP studies to provide an overview of related works on DOPs in the last decade. Nguyen et al. (2012a) carried out an in-depth survey of the state-of-the-art of academic research in the field using SI and EA for DOPs in four areas: benchmark problems/generators, performance measures, algorithmic approaches, and theoretical studies. Mavrovouniotis et al. (2017) presented a broad review on SI methods for optimizing DOPs focused on several classes of problems, such as discrete, continuous, constrained, multi-objective and classification problems, and real-world applications. Various aspects of using EA and SI for DOPs were also covered in many PhD theses and monographs (Weicker, 2003; Morrison, 2004; Nguyen, 2011; Branke, 2002; Younes, 2006; Goh and Tan, 2004).

2.1 Particle swarm optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Kennedy and Eberhart (1995), inspired by social behavior of bird flocking or fish schooling. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two *best* values. The first one is the best solution (based on fitness value) it has achieved so far which is called personal best (*Pbest*). Another *best* solution that particles move toward it is defined based on the neighborhood topology of the algorithm (Zavala, 2013). If the neighborhood topology is global star then this position is called global best (*Gbest*) and if the neighborhood topology is local based topologies like ring topology, then this position is called local best (*Lbest*). Each particle updates its velocity and positions with following equations:

$$\mathbf{v}^j_{i,t+1} = \chi \left(\mathbf{v}^j_{i,t} + c_1 r_{i,1}(g^*_i - x^j_{i,t}) + c_2 r_{i,2}(p^*_i - x^j_{i,t}) \right), \tag{2.1}$$

$$\mathbf{x}^j_{i,t+1} = x^j_{i,t} + \mathbf{v}^j_{i,t+1}, \tag{2.2}$$

where i is the index for the dimension. The constriction factor $\chi < 1$ acts like friction, slowing the particles, so that finer exploration is achieved (Eberhart and Shi, 2001). The inclusion of the previous generation's velocity in the calculation of the new velocity introduces a momentum into the particle's movement. c_1 and c_2 control the relative attraction to the global best and personal best found solutions, respectively. Finally, r_1 and r_2 are vectors of random variables drawn with uniform probability from [0, 1].

2.2 Tracking moving optima

TMO is the most popular approach in the DOP domain in which algorithms try to find the optimum and track it after each environmental change. One of the most important and challenging DOPs are problems with several competing local optima each having the potential to become the global optimum after an environmental change (Nguyen et al., 2012a). A multi-population strategy is one of the most effective approaches for solving this type of DOPs (Mavrovouniotis et al., 2017). Algorithms using this strategy have at least two subpopulations handling different tasks or covering different regions in the problem space.

Self Organizing Scouts (Branke et al., 2000) is a multi-population approach which utilizes a big subpopulation for global search and a number of small subpopulations for tracking changes of the identified peaks. This strategy has also been used with other metaheuristics such as PSO (Yazdani et al., 2013a; Li and Yang, 2008) and artificial fish swarm optimization (Yazdani et al., 2016, 2014).

Parrott and Li (2006) used a speciation method to split the population into subpopulations. Bird and Li (2007) presented a regression-based approach to enhance the convergence rate using speciation-based methods. Every subpopulation was confined to a hypersphere around the best solution.

A multi-population DE method was proposed in (du Plessis and Engelbrecht, 2012) which locates optima faster. This method was based on allowing subpopulations to compete for function evaluations based on their performance. This method also benefited from a reinitialization midpoint check mechanism that was aimed at maintaining subpopulations on different peaks.

Li and Yang (2009) proposed a method based on clustering for developing subpopulations, which was simplified and further improved in (Yang and Li, 2010). In (Du and Li, 2008), a method called MEPSO was proposed in which the population was divided into two parts. The first cluster was responsible for exploitation and the second one for exploration. Gaussian local search and differential mutation were used to improve diversity in the environment.

Blackwell and Branke (2006) proposed two multi-population methods, called MQSO and MCPSO. In MQSO, quantum particles appear at random positions, uniformly distributed around the swarm's global best. In MCPSO, some or all of the particles in each swarm have a 'charge', and charged particles repel each other, leading to larger diversity. The population size is equal for every sub-swarm, and the number of sub-swarms is fixed and pre-determined. An anti-convergence method ensures continued search for possible better peaks. In addition, a mechanism called exclusion is used to avoid several swarms converging to the same peak. A version of MQSO with an adaptive number of subpopulations, called AMQSO, was proposed in (Blackwell et al., 2008). AMQSO starts with one subpopulation and a new subpopulation is created if all previous subpopulations have converged. This method has significantly improved the performance.

Li et al. (2016) proposed a method to adapt the number of populations based on statistical data on how many populations have found new peaks. If this number is large, more populations will be introduced and vice versa. Additionally, a new heuristic clustering, a population hibernation scheme, a population exclusion scheme, a peak hiding method and two movement methods (to track peaks and avoid stagnation) were proposed.

A PSO with two types of sub-swarms called finder-tracker multi-swarm PSO was proposed in (Yazdani et al., 2013b). The finder swarm finds new uncovered peaks. When it converges to a peak, it creates a new tracker swarm to track the peak. An exclusion mechanism re-initializes the finder swarm if it converges to a peak that already has a tracker swarm on it. In addition, a mechanism to schedule tracker swarms called sleeping-awakening was proposed. It allocates more computational power to more promising swarms. Furthermore, a new method for re-diversification of tracker swarms (after a change) was proposed. The method re-initializes all particles randomly around the global best (Kennedy and Eberhart, 1995) and their velocity vector is randomly set based on the peak's shift severity.

Sharifi et al. (2015) proposed a PSO-based method in which a fuzzy social-only model PSO and local search were combined. In this method, a swarm of a fuzzy social-only model PSO was responsible for locating peaks in the search space. When a new peak was discovered by the PSO, a local search agent was created to cover the peak and track it after environmental changes. For controlling computational resources, three different methods were proposed to allocate the computational resources to the most promising areas of the search space.

Recently, in (Luo et al., 2017), for the first time, partially separable DOPs were investigated and a divide and conquer method was used in order to solve them. This method used differential grouping (Omidvar et al., 2014a) for detecting interactions between decision variables, then it used a species-based PSO as its optimizer (Parrott and Li, 2006; Preuss, 2010). This method utilized some initial knowledge about the number of peaks in each subfunction and number of generations between successive environmental changes which violates the black-box assumption.

2.3 DOP Benchmarks

Cobb and Grefenstette (1993) proposed a switching function method in which two landscapes A and B are used to generate the following three types of change: 1) Linear translation of peaks in A; 2) Changing the location of the optimum randomly while the rest of the search space remain unchanged; and 3) Switching between landscapes A and B.

Branke's Moving Peaks Benchmark (MPB) (Branke, 1999) is the most widely used benchmark in DOP. The search space generated by this benchmark consists of several peaks whose width, height, and location change over time. MPB is very flexible to generate functions with configurable dimensions, number of peaks, and peak dynamics. In the standard MPB, the widths and heights of peaks are changed by adding Gaussian noise. In some other studies such as (Huang et al., 2017; Fu et al., 2015), the width and height of each peak change using different dynamics such as small step, large step, and recurrence (Li et al., 2008). Similar to MPB, DF1 (Morrison and Jong, 1999) generates problem instances in which the width, height, and location of peaks change over time. The nature of the changes can be controlled by a logistic function to generate fixed, chaotic, or bifurcated step sizes. Another benchmark whose landscape consists of several peaks is Gaussian peak (Grefenstette, 1999). In this benchmark, the location of peaks change in random directions and the step sizes are uniformly distributed over an interval controlled by two levels of severity called abrupt and gradual (Grefenstette, 1999).

Dynamic rotation is another method for creating dynamic changes (Weicker and Weicker, 2000). In this benchmark, the landscape is combined with a visibility mask which allows a percentage of the search space to be masked with a predefined fitness value. GDBG (Li et al., 2008) is another benchmark generator that uses rotation as well as shifting to generate environmental changes. The magnitude of change in GDBG is defined using a rotation angle.

2.4 Switching cost in dynamic optimization problems

There are only a few papers that consider SC. In (Huang et al., 2017; Salomon et al., 2013; Avigad et al., 2010; Yazdani et al., 2018b), SC was considered as an objective in multi-objective problems. However, all of these works considered the SC as a separable and independent objective from the optimality objective function and the connection between these two objectives was not considered. Salomon et al. (2013) investigated SC as optimization of adaptation. A multi-objective problem was defined which considered the cost of the adaptation and the optimality while the adaptation takes place. In (Avigad et al., 2010), the need for rapid, low-cost changes in a design, in response to changes in performance requirements, within multi-objective problems, was investigated. An algorithm called ROOT/SC (Huang et al., 2017) was designed for ROOT. ROOT/SC is a multi-objective algorithm in which the first objective is survival time metric (Fu et al., 2013) and the second one is SC.

In real-world scenarios, usually SC is very important and needs to be considered. For example, in scheduling, changing the schedule may have significant impact on suppliers and customers, or, in the design of telephone networks, sending out engineers to change the physical infrastructure can be very expensive. In the taking-off/landing scheduling problem, it is desirable to keep the current implemented solution/schedule after an environmental change (E.Wilkins et al., 2008; Atkin et al., 2008) to avoid unfavorable disruptions in airport operations. ROOT methods were designed to avoid changing solutions as much as current solutions remain acceptable (Yu et al., 2010; Fu et al., 2013; Jin et al., 2013). Therefore, ROOT methods aim to minimize switching costs.

2.5 Robust optimization

The term “robust optimization” has come to encompass several approaches to protecting the decision-maker against parameter ambiguity and stochastic uncertainty (Gabrel et al., 2014). Uncertainty may affect the feasibility of a solution. In such circumstances, robust optimization seeks to obtain a solution that will be feasible for any realization taken by the unknown coefficients; however, complete protection from adverse realizations often comes at the expense of a significant deterioration in the objective. Moreover, uncertainties may happen in the system output (Beyer and Sendhoff, 2007). These uncertainties are due to imprecision in the evaluation of the output and performance of the system. This kind of uncertainty includes measuring errors and all kinds of approximation errors due to the use of models instead of the real physical objects (model errors). In addition, uncertainties could be results of environmental changes. These uncertainties enter the system via the θ-variables in Eq. (1.1) (Jin et al., 2013).

uncertainties in Objective which is called aleatory (Helton, 1997) or random uncertainties, are of intrinsically irreducible stochastic nature. That is, these kinds of uncertainties are of physical nature, e.g., the noise in electrical devices, humidity, wind load, quantum mechanical effects, temperature, server load, and material parameters. These uncertainties cannot be removed and must be considered in the optimization. Epistemic uncertainties are results of the lack of information about the problem of interest during designing process. These uncertainties are regarded as subjective, because it is due to a lack of knowledge that could, in principle, be reduced by increased efforts. Epistemic uncertainties include uncertainties about the model used to describe the reality, its operation conditions and boundary, also referred to as model form errors (Mahadevan and Rebba, 2006), and also the errors introduced by the numerical solution methods used (e.g. approximation error, convergence problems, discretization error).

Greiner (1994) used an evolution strategies algorithm for the evolution of robust optical filter designs. Du et al. (2018) proposed a multi-objective approach for robust order scheduling problems in the fashion industry by considering the preproduction events and the uncertainties in the daily production quantity. Pictet et al. (1998) proposed a genetic algorithm for robust optimization for financial time series prediction. A robustness measure according to mean value and variance, and a fitness sharing model were proposed for avoiding a high concentration of individuals in the vicinity of sharp peaks. In (Peng et al., 2016) a compound differential evolution algorithm was proposed for flexible robust optimization for hybrid power system for achieving coordination between reliability and economy. In (Lim et al., 2006) a priori knowledge on the desired robustness of the final design was used with a multi-objective evolutionary algorithm that converges to a solution with good nominal performance and maximal robustness.

Fertis (2009) and Xu (2009) investigated the connection between successful learning and robustness, and applications of robust decision-making to statistical estimation and machine learning. (Nguyen and Lo, 2012) investigated robust estimation and regression and presented computationally efficient methods for robust meancovariance estimation and robust linear regression using special mathematical programming models and semi-definite programming. Xu et al. (2009) analyzed regularized support vector machines and showed an equivalence with a robust optimization formulation, with implications both for analysis and algorithms. (Caramanis et al., 2011) described robust optimization in the context of machine learning in detail.

In this thesis, uncertainties resulted by environmental changes are investigated. As described in Chapter 1.1, Eq. (1.1) is considered as the DOP problem in this thesis. One widely used definition of robust solutions is a solution's expected performance over all possible disturbances Jin et al. (2013). Therefore, the resultant fitness of Eq. (1.1) is:

$$F(\mathbf{x}) = \int_{-\infty}^{+\infty} f(\mathbf{x} + \delta, \theta) p(\delta) d(\delta), \tag{2.3}$$

or

$$F(\mathbf{x}) = \int_{-\infty}^{+\infty} f(\mathbf{x}, \theta + \xi) p(\xi) d(\xi), \tag{2.4}$$

where δ and ξ show the noise in the design variables and environmental parameters, respectively. $p(\delta)$ and $p(\xi)$ are the probability density functions of δ and ξ, respectively.

2.6 Robust optimization over time

Yu et al. (2010) proposed ROOT as a new perspective on DOPs. A new framework for ROOT was proposed by Jin et al. (2013) with the algorithm searching for robust solutions by means of local fitness approximation and prediction. This method consists of a population-based optimization algorithm, a fitness approximator (to estimate fitness at any point in the search space), a fitness predictor (to predict future fitness values) and a database. In (Jin et al., 2013), an adapted radial-basis-function network (RBFN) is the local approximator and an autoregressive (AR) is the predictor. A database was used for storing, in each iteration, all of the individuals' positions alongside their fitness values and the associated time of storage. This database was then used for approximating fitness values of solutions in previous environments which in turn was used for training the predictor.

It is important to have sufficient samples across the search space in this database to maintain the accuracy of the approximation. Since optimization algorithms quickly converge to the most promising region in the search space, there would be regions that they would not visit (such as regions with bad fitness) which still are necessary for having a good training data. On the one hand, in (Jin et al., 2013), the algorithm needs to have enough information to be able to predict any solution in the search space which is dependent on the approximator. On the other hand, for having an appropriate approximation model, the training data needs to be properly distributed in the search space. For achieving this, in (Jin et al., 2013), the authors generate half of the population using a specific hypercube design after each environmental change. Therefore, for each environment, the database can contain at least one solution from each hypercube. However, in larger environments such as ones with larger search ranges and higher dimensions, the number of these hypercubes increases exponentially and becomes larger than the population size. As a result, the algorithm needs to evaluate a solution for each hypercube for adding to the database. These additional fitness evaluations become a challenge in larger problems.

To select a robust solution, (Jin et al., 2013) uses the sum of the solutions' current fitness value, its p previous fitness values (provided by the approximator) and its q future fitness values (provided by the predictor):

$$F(\mathbf{x}) = \sum_{l=t-p}^{t-1} \check{f}(\mathbf{x}, \theta^{(l)}) + f(\mathbf{x}, \theta^{(l)}) + \sum_{l=t+1}^{t+q} \hat{f}(\mathbf{x}, \theta^{(l)}), \tag{2.5}$$

where F is the sum of the approximated ($\check{f}$), actual (f), and predicted ($\hat{f}$) fitness values of $\mathbf{x}$ at time t. The performance of the proposed method in (Jin et al., 2013) depends on the accuracy of the approximation and prediction methods. In (Jin et al., 2013), a particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) was used as the optimizer. In addition, several performance indicators were proposed, of which one of the most important ones is E_{avg}, the average error of the robust solution sequence $S = (\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_k)$,

$$E_{\text{avg}} = \frac{1}{k} \sum_{i=1}^{k} e_i, \tag{2.6}$$

where

$$e_i = \frac{1}{n_i} \sum_{j=t_i}^{t_i+n_i-1} \left| \text{opt}^{(j)} - f(\mathbf{r}_i, \theta^{(j)}) \right|, \tag{2.7}$$

$\text{opt}^{(j)}$ is the optimum fitness value at the j^{th} environment, $\mathbf{r}_i$ is the i^{th} robust solution,

n_i is the number of environments for which $\mathbf{r}_i$ remained acceptable, and t_i is the time that $\mathbf{r}_i$ was chosen.

Another performance indicator is ρ, the robustness rate of the robust solution sequence,

$$\rho = 1 - \frac{k-1}{T-1}, \tag{2.8}$$

where k is the number of robust solutions. In Eq. (2.8), a smaller k causes ρ to increase and the ideal situation happens when the first robust solution can remain acceptable in all environments, i.e. $k = 1$. In addition, a new condition for checking whether a robust solution may be kept in a new environment was introduced. According to this condition, given a user defined threshold δ_{drop}, a robust solution $\mathbf{r}_i$ may be kept in the j^{th} environment if:

$$\left| \frac{f(\mathbf{r}_i, \theta^{(j)}) - \text{opt}^{(j)}}{\text{opt}^{(j)}} \right| \le \delta_{\text{drop}}. \tag{2.9}$$

Since $\text{opt}^{(j)}$ is not known usually, it can be replaced by the best position found by the algorithm in the j^{th} environment.

Fu et al. (2013) proposed two new robustness definitions and metrics, namely *survival time* and *average fitness*. The survival time is the maximum time interval starting from time t during which the fitness value of the robust solution remains acceptable:

$$\mathcal{S}\left(\mathbf{x}, \theta^{(t)}, V\right) = \begin{cases} 0 & \text{if } f(\mathbf{x}, \theta^{(t)}) < V \\ 1 + \max\{l \mid \forall i \in \{t, \dots, t+l\} : f\left(\mathbf{x}, \theta^{(i)}\right) \ge V\} & \text{otherwise} \end{cases} \tag{2.10}$$

where V is a user defined threshold. In Eq. (2.10), for each environment, $\mathcal{S}$ shows for how many environments the fitness value of the current solution has remained above V. Note the threshold V in (Fu et al., 2013) is easier to use than δ_{drop} in Eq. (2.9) from (Jin et al., 2013) which requires to know the optimum.

The robust solution is selected based on the predicted average fitness over a pre-defined time window ω as follows:

$$\mathcal{A}\left(\mathbf{x}, \theta^{(t)}, \omega\right) = \frac{1}{\omega} \sum_{i=0}^{\omega-1} f\left(\mathbf{x}, \theta^{(t+i)}\right). \tag{2.11}$$

When equations (2.10) and (2.11) are used as metrics, $f(\mathbf{x}, \theta^{(i)})$ for $i > t$ is the predicted fitness value of $\mathbf{x}$ in t^{th} environment ($\hat{f}(\mathbf{x}, \theta^{(t)})$) instead of its actual fitness value. In the experiments in (Fu et al., 2013), the authors assumed that the algorithm benefited from an ideal approximator so they used true previous fitness values instead of approximated fitness values for training the predictor. Consequently, the reported results in (Fu et al., 2013) were not subject to approximation errors. Additionally, a ROOT performance indicator was proposed in (Fu et al., 2013) based on Eq. (2.10) and Eq. (2.11) as follows:

$$\mathcal{P} = \frac{1}{T}\sum_{i=1}^{T}\mathcal{F}(i), \tag{2.12}$$

where $\mathcal{P}$ is performance of ROOT algorithm, $\mathcal{F}(i)$ is either $\mathcal{S}$ in Eq. (2.10) or $\mathcal{A}$ in Eq. (2.11). In (Fu et al., 2015), two definitions of ROOT in (Fu et al., 2013) i.e. survival time and average fitness were analyzed. Also, two different benchmark problems were proposed.

Guo et al. (2014) proposed a new two-layer multi-objective method to find robust solutions that can maximize both survival time and average fitness. Huang et al. (2017) proposed another multi-objective method for minimizing switching cost and maximize survival time. A PSO algorithm was used as the optimizer. Additionally, the algorithm used the acceptance threshold for robust solutions similar to (Fu et al., 2013). Euclidean distance between two solutions was used as the switching cost, and three different performance indicators were used:

$$\mathcal{F} = \frac{1}{T}\sum_{i=1}^{T}F_i, \tag{2.13}$$

$$\mathcal{R} = \frac{1}{T}\sum_{i=1}^{T}R_i, \tag{2.14}$$

$$\mathcal{C} = \frac{1}{T}\sum_{i=1}^{T}C_i, \tag{2.15}$$

where T is the number of environments, F_i is the fitness value of robust solutions, R_i is robustness (calculated by Eq. (2.10)) and C_i is switching cost in i^{th} environment. Switching cost is the Euclidean distance between robust solutions in successive environments.

All of the proposed methods by Jin et al. (2013); Fu et al. (2013, 2015); Huang et al. (2017) used predicted future fitness values of solutions for selecting robust solutions. Jin et al. (2013) used an RBFN for approximating previous fitness values of solutions and an AR was used for predicting future values. Fu et al. (2013) removed the approximation part and used true fitness values in previous environments for training the

AR in order to remove the negative effect of approximation errors on the performance of the algorithms. In (Fu et al., 2013), the authors used the same methods as in (Jin et al., 2013) for approximation and prediction to investigate the performance of the proposed algorithms in (Jin et al., 2013; Fu et al., 2013). In (Huang et al., 2017), the authors assumed that algorithms benefited from an ideal predictor without any error, so in their experiments the true future fitness values were used instead of the predicted values. However, removing the approximator and predictor from algorithms that work based on future fitness values of solutions clearly is a substantial simplification and the performance on a real-world problem where future fitness values are not available may be very different. Overall, for solving real-world problems, almost all the current ROOT methods (Jin et al., 2013; Fu et al., 2013, 2015; Guo et al., 2014; Huang et al., 2017) need to use approximation and prediction methods based on time series (Box et al., 2015).

2.7 Dynamic multi-objective optimization problems

For solving many dynamic real-world problems, the optimizer needs to simultaneously optimizing several competing objectives. These problems are dynamic multi-objective optimization problems (DMOOPs). In DMOOPs, the optimization goal is not only to evolve a near-optimal Pareto optimal front (POF), but also to continually and rapidly discover the desired one before the next environmental change (Azzouz et al., 2017). Farina et al. (2004) classified DMOOPs into four different types according to changes affecting the POF and the Pareto optimal solutions (POS) as follows:

(I). where the POS changes while the POF remains invariant;

(II). Where both POS and POF change;

(III). Where POF changes while POS remains invariant; and

(IV). Where both POS and POF remain invariant.

In (Farina et al., 2004), a baseline algorithm for DMOOPs as well as some test problems were suggested. The use of evolutionary multi-objective optimization methods in DOPs were investigated in (Bui et al., 2005). The first objective was the original single dynamic objective and the second objective was an artificial objective to promote diversity. In (Wei and Wang, 2012), a PSO algorithm for tracking the varying POS and POF obtained at each environment was developed in which a hyper rectangle search was used for predicting the optimal solutions of the next environment. additionally, a new crossover operator was designed for handling constraints. Wei and Jia (2013) follows the same goals as (Wei and Wang, 2012). The proposed PSO in (Wei and Jia, 2013) used a new points selection strategy for initializing swarm at each environment. Moreover, a local search operator was proposed to improve the exploitation.

Deb et al. (2007) proposed a Dynamic NSGA-II in which the diversity is introduced after each environmental change. After change detection, all parent positions were re-evaluated. This process allowed both offspring and parents to be evaluated using the changed objectives and constraints functions. Chen et al. (2009) proposed to a diversity maintenance method by considering the population diversity as an additional artificial objective. The goal of the proposed approach was to add a useful selection pressure addressed towards both the POS and the diversity maintenance.

Hatzakis and Wallace (2006) proposed a forecasting technique called Feed-forward Prediction Strategy for estimating the location of the POS. Then an anticipatory population called a prediction set is placed in the neighborhood of the forecast for accelerating the discovery of the POS in the new environment. Zhou et al. (2007) proposed an approach that predicted the new locations of a number of Pareto solutions in the search space when an environmental change is detected. At the beginning of each environment, individuals in the re-initialized population are generated around these predicted points.

A prediction strategy called dynamic predictive gradient strategy was proposed by Koo et al. (2010) for predicting the magnitude of changes and provide proper search direction in the changed environment. Moreover, a computational resource-aware memory technique was proposed for exploiting any periodicity in the dynamic problem. Liu (2010) proposed a Dynamic Multi-objective EA with core estimation of distribution which incorporates a core estimation of distribution model for predicting the location of POS in the next environment. In (Zhou et al., 2014), an approach called population prediction strategy was proposed in which a whole population was predicted rather than predicting some isolated points. This approach, divides the POS into two parts i.e. a center point and a manifold. (Muruganantham et al., 2016) proposed a dynamic multi-objective method in which a Kalman Filter-based prediction model was used. After each environmental changes, Kalman Filter is applied to the whole population to direct the search towards the new POS in the search space.

Wang and Li (2009) proposed several memory-based dynamic environment handling mechanisms to effectively utilizing the information from previous environments for accelerating and improving the optimization process after each environmental change. These mechanisms, including restart, explicit memory, local-search memory and hybrid memory, were working based on the stored archive solutions. After each environmental change, the new population is composed by: 1) random solutions in addition to memory ones using the explicit memory mechanism, 2) random solutions and solutions obtained by performing a Gaussian local search using the local-search memory mechanism or 3) random solutions, memory solutions and solutions generated by application of a local search using the hybrid memory mechanism.

In (Zheng, 2007), a dynamic multi-objective optimization EA was proposed in which the population is divided into $m + 1$ sub-populations where m is the number of objectives. Each of m sub-populations are responsible for optimizing one objective and the remained one subpopulation optimizes the average fitness value of all objectives. (Camara et al., 2007) proposed a procedure for adapting the Parallel Single Front Genetic Algorithm to DMOOPs. This approach was a parallel algorithm which used a dispatcher-worker scheme.

2.8 Dynamic time-linkage optimization problems

According to (Nguyen, 2011), many real-world continuous DOPs have time-linkage characteristics. The time-linkage property in DOPs refers to the fact that any solution chosen by the optimizer in the current environment could change the future problem space (Nguyen et al., 2012b). In dynamic time-linkage optimization problems (DTP), the optimizer should consider both the present and future i.e. it needs to predict the future behavior of the environment based on the chosen solution. DOPs with SC are DTPs because choosing a solution for the current environment will change the next environment's search space of the displacement/SC objective. In fact, when a solution is chosen for an environment, all the feasible solutions will be evaluated based on their distance from it in terms of displacement/SC.

Despite importance of time-linkage in DOPs, a very small number of researchers consider time-linkage property in their investigations and there are only few papers in the evolutionary dynamic optimization (EDO) academic community that investigate problems with the time-linkage property (Nguyen and Yao, 2009; Bosman, 2005; Branke and Mattfeld, 2005; Ursem et al., 2002). For solving online DTPs, Bosman (2005) pointed out that the optimizer needs to consider the feature of time-linkage, and for evaluating a solution, take its future influence into account alongside its current fitness value, and the method of optimizing both the present and the future is suggested to solve DTPs, instead of the traditional method of optimizing only the present (Bu et al., 2017). A prediction method EA + predictor (Bosman, 2005), was proposed in which the method made decisions according to both the current and the predicted fitness. This method was improved in (Bu et al., 2017) in order to enhance the performance in situations in which prediction is unreliable.

Bosman and Poutré (2007) investigated DTPs in stochastic environments in which environmental changes are not deterministic and proposed a strategy optimization method. At the first step, a problem-dependent strategy needs to be designed for the proposed strategy optimization method which is a decision maker that determines what to do in any given scenario. Then, the parameters of the strategy will be optimized during the evolutionary process. Bosman and Poutré (2007) indicated that the strategy optimization outperforms traditional expected value approaches and scenario-based optimization approaches.

Nguyen et al. are among the main pioneers in the field of solving DTPs by EDO (Nguyen, 2011; Nguyen et al., 2012b; Nguyen and Yao, 2013). In Nguyen and Yao (2009), Nguyen and Yao showed that using prediction cannot be efficient for all problems, because the information from the past could stand against the future. This type of DTPs is called prediction-deceptive by Nguyen and Yao (2013). They showed that predictors that utilize historical data for training, are not suitable for solving prediction-deceptive DTPs and even could achieve worse results in comparison with predictor-less methods. In fact, designing an appropriate approach to solve black-box prediction-deceptive DTPs (Nguyen and Yao, 2009) might be very hard or even impossible. However, in grey-box prediction-deceptive DTPs where some problem-specific information is provided, especially the knowledge of switching rules is available, avoiding being deceived by a prediction method is possible (Nguyen and Yao, 2009).

Scheduling and resource allocation (SRA) is classified as DTPs because the solution already in use affects the situation when a change happens. In SRA, the problem is allocating limited resource to tasks over time (Bui et al., 2012). Dynamic factors in SRA include failure of resources and arrival of new tasks. In (Bui et al., 2012; Branke and Mattfeld, 2000), it is argued that a suitable but inflexible solution made in the current environment might deteriorate the system efficiency when a change happens in the future. Branke and Mattfeld (2000) anticipated the flexibility of a solution and incorporated it into the objective function. Bui et al. (2012) modeled the dynamic planning problem as a multi-objective problem. After each environmental change, the current solution is adapted by information provided by a set of Pareto optimal solutions produced after each planning cycle.

2.9 Scaling up dynamic optimization problems

Despite the large body of literature on DOPs and algorithms, little attention has been given to their scalability. Indeed, the dimension of a typical DOP studied in the literature hardly exceeds twenty. Motivated by rapid technological advancements, large-scale optimization has gained popularity in recent years. However, the exponential growth in the size of the search space, with respect to an increase in the number of the decision variables, has made large-scale optimization a challenging task. For DOPs, however, the challenge is twofold. For such problems, not only should an algorithm be capable of finding the global optimum in the vastness of the search space but should also be able to track it over time. For multi-modal DOPs, where several optima have the potential to turn into the global optimum after environmental changes, the cost of tracking multiple moving optima also adds to the complexity. One of the best ways for solving large-scale problems is to determine variable interactions and using cooperative coevolution (CC) methods (Omidvar et al., 2017).

2.9.1 Variable interaction

Variable interaction or linkage refers to the extent to which the optimum of a variable depends on the values taken by other decision variables. For continuous optimization problems, variable interaction is defined as follows (Mei et al., 2016):

Definition 2.1 (Mei et al. (2016))**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. Decision variables x_i and x_j interact if a candidate solution $\mathbf{x}^\star$ exists, such that

$$\frac{\partial^2 f(\mathbf{x}^\star)}{\partial x_i \partial x_j} \neq 0. \tag{2.16}$$

Some functions exhibit an underlying interaction structure such that groups of decision variables can be optimized independently. These functions, which are called *partially separable*, are defined as follows:

Definition 2.2 (Omidvar et al. (2015))**.** A function $f(\mathbf{x})$ is partially separable with m independent components iff:

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \Big(\arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \dots), \dots, \arg\min_{\mathbf{x}_m} f(\dots, \mathbf{x}_m)\Big),$$

where $\mathbf{x} = (x_1, \dots, x_n)^\top$ is a decision vector of n dimensions, $\mathbf{x}_1, \dots, \mathbf{x}_m$ are disjoint sub-vectors of $\mathbf{x}$, and $2 \leq m \leq n$.

Additive separability is a special type of partial separability, which is defined as follows:

Definition 2.3 (Omidvar et al. (2015))**.** A function is additively separable if it has the following general form:

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_i), \; m > 1,$$

Algorithm 1: $(x^\star, f^\star) = \texttt{CC}(f)$

```
/*Main Framework of CC*/
$\mathbf{P} \leftarrow$ randomized initial population;
$\mathbf{c} \leftarrow$ randomized initial context vector;
//grouping stage
$\mathcal{G} = \texttt{Grouping}(f)$;
//optimization stage
while Termination Condition is Not Satisfied do
    for $\kappa = 1$ to $|\mathcal{G}|$ do
        $(\mathbf{P}, \mathbf{c}) = \texttt{Optimizer}(\mathbf{P}, \mathbf{c}, \mathcal{G}_\kappa)$;
$\mathbf{x}^\star = \mathbf{c}$ ; $f^\star = f(\mathbf{x}^\star)$ ;
return $(\mathbf{x}^\star, f^\star)$;
```

where $f_i(\cdot)$ is a nonseparable subfunction, and m is the number of nonseparable components of f. The definition of $\mathbf{x}$ and $\mathbf{x}_i$ is identical to what was given in Def. 2.2.

Definition 2.4 (Omidvar et al. (2015))**.** A function $f(\mathbf{x})$ is fully nonseparable if every pair of its decision variables interact with each other.

2.9.2 Cooperative coevolution

Cooperative coevolution (CC) has been proposed by Potter and Jong (2000) with the goal of allowing evolutionary algorithms the capacity to solve increasingly complex problems. The idea is based on decomposing a complex problem into subproblems of lower complexity which are coadapted within an evolutionary context. Algorithm 1 shows a high-level representation of CC. In the original implementation of CC, an n-dimensional problem is decomposed into n 1-dimensional problems each of which is optimized using a given optimizer in a round-robin fashion. In order to assign a fitness to each partial solution within a component, the individuals are evaluated within the context of a complete solution often referred to as the *context vector* (van den Bergh and Engelbrecht, 2004).

The round-robin optimization of components assumes a uniform contribution from each component which is often not the case for various reasons (Omidvar et al., 2015). The so-called *imbalance* among the contribution of components can be attributed to the following:

1. Nonuniform dimensionality of the underlying component functions.
2. Component functions with different landscapes and output ranges.
3. The dynamics of the optimizer, its convergence behavior, and stagnation.

Contribution-based cooperative coevolution (CBCC) (Omidvar et al., 2011, 2016) is an improved CC framework which addresses the imbalance issue by assigning more resources to components with higher overall contributions. An important aspect of a contribution-aware coevolutionary framework is maintaining an optimal balance between an exploration phase in which the contribution of components is updated, and an exploitation phase in which the most contributing component is optimized. This has resulted in many attempts to design various exploration/exploitation polices (Yang et al., 2017; Mahdavi et al., 2017b,a).

The original CC framework and its contribution-based counterpart has no explicit means of dealing with variable interactions. They only respond to interactions through the *cooperation* of individuals in updating the context vector, which acts as a message passing mechanism. The efficiency of this approach depends on the policy of constructing the context vector (Wiegand et al., 2001) as well as its update frequency (Omidvar et al., 2010). To alleviate this problem, many variable interaction analysis algorithms have been proposed with the aim of decomposing a large-scale problem into smaller *independent* components. There have been many attempts (Yang et al., 2008a; Chen et al., 2010) on this, among which the differential grouping family of algorithms showed the highest accuracy (Omidvar et al., 2014a; Mei et al., 2016; Sun et al., 2017). Differential grouping (DG) works on the basis of the following theorem:

Theorem 2.5 (Omidvar et al. (2014a))**.** *Let $f(\mathbf{x})$ be an additively separable function. $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, variables x_p and x_q interact if the following condition holds*

$$\Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_2}, \tag{2.17}$$

where

$$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(\dots, x_p + \delta, \dots) - f(\dots, x_p, \dots), \tag{2.18}$$

refers to the forward difference of f with respect to variable x_p with interval δ.

The quantities in Eq. (2.17) are real-valued numbers; therefore, the equality check cannot be evaluated exactly over the floating-point number field on computer systems. Consequently, the equality check needs to be converted to an inequality check by introducing a sensitivity parameter: $|\Delta^{(1)} - \Delta^{(2)}| > \epsilon$. Here, $\Delta^{(1)}$ and $\Delta^{(2)}$ denote the left and right hand side of Eq. (2.17), respectively. In the absence of representation and roundoff errors, ϵ can be theoretically set to zero; however, this is not usually the case and the optimal value of ϵ is often a nonzero positive number. This parameterization makes DG sensitive to choices of ϵ whose optimal value may vary from function to function and is difficult to tune by practitioners. To alleviate this problem, Omidvar et al. (2017) proposed DG2, a parameter-free version of DG, which automatically sets ϵ by estimating the bounds on the computational roundoff errors to maximize the accuracy of variable interaction detection.

2.9.3 Decomposition algorithms

Many decomposition algorithms have been proposed to decompose a black-box optimization problems into smaller subproblems. Static grouping is the simplest decomposition strategy in which the decision variables are grouped into arbitrary groups. In its simplest form, an n-dimensional problem is broken down into s k-dimensional problems. Examples of such methods are the divide-in-half method by jun Shi et al. (2005), and the method employed by van den Bergh and Engelbrecht (2004). These methods are oblivious of variable interactions which may have a significant impact on the optimization performance (Yang et al., 2008a).

Some other decomposition algorithms such as random grouping (Yang et al., 2008a), adaptive variable partitioning (Ray and Yao, 2009), delta grouping (Omidvar et al., 2010), and min/max variance decomposition (Liu and Tang, 2013) use various heuristics in order to form the groups based on variable interaction characteristics of the objective function. The drawback of these methods is their low grouping accuracy, and the fact that they presuppose the number and/or the size of components. These algorithms also divide the decision variables into s k-dimensional components. Improved versions of random grouping and delta grouping use a so-called multilevel strategy (Omidvar et al., 2010; Yang et al., 2008b) in which multiple fixed decompositions are used over the course of optimization.

More sophisticated decomposition methods such as variable interaction learning (Chen et al., 2010), meta-modelling decomposition (Mahdavi et al., 2017a), statistical learning decomposition (Sun et al., 2012), and differential grouping (Omidvar et al., 2014a) do not presuppose the number and/or size of components. Among these algorithms, differential grouping has shown superior performance with respect to grouping accuracy (Chen et al., 2010; Mahdavi et al., 2017a). Two major drawbacks of differential grouping are its sensitivity to the parameter ϵ and its poor accuracy in detecting interacting variables on functions with overlapping components. As reported in (Omidvar et al., 2014a), the grouping accuracy of differential grouping is low on the Rosenbrock function which has overlapping components with overlap size of one. Also, if differential grouping is used to find the interaction structure of functions with overlapping variables, the shared decision variables between two components will be placed in one group and will be excluded from other groups. It is not yet clear what an optimal decomposition may be for an overlapping function; nevertheless, an accurate identification of the underlying structure is essential to propose a meaningful decomposition.

Global Differential Grouping (GDG) (Mei et al., 2016), extended Differential Grouping (XDG) Sun et al. (2015), and DG2 Omidvar et al. (2017) are three variants of differential grouping, which aim at addressing the above shortcomings. XDG focuses on identifying indirect interactions in order to deal with the Rosenbrock function. The issue with XDG is that it inherits the sensitivity issue of differential grouping and also its method of inferring variable interaction may consider separable variables as nonseparable. GDG addresses the sensitivity issue of differential grouping by taking computational errors into account. However, the use of a global parameter to detect all interactions makes it unsuitable for imbalanced functions. GDG also addresses the problem of identifying overlapping functions by examining all pairs of variables for interaction. DG2 is a parameter-free version of DG, which automatically sets ϵ by estimating the bounds on the computational roundoff errors to maximize the accuracy of variable interaction detection.

2.9.4 Large-scale dynamic optimization problems

Recently, in (Luo et al., 2017), for the first time, partially separable DOPs were investigated and a divide and conquer method was used in order to solve them. This method used DG (Omidvar et al., 2014a) for detecting interactions between decision variables, then it used a species-based PSO as its optimizer (Parrott and Li, 2006; Preuss, 2010). This method utilized some initial knowledge about the number of peaks in each subfunction and number of generations between successive environmental changes which violates the black-box assumption. To the best of our knowledge, (Luo et al., 2017) is the only attempt to solve large-scale DOPs until now. Using prior knowledge about the problem, lack of analysis of the problem and uniformity in size of subfunctions in benchmark are among issues that make the proposed method in (Luo et al., 2017) unrealistic.

2.10 Summary

This chapter reviewed some fields in the optimization by EA and SI approaches. In this thesis, PSO (Subsection 2.1) is used as the core component optimizer in a multi-population framework (Subsection 2.2).

Switching cost (SC) is very important in DOPs. As described in Subsection 2.4, despite the importance of SC in DOPs, most of the academic research in this field have not considered it in designing algorithms. Lack of considering SC in designing algorithms for DOPs makes them unsuitable for many real-world DOPs. In fact, changing solutions in real-world problems is often costly. Furthermore, larger changes have higher cost and need more resources such as time, human resources and energy. Moreover, in many real-world DOPs, changing solutions frequently is not desirable or it may be very costly. Therefore, this thesis investigate DOPs with SC, their characteristics, and their challenges. Then new approaches for solving them are designed. Proposed algorithms in Chapters 3, 4 and 5 are designed to perform in dynamic environments under different conditions based on SC.

Robust optimization over time (ROOT) methods were reviewed in Subsection 2.6. ROOT is a way to handle DOPs with large SC. Additionally, ROOT is a way to perform robust optimization for problems with uncertainties in the objective function (Subsection 2.5). As reviewed in Subsection 2.6, all the previous ROOT methods (Jin et al., 2013; Fu et al., 2013, 2015; Guo et al., 2014; Huang et al., 2017) need to use approximation and prediction approaches based on time series (Box et al., 2015). As mentioned in Subsection 2.6, using algorithms that works based on approximation and prediction methods is not suitable for DOPs with larger search space. Therefore, a new framework for ROOT is proposed in Chapter 4 which does not use approximation and prediction methods and outperforms the state-of-the-art ROOT methods, especially, in larger problems. In Additionally, in Chapter 5, a novel adaptive algorithm is proposed which acts in a similar way to ROOT methods when SC is high.

As described in Subsections 2.4 and 1.3.1.1, DOPs with PSDR have not been investigated before, therefore, in Chapter 3, this class of DOP is investigated and a new approach is proposed for solving them. Chapter 3 shows that DOPs with PSDR has time-linkage property which were reviewed in Subsection 2.8. Furthermore, DOPs with PSDR have two objectives which categorize them as dynamic multi-objective optimization problems which were reviewed in Subsection 2.7.

As described in Subsection 2.9, little attention has been given to DOPs' scalability. Chapter 6 investigates large-scale DOPs. The moving peaks benchmark is analyzed and its limits for the study of large-scale DOPs are shown. A new benchmark generator based on the moving peaks benchmark is proposed for large-scale DOPs. Moreover, a cooperative coevolutionary multi-swarm PSO (CCMPSO) is proposed for solving this class of DOPs.

Chapter 3

A multi-objective time-linkage approach for dynamic optimization problems with previous-solution displacement restriction

In this chapter, DOPs with previous-solution displacement restriction (PSDR) are investigated. In the investigated DOPs with PSDR, the algorithm needs to find a new solution after an environmental change that is not much different from the previous one. For example, in the aircraft taking-off/landing scheduling problem (Atkin et al., 2008), this type of objective alongside the optimality objective (Nguyen, 2011) can be seen. Moreover, displacement between consecutive solutions can be seen as the switching cost (SC) in many problems (Huang et al., 2017) which needs to be minimized as the second objective. In fact, changing solutions in real-world problems is often costly. Therefore, larger changes have more cost and need more resources such as time and energy. As a result, when the optimization algorithms decision maker needs to choose the next solution after environmental changes, the displacement/SC to the new solution must be considered alongside with its optimality objective's fitness value. Moreover, DOPs with PSDR are dynamic time-linkage problems (Nguyen and Yao, 2009), because choosing a solution for the current environment will change the next environments search space of the displacement/SC objective. In fact, when a solution is chosen for an environment, all the feasible solutions will be evaluated based on their distance from it in terms of displacement/SC.

In this chapter, a new hybrid method based on particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) is proposed for DOPs with PSDR. The proposed algorithm that is denoted PSDR-hPSO is designed based on a multi-swarm PSO that is responsible for finding and tracking peaks based on the optimality objective and a single-swarm PSO (sPSO) whose task is to find the optimum solution according to both optimality and displacement/SC objectives. After each environmental change, a new decision maker chooses a peak according to the fitness values of peaks in the present and some of their characteristics which can be used to anticipate the future displacement/SC value. Then, sPSO uses the location information from the decision maker in order to accelerate the optimization process and improve the performance.

3.1 Problem definition

Here, DOPs defined in Eq. (1.1) and (1.2) are investigated. In this chapter, it is assumed that choosing a solution for each environment is possible and the system is capable of tolerating frequent changes in solutions. Therefore, for DOPs with PSDR in this chapter, the optimization algorithm needs to choose a new solution for each environment based on both optimality and displacement/SC objectives.

Since DOPs with PSDR has two conflicting objectives which need to be optimized concurrently, they can be categorized as dynamic multi-objective optimization problem (DMOOP) (Bui et al., 2005). In DMOOP, algorithms need to find a set of solutions close to the true Pareto-optimal front (POF) for each environment (Farina et al., 2004). A DMOOP with m objectives can be defined as follows:

$$F(\mathbf{x}) = \left\{ f_1(\mathbf{x}, \theta_1^{(t)}), f_2(\mathbf{x}, \theta_2^{(t)}), \cdots, f_m(\mathbf{x}, \theta_m^{(t)}) \right\} \tag{3.1}$$

where F is the objective function, f_i is the i^{th} objective, $\mathbf{x}$ is a design variable vector, θ_i is environmental parameters of i^{th} objective which is changing over time and t is the time index. DMOOPs are classified into different groups based on the POF and Pareto-optimal set (POS) conditions over time (Farina et al., 2004). The considered PSDR in this chapter is classified in the group in which both POF and POS change over time. However, even if the optimization algorithm finds a set of solutions close to POF for each environment, its decision maker needs to choose a solution from them. As a result, for solving PSDR, there is no need to find a set of solutions and the algorithm needs to search for the preferred solution from POS of each environment. One of the easiest ways to solve a DMOOP in this situation is to convert the problem defined by Eq. (3.1) to a weighted-sum optimization problem (Chankong and Haimes, 1983):

$$F_{w_1,w_2,\cdots,w_m}(\mathbf{x}) = w_1 f_1(\mathbf{x}, \theta_1^{(t)}) + w_2 f_2(\mathbf{x}, \theta_2^{(t)}) + \cdots + w_m f_m(\mathbf{x}, \theta_m^{(t)}) \tag{3.2}$$

where w_i is the constant value which is multiplied to the fitness value of the i^{th} objective function. By converting a MDOOP to a weighted-sum optimization problem as Eq. (3.2), by setting the w values, one of the solutions in POS as the optimum solution is chosen for each environment.

As mentioned before, DOPs with PSDR have the time-linkage feature. In dynamic time-linkage optimization problems (DTP), the optimizer should consider both the present and future i.e. it needs to predict the future behavior of the environment based on the chosen solution.

3.2 Proposed hybrid method for PSDR

DOPs with PSDR are counted as dynamic multi-objective and time-linkage problems. Consequently, the proposed algorithm needs to address all of the necessary requirements of DOPs, MOOPs and DTPs.

3.2.1 Addressing dynamic optimization problems' requirements

One way to tackle DOPs is using multi-swarm methods (Nguyen et al., 2012a; Mavrovouniotis et al., 2017). The proposed algorithm is equipped with a multi-swarm optimizer whose responsibility is to locate and track peaks. The multi-swarm PSO proposed in (Yazdani et al., 2013b) (FTmPSO) is used inside the proposed algorithm because it is competitive and easy to understand.

The aim of FTmPSO is to find all peaks and track them after environmental changes. However, due to the lack of information about the number of peaks, a free swarm needs to constantly search for possible uncovered peaks. Once a new optimum is found by a free swarm, it changes to a tracker swarm. The parameter settings of the finder swarm is different from trackers because of their different tasks. Yazdani et al. (2013b) showed that the population size in free-swarm (NP_{free}) should be higher than of the trackers. To test the convergence of a free swarm, the algorithm checks the differences between the $f(\mathbf{g}^{\star}_{\text{free}})$ at the current iteration itr with its value at $itr - k$ and if the difference is less than a threshold, then it is assumed that the free swarm has been converged.

After free swarm convergence detection, its better particles create a new tracker swarm ($NP_{\text{free}} \geq NP_{\text{tracker}}$). When a new tracker swarm is created, the free swarm will be reinitialized immediately in the search space in order to search for another uncovered peak. It is possible that a free swarm converges to a peak already covered by a tracker swarm. Tracking a peak by multiple swarms wastes a considerable amount of computational resource. Therefore, a mutual exclusion principle is enforced to avoid more than one swarm to cover the same peak. To establish the mutual exclusion, the mechanism proposed by Blackwell and Branke (2006) was utilized in (Yazdani et al., 2013b). According to the exclusion mechanism, when Euclidean distances between the global best of the free swarm and a tracker swarm is less than a threshold (r_{excl}), the algorithm assumes that the free swarm has converged to a covered peak. In this situation, the free swarm will be re-initialized. The value of r_{excl} is calculate as follows:

$$r_{\text{excl}} = 0.5\frac{\text{SR}}{\sqrt[D]{\text{m}}}, \tag{3.3}$$

where SR is the range of search space and m is the number of peaks. A similar conflict can also happen to two tracker swarms. This situation happens when a peak is covered by a larger peak. Therefore, its tracker swarm loses its own peak and starts converging to the larger peak's center. A similar situation happens when the free swarm convergence is detected before it enters into the mutual exclusion area of a covered peak. As a result, it becomes a tracker swarm and moves toward the peak's center. This is another case where the exclusion principle is enforced to control the computational overhead. To do so, the tracker swarm with worse global best fitness value $f(\mathbf{g}^{\star})$ will be removed. For determining tracker swarms which are under the exclusion condition, the Euclidean distance between all pairs of tracker swarms' $\mathbf{g}^{\star}$ position is calculated and compared with r_{excl} based on Eq. (6.12).

Another critical challenge of the population-based optimization algorithms in DOPs is diversity loss. FTmPSO is a reaction method (Nguyen et al., 2012a) in which the algorithm increases the diversity of trackers after change detection. When a change is detected, for each tracker swarm, one of the particles is located on the $\mathbf{g}^{*}$ position from the previous environment and other particles are randomized around the $\mathbf{g}^{\star}$ position with the radius of shift severity of the peak by Eq. (6.13):

$$\mathbf{P}_{i,j} = (p \cdot s_i \cdot \mathbf{r}) + \mathbf{g}_i^{\star(t-1),\text{end}}, \tag{3.4}$$

where $\mathbf{P}_{i,j}$ is the position of the jth particle of the ith tracker swarm and $\mathbf{g}_i^{\star(t-1),\text{end}}$ is its global best position at the end of the previous environment, s_i is the shift severity of the peak which is under cover of the ith tracker swarm, $p > 0$ determines the radius which the particles should scatter around the $\mathbf{g}_i^{\star(t-1),\text{end}}$ based on the s_i, and $\mathbf{r}$ is a uniformly distributed random number vector in range $[-1, 1]$. In Eq. (3.4), the $\mathbf{g}^{*}$ from the end of the previous environment is used instead of the previous peak center position.

Therefore, the diversity is introduced to the population of each tracker swarm as much as needed. In addition to (3.4), the velocity of particles in trackers is initialized to:

$$\mathbf{V}_{i,j} = q \cdot s_i \cdot \mathbf{r}, \tag{3.5}$$

where $\mathbf{V}_{i,j}$ is the velocity vector of the jth particle of the ith tracker swarm, and q determines the maximum percentage of s_i by which the velocity components should be chosen.

For addressing outdated memory challenge which happens after each environmental change, the fitness values of all $\mathbf{p}^{\star}$ positions of the free swarm will be re-evaluated after each environmental change. For tracker swarms, after re-diversification, the fitness values of particle positions are evaluated and the $\mathbf{p}^{\star}$ positions are set to particle positions.

For change detection, FTmPSO uses a beacon position. The beacon is evaluated in each iteration and if the calculated fitness value is different from the stored value, then the change is detected. Since detecting a change is a separate issue and in most real-world dynamic environments the occurrence of a change is obvious (e.g., arrival of new order, change in temperature) (Nguyen, 2011), in this thesis, it is assumed that the algorithm will be informed when an environmental change happens. Furthermore, FTmPSO utilizes a resource allocation mechanism and a local optimizer for improving the performance. In this thesis, for simplicity, these two mechanisms will not be used.

FTmPSO in the proposed algorithm has two main responsibilities: 1) tracking peaks; 2) gathering some information about each peak. For the first task, FTmPSO acts in the actual problem space without considering the displacement objective. For the second task, each sub-swarm stores the difference between fitness values of the best found position $\mathbf{g}$ (like *Gbest* in PSO (Kennedy and Eberhart, 1995)) at the end of each successive pair of environments inside its own database. The average of these values indicates the peaks height variance.

The exclusion mechanism (Blackwell and Branke, 2006) used by Yazdani et al. (2013b) does not allow more than one sub-swarm to cover the same peak. In the standard version of this mechanism, the swarm with lower $f(\mathbf{g})$ fitness is re-initialized. In the FTmPSO in this chapter, when the distance between two sub-swarms' $\mathbf{g}$ positions is less than a threshold r_{excl}, the older swarm is kept which has the bigger database and remove the younger one. Additionally, if the younger one's $f(\mathbf{g})$ is better, then its $\mathbf{g}$ information is copied to the older one. In this chapter, r_{excl} is calculated based on the one in (Blackwell et al., 2008) which is as follows:

$$r_{\text{excl}} = excl_{\text{factor}} \times \frac{SR}{TSN^{\frac{1}{D}}}, \tag{3.6}$$

where $excl_{\text{factor}}$ is a positive constant less than 1, SR is the search domain, TSN is the current number of sub-populations and D is the number of dimensions. It is worth mentioning that the original formula in (Blackwell and Branke, 2006) used number of peaks instead of TSN which usually is unknown in real-world problems. This was changed to TSN in (Blackwell et al., 2008).

3.2.2 Addressing multi-objective problems' requirements

Alongside the above mentioned single-objective FTmPSO, there is a sPSO which has one swarm and works on both PSDR objectives. Since a solution have to be picked for each environment, there is no need to find the POS. To handle the multi-objective and find the suitable solution among the POS, the objectives are combined into a weighted sum of objectives (Chankong and Haimes, 1983) as Eq. (3.2). In this chapter, the maximizing optimality objective is used which makes the fitness function of sPSO as follows:

$$F(\mathbf{x}) = f(\mathbf{x}, \theta^{(t)}) - (w \times DC(\mathbf{x}, \mathbf{X}_{t-1})) \tag{3.7}$$

where $f(\mathbf{x}, \theta^{(t)})$ is the optimality fitness function, is the chosen solution for the previous environment, w is the weight of the displacement cost (DC) function. The DC function calculates the Euclidean distance between the previous chosen solution and a new design variable vector as follows (Huang et al., 2017):

$$DC(\mathbf{x}, \mathbf{X}_{t-1}) = \|\mathbf{x} - \mathbf{X}_{t-1}\| \tag{3.8}$$

The w parameter in Eq. (3.7), controls the importance of the displacement cost in the optimization. Therefore, lower values of w result in finding solutions with better optimality and its higher values lead to finding solutions that are closer to $\mathbf{X}_{t-1}$. As a result, with setting of w, a preferred solution can be chosen from the POS. Moreover, the ratio of both objectives can be controlled in problems in which naturally the ratio between the two objectives' fitness values are very large or very small. For example, the fitness values of the first objective can vary between 1,000 and 2,000 while the maximum displacement is less than two.

3.2.3 Addressing dynamic time-linkage problems' requirements

For better performance, the proposed algorithm needs to consider the future environments alongside the current one. In most of the previous works on DTPs (Bosman, 2005; Bosman and Poutré, 2007; Bu et al., 2017), a predictor method like Autoregression was used. However, some of the DTPs can be predictor-deceptive (Nguyen et al., 2012b) and some other can be too random to be predicted with a reasonable error. In this thesis, problems with several peaks whose width, height and location change randomly over time are considered. In such problems, using predictors cannot help the algorithm and even can deteriorate the performance, because this type of problems are too random and there is no pattern in the dynamic. As a result, the prediction involves a high error rate. In these circumstances, if the algorithm considers only the current environment, the performance would be better than considering the future with a high error rate in prediction.

To tackle this challenge, a new decision maker is proposed based on the gathered information by FTmPSO's sub-swarms. After each environmental change, FTmPSO reacts to change in order to update memory, increasing diversity and updating database. Additionally, it sends the calculated height variances to the decision maker alongside the $\mathbf{g}$ information of all sub-swarms. Then, the decision maker chooses one of the peaks as follows:

$$\text{argmax}_{i=1}^{SN}\left(\Big[f(\mathbf{g}_i,\theta^{(t)})-(w\times DC(\mathbf{g}_i,\mathbf{X}_{t-1}))\Big]+\frac{1}{B}\sum_{k=1}^{B}\Big(f(\mathbf{g}_i,\theta^{(t)})-HV_k-(w\times DC(\mathbf{g}_i,\mathbf{g}_k))\Big)\right) \tag{3.9}$$

where CP is the chosen peak, SN is the number of sub-swarms, HV_k is the calculated height variance by the k^{th} sub-swarm, $\mathbf{g}_k$ is the k^{th} sub-swarm's $\mathbf{g}$ position, and B is the number of better peaks based on the $f(\mathbf{g}_k,\theta^{(t)})-HV_k$ i.e. based on the worst expected fitness values for k^{th} sub-swarm in the next environment. In Eq. (3.9), in the first part i.e. $\big[f(\mathbf{g}_k,\theta^{(t)})-(w\times DC(\mathbf{g}_i,\mathbf{X}_{t-1}))\big]$, the decision maker considers the current fitness value of peaks and the displacement cost from $\mathbf{X}_{t-1}$ to it. Therefore, it tries to maximize the combined objective by Eq. (3.7) for the current environment. In the second part, the decision maker tries to take the future of peaks into account by considering average displacement cost from them to the B best peaks.

By Eq. (3.9), it is tried to choose a peak that is closer to other good peaks. If it is needed to move the solution to another peak after environmental changes, a lower displacement cost will be endured. Therefore, the decision maker aims to choose reliable peaks which along with the good current situation, provide a better future combined objective value. After choosing the most reliable peak by Eq. (3.9), the $\mathbf{g}_{\text{CP}}$ is sent to the sPSO.

sPSO initializes its particles around the $\mathbf{g}_{\text{CP}}$ inside a cloud with a radius of r_{cloud}. Therefore, sPSO starts optimizing the combined objective by Eq. (3.7), according to the location determined by the decision maker. The initialization with the controlled diversity by the r_{cloud} decreases the chance of moving particles to further regions such as other peaks. In fact, it is possible to have other regions with better fitness in the

Algorithm 2: `PSDR-hPSO`

```
1  Initialize Finder swarm of FTmPSO;
2  repeat
3      if an environmental change happens then
4          Update memory;
5          Introducing diversity;
6          Update database;
7          Calculating height variance;
8          Choose a peak by Eq. (3.9);
9          Send information to sPSO;
10     Execute an iteration of PSO on FTmPSO's sub-swarms;
11     if the Finder swarm is converged then
12         Create a new tracker swarm;
13         Re-initialize finder swarm;
14     Execute exclusion mechanism for FTmPSO;
15     Update $r_{\text{excl}}$ if number of trackers is changed;
16     Execute an iteration of PSO for sPSO;
17 until stopping criterion is met;
```

current environment, however, the decision maker in Eq. (3.9) tries to choose a peak by taking the future displacement cost into account. As a result, the chosen peak may not be the best peak in the current environment but it is a better option based on current and future considerations. Pseudo code of the PSDR-hPSO is shown in Algorithm 2.

3.3 Experiments

3.3.1 Benchmark problems

The Moving Peaks Benchmark (MPB) (Branke, 1999) is the most popular benchmark in the DOP field. The standard baseline function of MPB is as follows:

$$f_t(\mathbf{x}) = \max_{i=1}^{i=m}\{h_t^i - (w_t^i \cdot \|\mathbf{x} - \mathbf{c}_t^i\|)\}, \tag{3.10}$$

where m is the number of peaks, $\mathbf{x}$ is a solution in the problem space, h_t^i, w_t^i and $\mathbf{c}_t^i$ are the height, width and center of the i^{th} peak in the t^{th} environment, respectively. In the modified version of MPB (mMPB) (Jin et al., 2013; Yazdani et al., 2017) used in this chapter, each peak has its own height and width severities. The height, width and center of a peak change from one environment to the next one as follows:

$$h_i^{(t+1)} = h_i^{(t)} + \alpha_i \cdot N(0,1), \tag{3.11}$$

$$w_i^{(t+1)} = w_i^{(t)} + \beta_i \cdot N(0,1), \tag{3.12}$$

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \tag{3.13}$$

where

$$\mathbf{v}_i^{(t+1)} = s \cdot \frac{(i-\lambda)\cdot \mathcal{R} + \lambda \cdot \mathbf{v}_i^{(t)}}{\|(i-\lambda)\cdot \mathcal{R} + \lambda \cdot \mathbf{v}_i^{(t)}\|}, \tag{3.14}$$

where $N(0,1)$ represents a random number drawn from a Gaussian distribution with mean 0 and variance 1, $\mathcal{R}$ is a uniformly generated random vector $\in [-0.5, 0.5]$, α_i is height severity i^{th} peak, β_i is width severity of i^{th} peak, s is the shift severity and λ is the correlation coefficient. The parameter settings of the mMPBR are shown in Table 3.1. The highlighted values in Table 3.1 are default parameter values of mMPB in this chapter.

TABLE 3.1: Parameter settings of mMPBR (default values are highlighted).

Parameter	Value(s)
Number of peaks, m	10,20,50
Evaluations between changes, f	1000,2500,5000
Shift severity, s	1,2,5
Height severity, α	Randomized in [1,10]
Width severity, β	Randomized in [0.1,1]
Peaks shape	Cone
Correlation coefficient, λ	0
Number of dimensions, D	2,5
Peaks location range, SR	[0,100]
Peak height, h	[30,70]
Peak width, w	[1,12]
Initial height value	50
Initial width value	6
Number of environments	100

3.3.2 Performance indicator

In this chapter, for measuring the performance of the algorithms, Eq. (3.15) is used:

$$\text{AF} = \frac{1}{N-1}\sum_{t=2}^{N}\Big(f(\mathbf{X}_t, \theta^{(t)}) - w \times DC(\mathbf{X}_t, \mathbf{X}_{t-1})\Big) \tag{3.15}$$

where AF is average fitness of chosen solutions for all environments and $\mathbf{X}_t$ is the chosen solution for the t^{th} environment.

3.3.3 Compared algorithms and parameter settings

The proposed algorithm is compared with two TMO algorithms on mPSO. The first one chooses the best solution according to optimality in each environment (it is called it optimality TMO (oTMO)) so it does not consider the displacement objective. The second method uses Eq. (3.7) as fitness function (it is called it combined objective TMO (cTMO)).

PSDR-hPSO, oTMO and cTMO use a simplified version of the FTMPSO (Yazdani et al., 2013b) in which the exploiter particle and sleeping awakening mechanisms are disabled. It is assumed that all algorithms are informed about environmental changes. The parameter setting of them is shown in Table 3.2. sPSO in PSDR-hPSO is working based on (Eberhart and Shi, 2001), c_1=c_2=2.05, χ is 0.729843788 and the population size is 10. For the PSDR-hPSO, the value of B in Eq. (3.9), is set to half of the sub-swarm number. r_{cloud} for initializing sPSO is equal to the shift severity of peaks which is learned by averaging the Euclidean distances between best found position **g** positions at the end of successive environments. All experiments are done with different values of w i.e. 0.5,1 and 2.

Experimental results are obtained by performing 30 independent runs and the best results based on Wilcoxon signed-rank test with significance level of 0.05 are highlighted in each table. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used to compare two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ (Wilcoxon, 1945). It can be used as an alternative to the paired Student's t-test, t-test for matched pairs, or the t-test for dependent samples when the population cannot be assumed to be normally distributed.

TABLE 3.2: Parameter setting of FTmPSO, oTMO and cTMO

Parameter	Value
$C1, C2$	2.05
χ	0.729843788
$Trackers'\ population\ size$	5
$Finder's\ population\ size$	10
$Exclusion\ fatcor$	0.5
P	1
Q	1
$Convergence\ limit$	1
k	10
Stop criterion	Max fitness evaluation number

3.3.4 Experimental results

Table 3.3 shows the obtained results by PSDR-hPSO, oTMO and cTMO on mMPB with different numbers of peaks m=10, 20 and 50 (all other mMPB parameters have default values). The results show that the performance of the PSDR-hPSO is better than that of oTMO and cTMO in all test instances in Table 3.3. By increasing the number of peaks, due to increasing the density of peaks in the landscape, the performance of all methods increased noticeably. Indeed, by increasing peak density, the displacement cost by Eq. (3.8) would be decreased because the average distance between peaks is smaller. On the other hand, when the average distance between peaks is larger, the displacement cost is higher which leads to lower performance.

In all the test instances in Table 3.3, with increasing w, the performance decreases significantly. In fact, by increasing w, the influence of displacements cost in the combined fitness function by Eq. (3.7) increased which leads to larger changes in the problem space made by this fitness function. This issue affects the performance of the cTMO and PSDR-hPSO directly because both of them use Eq. (3.7). Moreover, although oTMO acts independently from the displacement objective, it is affected by the value of w because the performance of algorithms is calculated by Eq. (3.15) that takes the average displacement cost between successive solutions into account. Higher values of w increase the distance between the optimum by the optimality objective and the optimum by the Eq. (3.7) that deteriorates the results of oTMO.

The obtained results by cTMO are better than those of oTMO in Table 3.3. The reason is that cTMO uses Eq. (3.7) as objective function and considers displacement cost in the optimization. Moreover, PSDR-hPSOs performance is better than that of cTMO in all test instances. The first reason is that in PSDR-hPSO, the FTmPSO acts based on the optimality objective. As a result, its tracker-swarms are able to track peaks more easily than cTMOs tracker-swarms which need to tolerate larger changes in peaks especially when the w is larger. In fact, displacement cost can enlarge the relocation distance of peaks after environmental changes.

The second reason which is the most important one is that the PSDR-hPSO's decision maker in Eq. (3.9) considers the future of the search space alongside the current one. As discussed before, PSDRs are classified as problems with the time-linkage feature. Therefore, PSDR-hPSO's performance which considers both present and future is better than that of cTMO which acts based on "optimizing only the present".

Figure 3.1 shows the time-linkage property of the PSDR clearly. In this figure, a MPB with 5 peaks of equal height and width is shown in the sub-figure (I). Other sub-figures are made by Eq. (3.7) with w=1 and when one of the peak centers is chosen as the

TABLE 3.3: Results on test instances with different number of peaks P.

Algorithm	P=10			P=20			P=50		
	w=0.5	w=1	w=2	w=0.5	w=1	w=2	w=0.5	w=1	w=2
oTMO	52.67 (0.97)	50.38 (0.97)	48.48 (1.02)	55.28 (0.67)	51.33 (1.02)	49.55 (1.25)	57.12 (0.79)	51.61 (1.00)	51.59 (0.96)
cTMO	54.74 (0.75)	50.88 (1.04)	49.11 (1.42)	56.47 (0.95)	53.28 (1.16)	50.44 (1.21)	57.57 (0.57)	54.22 (1.08)	51.89 (1.11)
PSDR-hPSO	55.78 (0.76)	51.13 (1.48)	49.75 (2.04)	59.27 (0.45)	55.03 (0.67)	53.98 (0.87)	60.53 (0.56)	55.82 (0.68)	54.31 (0.92)

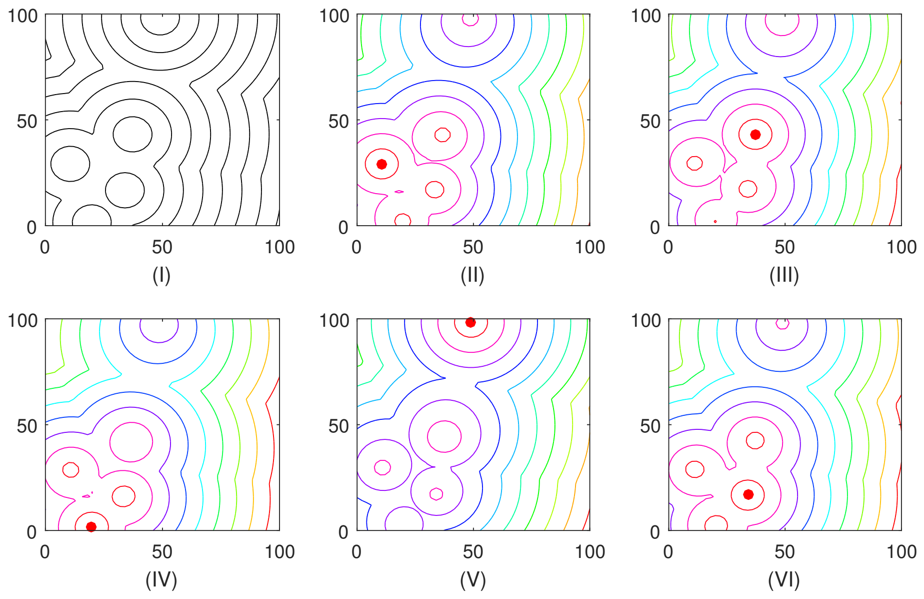


FIGURE 3.1: Time-linkage property of the PSDR by choosing different peak centers as the current solution.

current solution (the chosen peak center is illustrated by a red filled circle). According to Eq. (3.7), when a peak center is chosen as a solution, all the feasible solutions fitness values in the search space are affected by their distance to it. Therefore, choosing a solution has influence on the future environments. It is worth mentioning that PSDR-hPSO will not choose the peak in sub-figure (V) because this peak is far away from other peaks, so the average displacement cost between it and the B best peaks in Eq. (3.9) is high which deteriorates its chance to be chosen. Therefore, the algorithm avoids the high displacement cost in the future.

The results of algorithms on mMPB with different shift severities are reported in Table 3.4. Similar to the result of Table 3.3, PSDR-hPSO outperformed cTMO and oTMO algorithms in all test instances. By increasing shift severities, the performance algorithms is deteriorated. Higher shift severities increase the displacement cost in the

Table 3.4: Results on test instances with different shift severities s.

Algorithm		s=1			s=2			s=5	
	w=0.5	w=1	w=2	w=0.5	w=1	w=2	w=0.5	w=1	w=2
oTMO	55.28 (0.67)	51.33 (1.02)	49.55 (1.25)	54.29 (0.59)	49.34 (1.04)	48.05 (0.95)	53.53 (0.80)	48.42 (1.19)	43.91 (0.96)
cTMO	56.47 (0.95)	53.28 (1.16)	50.44 (1.21)	55.62 (0.87)	51.34 (1.03)	49.18 (1.09)	54.64 (0.82)	50.22 (1.35)	46.31 (1.04)
PSDR-hPSO	59.27 (0.0.45)	55.03 (0.67)	53.98 (0.87)	58.00 (0.42)	54.43 (0.82)	49.96 (1.16)	57.67 (0.56)	51.85 (1.03)	47.97 (1.12)

Table 3.5: Results on test instances with different change frequencies f.

Algorithm		f=1000			f=2500			f=5000	
	w=0.5	w=1	w=2	w=0.5	w=1	w=2	w=0.5	w=1	w=2
oTMO	53.31 (0.75)	49.39 (1.13)	46.51 (1.26)	54.45 (1.43)	50.69 (1.49)	48.16 (1.13)	55.28 (0.67)	51.33 (1.02)	49.55 (1.25)
cTMO	51.66 (1.11)	48.63 (0.92)	46.43 (1.23)	55.13 (1.06)	51.40 (1.39)	49.38 (1.16)	56.47 (0.95)	53.28 (1.16)	50.44 (1.21)
PSDR-hPSO	55.71 (0.57)	52.55 (0.94)	47.14 (1.40)	57.73 (0.70)	54.48 (0.97)	51.92 (1.95)	59.27 (0.45)	55.03 (0.67)	53.98 (0.87)

circumstances in which a peak is chosen for more than one successive environment. Furthermore, peaks relocate with larger steps which makes the tracking task harder for tracker swarms.

Table 3.5 indicates the performance of algorithms on mMPB with different change frequencies. Again, the best results in all problem instances belong to PSDR-hPSO. Lower values of f means higher change frequencies. In such a situation, there is insufficient time for algorithms to do a good search because the environments change more rapidly, which leads to worse performance. In mMPB with f=1000, it exceptionally can be seen the results obtained by oTMO are better than those of cTMO. As mentioned before, the displacement causes an additional step size to the shift severity of peaks when Eq. (3.7) is used as the objective function. Thus, in higher change frequencies and with larger relocating of peaks after each environmental change, cTMO has a harder job for tracking peaks which deteriorates its performance. On the other hand, FTmPSO in PSDR-hPSO uses the optimality objective function and as a result, it is not involved with this issue.

3.4 Summary

In this chapter, DOPs with PSDRs were investigated and a novel multi-objective and time-linkage based method was proposed. The proposed method utilizes a multi-swarm

PSO for tracking peaks in the optimality objective search space. A new decision maker was designed which uses the information gathered by sub-swarms of multi-swarm PSO in order to choose a peak. The information transferred to the decision maker consists of location, fitness value and height variance of peaks. The proposed decision maker chooses a peak based on the current environments peaks fitness values, their worst expected fitness values for the next environment, their distance to a pre-defined number of better peaks and also the future displacement cost for changing solutions. After each environmental change, the information of the chosen peak is sent to a single swarm PSO which works on the combined objectives i.e. optimality and displacement cost and is responsible for finding a solution for each environment. The single swarm PSO initializes its particles around the received peak location from the decision maker. The experimental results showed that the proposed algorithm outperformed other tested methods which only focus on optimizing the present environment.

The proposed method and the related experimental results in this chapter showed that considering future in choosing next solution procedure can improve the performance significantly. However, this context need to be investigated with different methodologies on different testbeds. Although the multi-objective handling mechanism used in this chapter is an easy way, it might not be an efficient way. Considering other ways including non-linear combinations of weighted objectives, and finding POS for each environment should be investigated in future work.

Chapter 4

Robust optimization over time by learning problem space characteristics

As mentioned in Subsection 1.3.1.2, when SC in DOPs is very large, or frequently changing solutions is undesirable, the algorithm needs to keep solutions as long as they remain acceptable. To address such a problem, Yu et al. (2010) proposed an approach for solving DOPs under the above mentioned circumstances: finding solutions that are robust over the course of time. A robust solution is one that is not necessarily the best in the current environment, but that remains acceptable over several environments. A found robust solution can be utilized until its quality degrades to an unacceptable level.

In case the current robust solution becomes unsatisfactory, a new robust solution must be chosen. The process of finding a sequence of robust solutions is referred to as robust optimization over time (ROOT) (Yu et al., 2010; Fu et al., 2013; Jin et al., 2013). For ROOT, the main goal is to minimize the number of times the chosen solution has to be changed because its performance drops below an acceptable level, or to maximize the average number of environments in which a robust solution remains acceptable. Thus, the best case is that the first robust solution remains acceptable for all of the T environments and the worst case is that the number of robust solutions is equal to the number of environments (none of the solutions remained acceptable after even a single environmental change).

In this chapter, a new framework for ROOT is proposed. Its novelty and contribution are as follows. First, this framework uses multi-population methods to track and monitor each peak and learn about their characteristics. Second, in contrast to previous state-of-the-art frameworks which are based on predicting future fitness values of solutions, the proposed framework tries to predict future behaviors of peaks and then selects the next robust solution based on this information. Third, four new strategies to select the next robust solution are proposed. Finally, the proposed framework is empirically evaluated on a wide range of problem settings (different dimensions, change frequencies, shift severities and number of peaks), providing a detailed analysis on the performance of the new framework and demonstrating that it achieves better results than current state-of-the-art ROOT algorithms.

4.1 The proposed framework

According to Section 2.6, almost all current methods on ROOT need to use approximation and prediction methods based on time series (Box et al., 2015). The accuracy of this approach depends on the amount of data available, i.e. past and current fitness values covering the representative regions of the search space. In problems with a large number of dimensions and/or large search space and/or high change frequency, a very large amount of data is required to obtain an accurate approximation. This may be impossible to achieve.

In this chapter, a new framework for ROOT that does not rely on predicted future values of solutions is proposed. Consequently, the proposed framework does not require complicated approximation and prediction methods for predicting solution fitness values. Instead, a multi-population algorithm is responsible for finding peaks, tracking them after environmental changes and gathering information about their behavior. This information will be used to predict the future behavior of peaks. When the current solution becomes unacceptable, the next robust solution will be selected by a decision rule based on information collected by sub-populations such as shift severity or height severity. In this chapter, four such decision rules are proposed.

4.1.1 The multi-population/multi-swarm method

In this section, the necessary characteristics of multi-population (or multi-swarm) methods that can be used inside the proposed ROOT framework are described. It is assumed a multi-population algorithm would continuously try to identify new peaks and track them after an environmental change. Knowledge about the problem such as number of peaks and their shift severities should not be necessary. Additionally, the algorithm should be able to adapt to the number of populations as needed. For example, the proposed multi-population methods in (Yazdani et al., 2013b; Blackwell et al., 2008) have such characteristics.

The other requirement is preventing overcrowding, i.e., each peak should be covered by at most one sub-population. Typically, algorithms use an exclusion mechanism (Blackwell and Branke, 2006) for this purpose. If the distance between the best found positions of two populations drops below some exclusion radius r_{excl}, the population with the worse best found position is re-initialized. One way for calculating r_{excl} without a need to know the number of peaks was described in Eq. (3.6). Note that, in the proposed framework, each sub-population separately records some information about its covered peak. Therefore, the exclusion mechanism for the proposed framework should allow such a record to be transferred from one population to another before the population is re-initialized. If the surviving population is younger (according to the environment number in which it was created), then before the algorithm re-initializes the older one, its database will be transferred to the surviving one.

Another characteristic that a compatible multi-population method should have is being able to track peaks. Therefore, the populations that are responsible to cover and track peaks need to be able to deal with diversity loss (Nguyen et al., 2012a). Nguyen et al. (2012a) grouped methods that deal with diversity into two categories: methods that maintain diversity during the search and the methods that introduce diversity when changes occur. Additionally, to track peaks, populations need to deal with the outdated memory issue that happens after environmental changes. In fact, after changes, the fitness values stored by the algorithm may have changed. This issue can be addressed by re-evaluating all individuals after environmental changes.

Finally, since the algorithms need to be able to react to an environmental change, e.g. by updating memory and calculating and storing some information such as shift severity of peaks, they need to know when a change has occurred. Since detecting a change is a separate issue and in many real-world dynamic environments the occurrence of a change is obvious (e.g., arrival of new order, change in temperature) (Nguyen, 2011), in this chapter, as in all previous algorithms of ROOT (Jin et al., 2013; Fu et al., 2013, 2015; Guo et al., 2014; Huang et al., 2017), it is assumed the information about environmental change events is known and does not need to be detected.

4.1.2 New decision making process for choosing robust solutions

The proposed framework acts based on information gathered by sub-populations tracking peaks. Note that at the t^{th} environment, only sub-populations which were created at the $(t-2)^{\text{th}}$ environment and before that are considered. There are three types of information stored in each sub-population's database:

1. The Euclidean distance between best found positions (such as *Gbest* in PSO) at the end of each successive pair of environments. The average of these distances indicates peaks Shift Severity.

$$S_i = \frac{1}{t - b_i - 1} \times \sum_{k=b_i+1}^{t-1} \left\| \mathbf{g}_i^{(k),\text{end}} - \mathbf{g}_i^{(k-1),\text{end}} \right\|, \tag{4.1}$$

where S_i is the estimated shift severity of the peak covered by the i^{th} sub-population, b_i is the environment in which the i^{th} sub-population has been created, t is the current environment number, and $\mathbf{g}_i^{(k),\text{end}}$ is the best found position of the i^{th} sub-population at the end of the k^{th} environment.

2. The differences between fitness values of its best found positions before and after each environmental change. The average of these values indicates the variance of fitness values of the best found position after environmental changes.

$$FV_i = \frac{1}{t - b_i} \times \sum_{k=b_i}^{t-1} \left| f\left(\mathbf{g}_i^{(k),\text{end}}, \theta^{(k)}\right) - f\left(\mathbf{g}_i^{(k+1),\text{beginning}}, \theta^{(k+1)}\right) \right|, \tag{4.2}$$

where FV_i is the fitness variance of the peak covered by the i^{th} sub-population, $f(\mathbf{g}_i^{(k),\text{end}}, \theta^{(k)})$ is the fitness value of the best found position by the i^{th} sub-population at the end of the k^{th} environment and $f(\mathbf{g}_i^{(k+1),\text{beginning}}, \theta^{(k+1)})$ is the re-evaluated fitness value of this position at the beginning of the next environment.

3. The fitness difference between best found positions at the end of each successive pair of environments. The average of these (called height variance) indicates a peak's height variability.

$$HV_i = \frac{1}{t - b_i - 1} \times \sum_{k=b_i+1}^{t-1} \left| f\left(\mathbf{g}_i^{(k),\text{end}}, \theta^{(k)}\right) - f\left(\mathbf{g}_i^{(k-1),\text{end}}, \theta^{(k-1)}\right) \right|, \tag{4.3}$$

where HV_i is the calculated height variance of the peak covered by the i^{th} sub-population. The database of each sub-population will be updated after each environmental change.

If at t^{th} environment, the fitness value of the current robust solution $\mathbf{r}$ is greater than the threshold V, then it will be kept for at least another environment. Otherwise, after the computational budget (Fu et al., 2015) η which is usually until the end of the current environment, the following procedure will be executed:

Step 1: Pre-selection: Remove from consideration each sub-population i if the current $f(\mathbf{g}_i, \theta^{(t)}) < (FV_i + V)$. FV_i shows how much the fitness value of a position on peak i (covered by i^{th} sub-population) is expected to change after an environmental change.

Thus, if $f(\mathbf{g}_i, \theta^{(t)}) < (FV_i + V)$, in the next environment $f(\mathbf{g}_i, \theta^{(t+1)})$ will likely be below the threshold so this position is not considered a robust solution.

For the remaining candidates $\mathbf{g}$, the proposed framework executes the second step for choosing one of the candidates' $\mathbf{g}$ as the next robust solution. If there is no candidate peak, then the algorithm chooses the $\mathbf{g}$ with the highest fitness value.

Step 2: Four different strategies for choosing the next robust solution (NRS) are proposed as follows:

- The $\mathbf{g}$ with the highest fitness value minus its FV is chosen.

$$\text{NRS} = \text{argmax}_{i=1}^{e} \left(f(\mathbf{g}_i, \theta^{(t)}) - FV_i \right), \tag{4.4}$$

  where e is the number of candidate $\mathbf{g}$ remaining from the first step.

- The $\mathbf{g}$ with the lowest calculated shift severity S by Eq. (4.1) is chosen.

$$\text{NRS} = \text{argmin}_{i=1}^{e}(S_i), \tag{4.5}$$

- The $\mathbf{g}$ with the lowest height variance calculated by Eq. (4.3) is chosen.

$$\text{NRS} = \text{argmin}_{i=1}^{e}(HV_i), \tag{4.6}$$

- The $\mathbf{g}$ with the lowest value obtained by Eq. (4.7) is chosen.

$$\text{NRS} = \text{argmin}_{i=1}^{e} \left(\frac{S_i}{S_{\max}} + \frac{HV_i}{HV_{\max}} \right), \tag{4.7}$$

In the 4th strategy, both height variance HV and shift severity S are used. The values of HV and S of each candidate peak are divided by their maximum values ($S_{\max}$ and $HV_{\max}$) to be normalized in the range $(0, 1)$. The proposed framework checks the acceptability of the current robust solution. If it is not acceptable, it will execute steps 1 and 2 above to choose the next robust solution. If there is no option, the best $\mathbf{g}$ is chosen as NRS. The pseudo code of the proposed framework is shown in Algorithm 3.

4.2 Experiments and analysis

4.2.1 Performance indicators

The most important goal of ROOT i.e. the survival time is considered here. The performance indicator in Eq. (2.12) will be used for the survival time definition in Eq. (2.10).

Algorithm 3: `ROOT framework equipped with a multi-population method`

```
1  Initialize multi-population method;
2  repeat
3      if an environmental change is happened then
4          forall sub-population do
5              Update Database;
6              Calculate $S$, $FV$ and $HV$ by Eq. (4.1), Eq. (4.2) and Eq. (4.3);
7              Update Memory;
8              Other actions for the embedded multi-population method based on its procedure
                 (such as introducing diversity);
9      if computational budget $\eta$ is finished then
10         if the robust solution is not acceptable then
11             Identify candidate $\mathbf{g}$ by Step 1 in Section 4.1.2;
12             Choose one of the candidates $\mathbf{g}$ based on a strategy in Section 4.1.2;
13     Execute an iteration of the multi-population method including finding and tracking peaks;
14     Create or remove sub-populations if needed (based on the procedure of the
         multi-population method);
15     forall pair of sub-populations $i$ and $j$ do
16         if $\|\mathbf{g}_i - \mathbf{g}_j\| < r_{\text{excl}}$ then
17             if $f(\mathbf{g}, \theta^{(t)})$ value of the younger one is better then
18                 Copy the older ones database to the newer one;
19             Keep the sub-population with better $f(\mathbf{g}, \theta^{(t)})$ and remove or the other one;
20     Update $r_{\text{excl}}$ by Eq. (3.6);
21 until stopping criterion is met;
```

Furthermore, the performance indicator in Eq. (2.13) is used to show the average fitness value of robust solutions when the proposed methods are compared with the state-of-the-art ROOT algorithms.

4.2.2 Benchmark functions

Moving peaks benchmark (MPB) (Branke, 1999) is the most popular benchmark function in the DOP field. In its standard form, all peaks are behaving identically, so no solution is more robust than another. This is why in ROOT researchers used various modified versions (Jin et al., 2013; Fu et al., 2013, 2015; Guo et al., 2014; Huang et al., 2017; Yazdani et al., 2017).

(Jin et al., 2013) used three different benchmark generators namely the modified MPB with different height and width severities for each peak; the modified dynamic rotation generator (Li et al., 2009) with different height and width severities for each peak; and finally the modified dynamic composition benchmark generator (Li et al., 2009) with only different height severity for each peak. Each of these three benchmark generators was used with three different numbers of dimensions which resulted in nine test instances in total. Fu et al. (2013) and Yazdani et al. (2017) used a modified version of MPB with different height and width severities. One problem instance of this version was used for testing the algorithm on a 2-dimensional search space. Fu et al. (2015) proposed two different benchmark problems, one specifically designed for maximizing survival time and another for maximizing average fitness. These two benchmarks used two different modified versions of the baseline fitness function of MPB. Furthermore, rotation rather than translation was used to move peaks after environmental changes. The authors used six different dynamics (Li et al., 2009) on their two benchmarks. Huang et al. (2017) used a modified MPB with different height and width severities for peaks. For changing heights and widths of peaks, the benchmark used three different dynamics: small step, random and recurrent (Li et al., 2009), but they used the standard peak center relocation also used in the standard MPB (Branke, 1999).

In this chapter, and similar to ROOT papers in (Jin et al., 2013; Fu et al., 2013; Guo et al., 2014; Huang et al., 2017; Yazdani et al., 2017), the standard baseline function of MPB as shown in Eq. (3.10) is used. In the modified version of MPB for ROOT (mMPBR) used in this chapter, each peak has its own height and width severity. This is similar to the benchmarks in previous ROOT papers (Jin et al., 2013; Fu et al., 2013; Guo et al., 2014; Huang et al., 2017; Yazdani et al., 2017). Additionally, different shift severities for different peaks are used, although in the experiments also the effect of having the same shift severity for all peaks is investigated. The reason for having different height, width and shift severities for each peak is to have different levels of robustness among them. The height and width are calculated by Eq. (3.11) and Eq. (3.12). The center of a peak changes from one environment to the next as follows:

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \tag{4.8}$$

where

$$\mathbf{v}_i^{(t+1)} = s_i \cdot \frac{(1-\lambda)\cdot\mathcal{R} + \lambda\cdot\mathbf{v}_i^{(t)}}{\left\|(1-\lambda)\cdot\mathcal{R} + \lambda\cdot\mathbf{v}_i^{(t)}\right\|}, \tag{4.9}$$

where s_i is the shift severity of the i^{th} peak, $\mathcal{R}$ is a uniformly generated random vector $\in [-0.5, 0.5]$ and λ is the correlation coefficient.

The parameter settings of the mMPBR are shown in Table 4.1. The highlighted values in Table 4.1 are default parameter values of mMPBR which builds the default scenario of the benchmark in this chapter. In the experiments, different numbers of peaks, change frequencies, dimensions and shift severities are used in order to test the sensitivity of the proposed algorithm. For investigating the impact of different parameter settings of mMPBR on the algorithms' performance, most of the default parameter settings are kept and 1 or 2 parameters are changed to build each experiment.

Table 4.1: Parameter settings of mMPBR (default values are highlighted)

Parameter	Value(s)
Number of peaks, m	2,5,10,20,30,50,100,200
Evaluations between changes, f	1000,2500,5000
Shift severity, s	1,5,randomized in [0.5,1],[0.5,3],[0.5,5]
Height severity, α	Randomized in [1,15]
Width severity, β	Randomized in [0.1,1.5]
Peaks shape	Cone
Correlation coefficient, λ	0
Number of dimensions, D	2,**5**,10
Peak location range, SR	[-50,50]
Peak height range	[30,70]
Peak width range	[1,12]
Initial height value	50
Initial width value	6
Number of environments	100

4.2.3 Algorithms and parameter settings

In the experiments, FTmPSO proposed by Yazdani et al. (2013b) is used inside the proposed framework as the multi-swarm method. There are three major reasons for this choice. First, it is very simple, which makes it easy to analyze the impact of the framework on performance. Second, it is a competitive TMO algorithm. Third, with minimal modifications, this method is compatible with the framework according to Section 4.1.1: (a) it uses Eq. (3.6) for determining the exclusion radius r_{excl}; (b) its exclusion mechanism allows the transfer of peak information from one swarm to another; (c) it uses the learned shift severity by Eq. (4.1) instead of the true shift that was used in the original paper. Additionally, the exploiter particle and awakening-sleeping mechanisms proposed in its original paper are not used here. The reason is that these two mechanisms improve the exploitation on the best peak which is not useful in ROOT. Readers are referred to (Yazdani et al., 2013b) for more details of this multi-swarm algorithm. Integrated in the framework, the algorithm has four versions depending on the chosen strategies (Section 4.1.2). The four versions are RFTmPSO-s1 to RFTmPSO-s4, based on strategies 1 to 4, respectively.

The parameter setting of FTmPSO inside the proposed framework is shown in Table 4.2. Since the task of the multi-population methods in the proposed framework is similar to their original purpose (TMO), parameter settings suggested in the original paper can be used here as well. A sensitivity analysis on RFTmPSO is provided in in the following to illustrate the effect of different FTmPSO parameter settings on the ROOT performance. Based on this analysis, the parameter settings in Table 4.2 have been chosen.

TABLE 4.2: Parameter settings of FTmPSO

Parameter	Value(s)
C_1, C_2	2.05
χ	0.729843788
Tracker-swarm's Population Size	5
$excl_{\text{factor}}$	0.1
r_{excl}	calculated by (3.6)
Finder-swarm's Population Size	10
P	1
Q	1
$Conv-limit$	1
k	10
Stop criterion	Max fitness evaluation number

For sensitivity analysis, the effect of different parameter settings of FTmPSO (Yazdani et al., 2013b) are investigated which is the multi-swarm method embedded in the proposed ROOT framework. To test the sensitivity to a particular parameter, this parameter is changed while keeping all other parameters as specified in Table 4.2. Moreover, the effect of different population sizes on the performance of the ROOT-PV method is investigated. Experiments are done on mMPBR with its default parameter setting reported in Table 4.1. All experimental results are obtained by performing 30 independent runs. Best results based on Wilcoxon signed-rank test with Holm-Bonferroni p-value correction, $\alpha = 0.05$ are highlighted in each table.

The first set of experiments examines the effect of the tracker-swarms population size on the obtained survival time by all four versions of RFTmPSO which is reported in Table 4.3. Overall, results demonstrate that five-particle tracker-swarms are best. According to Section 4.1.1, the multi-swarm algorithm in the proposed ROOT framework needs to track multiple optima (TMO), which is similar to its original purpose. If the multi-swarm method tracks peaks properly, more accurate information can be provided for the phase of selecting more robust solutions.

Table 4.4 illustrates the obtained results from algorithms with different numbers of particles in the finder-swarm (a sub-swarm in FTmPSO that is responsible for finding uncovered peaks). As it can be observed, the best performance overall is obtained when the finder-swarms population size is 10. Lower values result in decreasing ability of this sub-swarm to find uncovered peaks. On the other hand, a higher population size of the finder-swarm results in a waste of computational resources (fitness evaluations). A deeper analysis using visual plots indicated that a larger finder-swarm leads to a

Table 4.3: The obtained average survival time (and standard error) from the RFTmPSO algorithms with different sub-swarm's population size (TPS) on the default scenario of mMPBR.

V	TPS	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
	3	5.12(0.49)	5.25(0.61)	4.54(0.42)	5.82(0.47)
	4	5.53(0.35)	5.56(0.62)	4.53(0.36)	6.16(0.68)
40	5	5.48(0.60)	5.60(0.82)	4.89(0.81)	6.14(0.85)
	7	5.17(0.73)	5.54(0.86)	4.76(0.76)	6.19(0.88)
	10	4.95(0.37)	5.02(0.67)	4.52(0.62)	5.16(1.28)
	3	3.61(0.43)	3.65(0.46)	3.33(0.39)	3.84(0.33)
	4	3.90(0.32)	3.59(0.31)	3.52(0.31)	4.05(0.39)
50	5	3.90(0.34)	3.63(0.33)	3.44(0.34)	4.22(0.41)
	7	3.81(0.40)	3.66(0.50)	3.36(0.60)	4.23(0.49)
	10	3.34(0.32)	3.41(0.40)	3.26(0.42)	4.17(0.78)
	3	2.33(0.24)	2.26(0.24)	2.29(0.20)	2.71(0.20)
	4	2.36(0.20)	2.12(0.15)	2.36(0.23)	2.77(0.26)
50	5	2.55(0.21)	2.43(0.26)	2.51(0.28)	2.77(0.31)
	7	2.62(0.40)	2.21(0.41)	2.50(0.44)	2.83(0.24)
	10	2.19(0.27)	2.42(0.26)	2.14(0.28)	2.66(0.30)

convergence to better peaks, due to an increase in exploration ability. Consequently, smaller peaks may not be detected until they become larger which leads to a decrease in the accuracy of the gathered information by FTmPSO.

Table 4.4: The average survival time (and standard error) obtained from the RFTmPSO algorithms with different finder-swarms population size (FPS) on the default scenario of mMPBR.

V	FPS	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
	5	4.83(0.44)	4.95(0.36)	4.21(0.33)	5.28(0.36)
	7	5.01(0.50)	5.35(0.55)	4.71(0.49)	5.90(0.48)
40	10	5.48(0.60)	5.60(0.82)	4.89(0.81)	6.14(0.85)
	12	5.39(0.44)	5.30(0.39)	4.59(0.35)	5.82(0.55)
	15	5.26(0.60)	5.18(0.59)	4.47(0.42)	5.45(0.47)
	5	3.08(0.28)	3.16(0.24)	3.13(0.31)	3.64(0.85)
	7	3.52(0.40)	3.49(0.40)	3.40(0.35)	4.29(0.37)
45	10	3.90(0.34)	3.63(0.33)	3.44(0.34)	4.22(0.41)
	12	3.89(0.30)	3.62(0.27)	3.34(0.31)	4.04(0.30)
	15	3.34(0.56)	3.40(0.35)	3.29(0.31)	3.90(0.37)
	5	2.01(0.19)	1.97(0.18)	2.08(0.19)	2.16(0.41)
	7	2.63(0.34)	2.33(0.32)	2.65(0.30)	2.56(0.28)
50	10	2.55(0.21)	2.43(0.26)	2.51(0.28)	2.77(0.31)
	12	2.45(0.15)	2.47(0.20)	2.46(0.18)	2.59(0.24)
	15	2.25(0.47)	3.10(0.47)	2.58(0.32)	2.57(0.28)

Table 4.5 shows the effect of different values of the *Conv-limit*, parameter which is

used for determining finder-swarms convergence. According to the results presented in Table 4.5, decreasing the values of *Conv-limit* leads to decreasing the performance of algorithms because the finder-swarms convergence condition will not be met in an appropriate time. Therefore, it will take more time for the finder-swarm to create a tracker-swarm on the peak and continue its search for finding other possible uncovered peaks. By increasing the value of *Conv-limit* up to 1, the algorithms' efficiency increases. However, increasing its value beyond 1 will deteriorate performance, probably because a high value of *Conv-limit* results the finder-swarm being considered converged very early, which leads to the creation of unnecessary tracker-swarms, wasting computational resources.

Table 4.5: The average survival time (and standard error) obtained from the RFTmPSO algorithms with different *Conv-limit* (CL) on the default scenario of mMPBR.

V	CL	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
40	0.1	4.28(0.51)	4.84(0.63)	4.47(0.72)	4.88(0.71)
	0.5	4.53(0.45)	5.19(0.47)	4.90(0.44)	5.38(0.47)
	1	4.89(0.81)	5.60(0.82)	5.48(0.60)	6.14(0.85)
	2	4.85(0.52)	5.63(0.71)	5.42(0.53)	5.80(0.69)
	5	4.51(0.54)	5.23(0.50)	5.06(0.77)	5.27(0.57)
45	0.1	3.18(0.29)	3.13(0.37)	3.25(0.42)	3.80(0.43)
	0.5	3.23(0.32)	3.39(0.26)	3.51(0.31)	4.04(0.30)
	1	3.44(0.34)	3.63(0.33)	3.90(0.34)	4.22(0.41)
	2	3.24(0.41)	3.66(0.46)	3.55(0.38)	3.98(0.53)
	5	3.01(0.46)	3.06 (0.48)	3.23(0.61)	3.78(0.52)
50	0.1	2.24(0.24)	2.21(0.26)	2.09(0.27)	2.32(0.29)
	0.5	2.38(0.21)	2.43(0.20)	2.40(0.27)	2.81(0.24)
	1	2.51(0.28)	2.43(0.26)	2.55(0.21)	2.77(0.31)
	2	2.32(0.26)	2.40(0.30)	2.51(0.30)	2.75(0.30)
	5	2.26(0.33)	2.15(0.32)	1.98(0.40)	2.21(0.43)

Another parameter involved in finder-swarm convergence determination is K. The effect of its different values on the algorithms' performance is shown in Table 4.6. Similar to *Conv-limit*, lower values of K result in creating more unnecessary tracker-swarms whereas higher values delay the finder-swarm convergence determination. According to Table 4.6, best performance is generally obtained with $K = 10$.

P and Q control the diversity introducing of tracker-swarms after environmental changes. On the one hand, lower values of P and Q result in a lower initial diversity of tracker-swarms at the beginning of each environment which leads to a decrease in their tracking ability. On the other hand, higher values of these parameters cause over-diversification of tracker-swarms which leads to increasing the possibility of tracker-swarms migrating to other peaks when peaks are very close to each other. Additionally, over-diversification

Table 4.6: The average survival time (and standard error) obtained from the RFTmPSO algorithms with different K on the default scenario of mMPBR.

V	K	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
	5	4.71(4.06)	5.24(0.68)	5.29(0.84)	6.00(0.69)
	7	4.83(0.37)	5.40(0.45)	5.32(0.41)	6.07(0.50)
40	10	4.89(0.81)	5.60(0.82)	5.48(0.60)	6.14(0.85)
	12	4.90(0.47)	5.31(0.47)	5.53(0.54)	5.90(0.54)
	15	4.71(0.45)	5.28(0.52)	5.04(0.40)	5.58(0.66)
	5	3.06(0.50)	3.35(0.55)	3.31(0.34)	3.95(0.60)
	7	3.16(0.27)	3.65(0.29)	3.52(0.27)	3.90(0.41)
45	10	3.44(0.34)	3.63(0.33)	3.90(0.34)	4.22(0.41)
	12	3.26(0.44)	3.45(0.43)	4.05(0.48)	4.03(0.40)
	15	3.40(0.32)	3.34(0.27)	3.62(0.23)	3.84(0.51)
	5	2.32(0.36)	2.33(0.36)	2.37(0.34)	2.75(0.32)
	7	2.20(0.20)	2.54(0.21)	2.42(0.21)	2.81(0.20)
50	10	2.51(0.28)	2.43(0.26)	2.55(0.21)	2.77(0.31)
	12	2.29(0.20)	2.59(0.22)	2.51(0.22)	2.75(0.27)
	15	1.95(0.14)	2.12(0.16)	1.99(0.17)	2.66(0.30)

decreases the exploitation ability. The results of using different values of P and Q are reported in Tables 4.7 and 4.8. Note that, although in (Yazdani et al., 2013b) P and Q work based on the shift severity of peaks was available for FTmPSO as an initial knowledge, here algorithms have to learn about peaks shift severities by themselves using (4.1).

Table 4.7: The average survival time (and standard error) obtained from the RFTmPSO algorithms with different Q on the default scenario of mMPBR.

V	Q	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
	0.5	5.20(0.37)	5.55(0.46)	5.15(0.56)	6.00(0.69)
	0.75	5.39(0.51)	5.30(0.60)	5.10(0.74)	6.17(0.68)
40	1	5.48(0.60)	5.60(0.82)	4.89(0.81)	6.14(0.85)
	1.5	5.15(0.40)	5.22(0.52)	4.93(0.70)	5.67(0.75)
	2	5.02(0.53)	4.97(0.83)	4.59(0.73)	5.14(0.81)
	0.5	3.65(0.32)	3.48(0.34)	3.21(0.33)	4.14(0.34)
	0.75	3.71(0.40)	3.60(0.39)	3.36(0.41)	4.27(0.40)
45	1	3.90(0.34)	3.63(0.33)	3.44(0.34)	4.22(0.41)
	1.5	3.91(0.25)	3.68(0.45)	3.18(0.39)	4.31(0.46)
	2	3.45(0.44)	3.25(0.44)	3.19(0.53)	4.12(0.48)
	0.5	2.36(0.26)	2.29(0.24)	2.25(0.27)	2.38(0.25)
	0.75	2.56(0.21)	2.35(0.22)	2.56(0.22)	2.72(0.23)
50	1	2.55(0.21)	2.43(0.26)	2.51(0.28)	2.77(0.31)
	1.5	2.58(0.26)	2.39(0.20)	2.49(0.26)	2.63(0.26)
	2	2.28(0.26)	2.23(0.27)	2.20(0.28)	2.26(0.26)

TABLE 4.8: The average survival time (and standard error) obtained from the RFTmPSO algorithms with different P on the default scenario of mMPBR.

V	P	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
	0.5	5.24(0.35)	5.62(0.51)	4.58(0.56)	5.91(0.48)
	0.75	5.40(0.55)	5.46(0.56)	4.70(0.60)	5.61(0.57)
40	1	5.48(0.60)	5.60(0.82)	4.89(0.81)	6.14(0.85)
	2	5.49(0.46)	5.42(0.60)	5.14(0.66)	6.08(0.92)
	5	5.28(0.70)	5.18(0.86)	4.64(0.78)	5.87(0.54)
	0.5	3.68(0.30)	3.68(0.37)	3.22(0.35)	3.85(0.39)
	0.75	3.71(0.36)	3.53(0.37)	3.29(0.28)	4.15(0.34)
45	1	3.90(0.34)	3.63(0.33)	3.44(0.34)	4.22(0.41)
	2	3.88(0.38)	3.69(0.38)	3.31(0.55)	4.14(0.34)
	5	3.71(0.41)	3.61(0.45)	3.20(0.67)	3.79(0.54)
	0.5	2.33(0.29)	2.34(0.31)	2.56(0.26)	2.78(0.24)
	0.75	2.36(0.22)	2.49(0.27)	2.19(0.22)	2.39(0.26)
50	1	2.55(0.21)	2.43(0.26)	2.51(0.28)	2.77(0.31)
	2	2.38(0.19)	2.44(0.31)	2.50(0.35)	2.82(0.26)
	5	2.25(0.27)	2.32(0.33)	2.40(0.33)	2.75(0.28)

The last investigated parameter is $excl_{\text{factor}}$ which is used in (3.6). According to Table 4.9, for embedded multi-swarm methods in the proposed framework, this threshold needs to be lower (i.e. 0.1) in comparison with its value in the original references (Blackwell and Branke, 2006; Blackwell et al., 2008) (i.e. 0.5), because it is desirable to avoid losing information about a peak only because it moves close to another peak. Therefore, sub-swarms need to be closer to be involved in exclusion. In fact, higher values of $excl_{\text{factor}}$ increase the possibility of involving sub-swarms whose under covered peaks are close to each other in exclusion condition. Consequently, the algorithm may lose valuable information about a peak by removing a sub-swarm by the exclusion mechanism.

4.3 Experimental results

This chapter's experimental results are reported in two parts. In the first part, the performance of the proposed framework with four strategies from Section 4.1.2 is investigated on several problem instances with different characteristics. The second part compares the proposed methods embedded into different multi-swarm methods with the state-of-the-art ROOT methods and compares their behaviors on different problem instances.

All experimental results are obtained by performing 30 independent runs. To test the statistical significance of the reported results, a multiple comparison test is performed and the best results based on Wilcoxon signed-rank test with Holm-Bonferroni p-value

Table 4.9: The average survival time (and standard error) obtained from the RFTmPSO algorithms with different $excl_{\text{factor}}$ (EF) on the default scenario of mMPBR.

V	EF	RFTmPSO-s1	RFTmPSO-s2	RFTmPSO-s3	RFTmPSO-s4
	0.05	5.13(0.55)	5.21(0.68)	4.32(0.71)	5.52(0.72)
	0.1	**5.48(0.60)**	**5.60(0.82)**	**4.89(0.81)**	**6.14(0.85)**
40	0.25	5.23(0.48)	5.26(0.65)	4.58(0.60)	5.70(0.54)
	0.5	**5.47(0.54)**	5.35(0.79)	**4.83(0.68)**	5.86(0.79)
	1	4.86(1.11)	5.08(0.71)	4.39(1.07)	4.47(1.07)
	0.05	3.63(0.40)	**3.67(0.51)**	3.31(0.57)	**4.27(0.51)**
	0.1	**3.90(0.34)**	**3.63(0.33)**	3.44(0.34)	**4.22(0.41)**
45	0.25	3.52(0.31)	**3.67(0.36)**	3.28(0.38)	3.91(0.39)
	0.5	3.55(0.42)	**3.70(0.51)**	**3.62(0.46)**	4.07(0.47)
	1	3.31(0.69)	3.35(0.71)	3.13(0.60)	3.05(0.60)
	0.05	2.34(0.28)	**2.47(0.29)**	2.24(0.37)	2.20(0.31)
	0.1	**2.55(0.21)**	**2.43(0.26)**	**2.51(0.28)**	**2.77(0.31)**
50	0.25	2.23(0.29)	2.31(0.30)	**2.54(0.23)**	2.32(0.27)
	0.5	**2.55(0.24)**	**2.43(0.32)**	2.37(0.20)	2.00(0.32)
	1	2.02(0.51)	2.04(0.55)	1.92(0.54)	1.93(0.54)

correction and $\alpha = 0.05$ are highlighted in each table. If there is more than one highlighted result, it means that they are not significantly different.

4.3.1 Analyzing the proposed framework on problems with different characteristics

Tables 4.10 and 4.11 show the average survival time of RFTmPSO with four different strategies on mMPBR with different numbers of peaks. All other mMPBR parameters are set to default values. The worst results are obtained in mMPBR with 2 peaks. All versions of RFTmPSO perform identically in this instance because the number of options for choosing the next robust solution is limited. Also, when the number of peaks is low, there are large areas of low fitness because there are few peaks to cover these areas. As a result, the average solution quality is lower, and robust solutions can lose their quality more quickly. By increasing the number of peaks, the average survival time increases because peaks are likely to overlap and support robust solutions. Increasing the number of peaks also increases the performance difference between different versions of RFTmPSO because there are more peaks with different characteristics and RFTmPSO has more options to choose the best of them based on the robust solution selection strategies. The best results are obtained on mMPBR with 50 peaks, but when the number of peaks is increased to 100 and 200, performance decreases. The reason is that the algorithm can no longer cover all peaks because of their large number. Furthermore, the algorithm cannot perform a good local search to track peaks because the number of tracker swarms is large.

TABLE 4.10: Average survival time (and standard error) on mMPBR with peak number $m = \{2, 5, 10, 20\}$, $f = 2500$, s randomized $\in [0.5, 3]$ and $D = 5$.

V	Algorithm	Peak Number, m			
		2	5	10	20
40	RFTmPSO-s1	3.08(0.65)	3.98(0.39)	4.80(0.62)	5.48(0.60)
	RFTmPSO-s2	3.41(0.80)	4.13(0.46)	4.53(0.67)	5.60(0.82)
	RFTmPSO-s3	3.36(0.80)	3.81(0.45)	4.21(0.61)	4.89(0.81)
	RFTmPSO-s4	3.36(0.81)	3.86(0.45)	4.67(0.68)	6.14(0.85)
40	RFTmPSO-s1	2.19(0.52)	2.55(0.31)	3.34(0.40)	3.90(0.34)
	RFTmPSO-s2	2.30(0.58)	2.59(0.30)	3.16(0.38)	3.63(0.33)
	RFTmPSO-s3	2.29(0.58)	2.40(0.29)	3.00(0.37=)	3.44(0.34)
	RFTmPSO-s4	2.30(0.55)	2.48(0.28)	3.23(0.38)	4.22(0.41)
40	RFTmPSO-s1	1.33(0.39)	1.51(0.19)	2.40(0.35)	2.55(0.21)
	RFTmPSO-s2	1.35(0.39)	1.51(0.18)	2.10(0.25)	2.43(0.26)
	RFTmPSO-s3	1.34(0.39)	1.46(0.17)	2.02(0.24)	2.51(0.28)
	RFTmPSO-s4	1.34(0.39)	1.51(0.18)	2.18(0.27)	2.77(0.31)

TABLE 4.11: Average survival time (and standard error) on mMPBR with different peak number $m = \{30, 50, 100, 200\}$, $f = 2500$, s randomized $\in [0.5, 3]$ and $D = 5$.

V	Algorithm	Peak Number, m			
		30	50	100	200
40	RFTmPSO-s1	6.46(0.75)	7.41(0.83)	6.19(0.47)	6.26(0.55)
	RFTmPSO-s2	8.11(1.18)	7.84(0.99)	5.73(0.47)	6.11(0.60)
	RFTmPSO-s3	6.65(1.08)	7.01(0.96)	5.87(0.71)	5.80(0.61)
	RFTmPSO-s4	8.21(1.16)	8.23(0.98)	6.51(0.46)	6.89(0.62)
40	RFTmPSO-s1	4.31(0.39)	5.20(0.47)	4.98(0.60)	4.90(0.43)
	RFTmPSO-s2	5.23(0.73)	5.73(0.63)	5.05(0.55)	5.07(0.34)
	RFTmPSO-s3	4.77(0.57)	5.95(0.76)	4.36(0.55)	4.50(0.39)
	RFTmPSO-s4	5.31(0.61)	6.16(0.62)	5.43(0.58)	5.44(0.37)
40	RFTmPSO-s1	3.26(0.46)	3.65(0.40)	3.17(0.25)	3.31(0. 33)
	RFTmPSO-s2	3.19(0.34)	3.94(0.46)	3.27(0.40)	3.33(0.30)
	RFTmPSO-s3	2.90(0.32)	3.93(0.55)	3.22(0.31)	3.20(0.32)
	RFTmPSO-s4	3.67(0.56)	4.10(0.51)	3.39(0.37)	3.57(0.29)

In problems with a higher number of peaks such as 100 and 200, the density of peaks is high. As a result, it is highly likely that some peaks are covered by higher peaks. In such a case, the tracker swarm will lose its covered peak, and hence their associated information, leading to a worse performance. However, the multi-population algorithm would search for possible uncovered peaks all the time (Section 4.1.1). Therefore, when a peak hidden by another peak re-appears, the multi-swarm algorithm would be able to find it and start gathering information about it again. Although algorithm performances are worse for 100 and 200 peaks in comparison with 50 peaks, the average survival time values are still very good. This demonstrates the ability of the proposed methods in dealing with a higher number of peaks.

When increasing the threshold V, the performance of RFTmPSO decreases because the survival time of solutions in the problem space decreases. No algorithm can do anything about this. Also, the performance of RFTmPSO versions is closer when V is high because the number of options for choosing the next robust solution decreases.

Tables 4.10 and 4.11 show that RFTmPSO-s2 performs better than RFTmPSO-s3. This illustrates that the effect of shift severity on the life cycle length of robust solutions is more important than the effect of height variance. However, when both parameters are considered (RFTmPSO-s4), as in (4.7), the performance is improved. RFTmPSO-s4 performs best overall. RFTmPSO-s1 could rarely outperform other versions of RFTmPSO which means that considering fitness variance in (4.4) for choosing the next robust solution is not the best way.

Tables 4.12 and 4.13 show the obtained average survival time for RFTmPSO for mMPBR with different numbers of peaks, different numbers of dimensions and default values for other parameters. The proposed RFTmPSO algorithms can find robust solutions in high numbers of dimensions and peaks. When the peak number increases to 50, the performance improves regardless of the number of dimensions. Increasing the number of peaks further to 100 or 200 leads to a slight deterioration of results. The average survival time is also lower because the problems become more complex for algorithms. Overall, RFTmPSO-s4 maintains its superiority.

Tables 4.14 and 4.15 show the results of testing RFTmPSO on mMPBR with different shift severities in 5 and 10 dimensions, with default values for other parameters. As expected, when shift severity increases, the average survival time decreases because tracking peaks with higher shift severities is harder for tracker swarms and their ability of gathering information decreases. More importantly, the maximal possible survival time decreases due to the increased shift severities. Also, robust solutions become unacceptable more quickly because peaks move with larger steps. The worst results are observed when all peaks have the same high shift severity of 5. When all peaks have the same severity, information on a peaks shift severity is not useful. Thus, RFTmPSO-s4 does not perform better than other algorithms because it relies on learning the difference of peak shift severities. On the other hand, RFTmPSO-s3, which does not use information about shift severity, has the best results on these problems.

TABLE 4.12: Average survival time (and standard error) on mMPBR with different $m = \{5, 10\}$, different D, $f = 2500$ and s randomized in [0.5,3].

V	Algorithm	$m=5$			$m=10$		
		D=2	D=5	D=10	D=2	D=5	D=10
40	RFTmPSO-s1	4.83 (0.87)	3.98 (0.39)	**3.91** (0.75)	6.14 (0.92)	**4.80** (0.62)	4.33 (0.47)
	RFTmPSO-s2	**4.98** (0.86)	**4.13** (0.46)	3.77 (0.89)	7.23 (1.59)	4.53 (0.67)	**4.39** (0.52)
	RFTmPSO-s3	4.08 (0.44)	3.81 (0.45)	3.76 (0.88)	6.82 (1.41)	4.21 (0.61)	**4.42** (0.57)
	RFTmPSO-s4	4.84 (0.86)	3.86 (0.45)	**3.83** (0.88)	**8.03** (1.62)	4.67 (0.68)	**4.49** (0.56)
45	RFTmPSO-s1	3.15 (0.48)	**2.55** (0.31)	**2.45** (0.43)	4.66 (0.87)	**3.34** (0.40)	**3.23** (0.41)
	RFTmPSO-s2	**3.24** (0.49)	**2.59** (0.30)	**2.46** (0.43)	**5.24** (1.14)	3.16 (0.38)	2.95 (0.34)
	RFTmPSO-s3	**3.31** (0.53)	2.40 (0.29)	2.36 (0.42)	5.17 (1.15)	3.00 (0.37)	3.02 (0.38)
	RFTmPSO-s4	**3.30** (0.51)	2.48 (0.28)	**2.46** (0.42)	**5.32** (1.14)	**3.23** (0.38)	3.00 (0.38)
50	RFTmPSO-s1	**1.99** (0.36)	**1.51** (0.19)	**1.48** (0.33)	2.40 (0.37)	**2.40** (0.35)	1.85 (0.26)
	RFTmPSO-s2	**1.92** (0.30)	**1.51** (0.18)	**1.45** (0.32)	2.57 (0.40)	2.10 (0.25)	**1.94** (0.28)
	RFTmPSO-s3	1.87 (0.31)	**1.46** (0.17)	**1.48** (0.32)	2.46 (0.40)	2.02 (0.24)	**1.94** (0.28)
	RFTmPSO-s4	**1.95** (0.33)	**1.51** (0.18)	**1.48** (0.33)	**2.68** (0.40)	2.18 (0.27)	**1.97** (0.29)

In instances in which each peak has its own randomly generated shift severity, RFTmPSO-s4 and RFTmPSO-s2 obtain the best results. In these instances, some peaks have higher values of shift severity which make them less reliable for carrying robust solutions and vice versa. Therefore, algorithms that learn about shift severities such as RFTmPSO-s4 and RFTmPSO-s2 can find more robust solutions. RFTmPSO-s4 obtains the best results due to using both types of information (shift severity and HeightVar). Similar to Tables 4.12 and 4.13, in Tables 4.14 and 4.15, the average survival time values are lower in 10-dimensions than in 5-dimensions.

Tables 4.16 and 4.17 show the average survival time by RFTmPSO in mMPBR with different numbers of peaks and change frequencies, with default values for other parameters. Like in previous experiments, RFTmPSO-s4 has better performance overall in environments with higher change frequencies. In problem instances with fewer evaluations per change (lower f, higher change frequency), the average survival time decreases

Table 4.13: Average survival time (and standard error) on mMPBR with different $m = \{20, 50, 100\}$, different D, $f = 2500$ and s randomized in [0.5,3].

V	Algorithm	$m = 20$			$m = 50$			$m = 100$		
		D=2	D=5	D=10	D=2	D=5	D=10	D=2	D=5	D=10
40	RFTmPSO-s1	7.42	5.48	4.35	7.87	7.41	5.37	7.41	6.19	4.97
		(1.09)	(0.60)	(0.35)	(1.04)	(0.83)	(0.44)	(2.00)	(0.47)	(0.43)
	RFTmPSO-s2	9.26	5.60	4.22	7.91	7.84	5.60	7.50	5.73	4.62
		(1.32)	(0.82)	(0.34)	(0.68)	(0.99)	(0.50)	(1.44)	(0.47)	(0.44)
	RFTmPSO-s3	7.48	4.89	4.05	8.15	7.01	5.46	6.91	5.87	5.21
		(1.17)	(0.81)	(0.43)	(0.98)	(0.96)	(0.78)	(0.69)	(0.71)	(0.46)
	RFTmPSO-s4	9.71	6.14	4.22	8.47	8.23	5.93	8.29	6.51	5.30
		(1.40)	(0.85)	(0.38)	(1.04)	(0.98)	(0.69)	(1.34)	(0.46)	(0.39)
45	RFTmPSO-s1	5.11	3.90	3.29	6.52	5.20	4.50	6.42	4.98	3.29
		(0.63)	(0.34)	(0.29)	(1.02)	(0.47)	(0.48)	(0.58)	(0.60)	(0.24)
	RFTmPSO-s2	5.67	3.63	3.30	6.10	5.73	4.74	6.06	5.05	3.26
		(0.68)	(0.33)	(0.29)	(0.59)	(0.63)	(0.58)	(0.48)	(0.55)	(0.25)
	RFTmPSO-s3	5.03	3.44	3.25	6.27	5.95	4.81	5.72	4.36	3.68
		(0.64)	(0.34)	(0.33)	(0.62)	(0.76)	(0.57)	(0.55)	(0.55)	(0.28)
	RFTmPSO-s4	5.65	4.22	3.51	7.03	6.16	5.03	6.48	5.43	3.87
		(0.68)	(0.41)	(0.33)	(1.23)	(0.62)	(0.57)	(0.52)	(0.58)	(0.27)
50	RFTmPSO-s1	3.32	2.55	2.26	4.52	3.65	2.99	4.27	3.17	2.34
		(0.39)	(0.21)	(0.21)	(0.86)	(0.40)	(0.35)	(0.36)	(0.25)	(0.14)
	RFTmPSO-s2	3.58	2.43	2.11	4.74	3.94	3.18	4.49	3.27	2.25
		(0.45)	(0.26)	(0.21)	(0.83)	(0.46)	(0.36)	(0.40)	(0.40)	(0.16)
	RFTmPSO-s3	3.49	2.51	2.23	4.62	3.93	3.06	3.75	3.22	2.51
		(0.46)	(0.28)	(0.25)	(0.65)	(0.55)	(0.36)	(0.37)	(0.31)	(0.15)
	RFTmPSO-s4	3.75	2.77	2.19	4.19	4.10	3.18	4.57	3.39	2.64
		(0.45)	(0.31)	(0.23)	(0.54)	(0.51)	(0.34)	(0.39)	(0.37)	(0.17)

because the accuracy of gathered information and the local search in each peak decrease. This is due to a lack of time to react to changes. For $f = 500$, the difference between results obtained by methods is small due to lower information accuracy. When f increases, the difference between the methods becomes more noticeable.

The average survival time in problems with a small number of peaks does not decrease significantly when f is small. The reason is that a small number of peaks means a small number of sub-swarms, so the algorithm has enough time for exploitation before the next environmental change. On the other hand, when the number of peaks is high, the algorithm has many sub-swarms and so can perform fewer iterations of exploiting before the next environmental change. This leads to less accurate information and lower performance.

TABLE 4.14: Average survival time (and standard error) on mMPBR with different shift severities s, $D = 5$, $m = 20$ and $f = 2500$.

V	Algorithm	5 Dimensional				
		s=1	s=5	s=r(0.5,1)	s=r(0.5,3)	s=r(0.5,5)
40	RFTmPSO-s1	7.87(0.80)	1.21(0.10)	10.17(0.64)	5.48(0.60)	5.35(0.58)
	RFTmPSO-s2	7.47(0.61)	1.08(0.07)	11.64(1.17)	5.60(0.82)	5.98(0.88)
	RFTmPSO-s3	8.26(0.69)	1.20(0.12)	10.40(0.92)	4.89(0.81)	5.45(0.91)
	RFTmPSO-s4	8.10(0.81)	1.16(0.10)	11.96(1.14)	6.14(0.85)	5.94(1.00)
40	RFTmPSO-s1	5.98(0.56)	0.78(0.07)	7.63(0.64)	3.90(0.34)	3.59(0.41)
	RFTmPSO-s2	5.76(0.56)	0.73(0.05)	6.87(0.60)	3.63(0.33)	3.51(0.46)
	RFTmPSO-s3	6.50(0.68)	0.79(0.06)	8.02(0.73)	3.44(0.34)	3.67(0.56)
	RFTmPSO-s4	6.42(0.71)	0.77(0.06)	8.38(0.72)	4.22(0.41)	3.85(0.50)
40	RFTmPSO-s1	4.49(0.52)	0.43(0.05)	5.21(0.37)	2.55(0.21)	2.18(0.26)
	RFTmPSO-s2	3.82(0.37)	0.41(0.04)	4.72(0.51)	2.43(0.26)	2.43(0.36)
	RFTmPSO-s3	4.63(0.50)	0.45(0.04)	5.50(0.60)	2.51(0.28)	2.40(0.36)
	RFTmPSO-s4	4.32(0.45)	0.42(0.04)	5.61(0.62)	2.77(0.31)	2.45(0.35)

TABLE 4.15: Average survival time (and standard error) on mMPBR with different shift severities s, $D = 10$, $m = 20$ and $f = 2500$.

V	Algorithm	10 Dimensional				
		s=1	s=5	s=r(0.5,1)	s=r(0.5,3)	s=r(0.5,5)
40	RFTmPSO-s1	5.26(0.37)	1.05(0.07)	9.19(1.11)	4.35(0.35)	3.94(0.46)
	RFTmPSO-s2	4.02(0.32)	1.02(0.08)	8.74(1.12)	4.22(0.34)	3.75(0.41)
	RFTmPSO-s3	5.68(0.57)	1.08(0.07)	9.63(1.26)	4.05(0.43)	3.70(0.50)
	RFTmPSO-s4	5.34(0.39)	1.08(0.07)	9.77(1.13)	4.22(0.38)	3.91(0.44)
40	RFTmPSO-s1	4.02(0.34)	0.73(0.05)	6.65(0.95)	3.29(0.29)	2.68(0.28)
	RFTmPSO-s2	3.25(0.28)	0.70(0.06)	6.61(1.12)	3.30(0.29)	2.67(0.28)
	RFTmPSO-s3	4.17(0.39)	0.75(0.06)	7.22(1.03)	3.25(0.33)	2.65(0.31)
	RFTmPSO-s4	4.10(0.34)	0.74(0.06)	7.62(1.16)	3.51(0.33)	2.79(0.32)
40	RFTmPSO-s1	2.56(0.22)	0.40(0.03)	4.86(1.00)	2.26(0.21)	1.80(0.25)
	RFTmPSO-s2	1.99(0.14)	0.40(0.03)	4.75(0.98)	2.11(0.21)	1.80(0.21)
	RFTmPSO-s3	2.63(0.22)	0.41(0.03)	5.36(1.09)	2.23(0.25)	1.74(0.27)
	RFTmPSO-s4	2.62(0.21)	0.41(0.03)	5.43(1.09)	2.19(0.23)	1.81(0.27)

4.3.2 Comparison with other methods

According to the reported results in Tables 4.10 to 4.17 and based on the multiple comparison statistical analysis, the fourth strategy outperforms other strategies of the proposed framework. In this part, three different multi-swarm methods including FTmPSO (Yazdani et al., 2013b), AmQSO (Blackwell et al., 2008) and mNAFSA (Yazdani et al., 2014) are used inside the proposed ROOT framework in combination with Strategy 4 (s4) to investigate the effect of the multi-swarm methods performance on the ROOT framework.

Table 4.16: Average fitness value (and standard error) on mMPBR with $m = \{5, 10\}$ and evaluation between changes f, s randomized in [0.5,3] and D=5.

V	Algorithm	$m = 5$			$m = 10$		
		f=500	f=1000	f=2500	f=500	f=1000	f=2500
40	RFTmPSO-s1	2.80 (0.28)	3.54 (0.73)	3.98 (0.39)	4.03 (0.34)	4.13 (0.46)	4.80 (0.62)
	RFTmPSO-s2	2.49 (0.29)	3.78 (0.79)	4.13 (0.46)	3.90 (0.32)	4.25 (0.49)	4.53 (0.67)
	RFTmPSO-s3	2.77 (0.29)	3.32 (0.65)	3.81 (0.45)	3.92 (0.42)	4.03 (0.44)	4.21 (0.61)
	RFTmPSO-s4	2.80 (0.28)	3.90 (0.80)	3.86 (0.45)	4.01 (0.29)	4.37 (0.48)	4.67 (0.68)
45	RFTmPSO-s1	2.04 (0.23)	2.06 (0.44)	2.55 (0.31)	2.80 (0.29)	3.05 (0.33)	3.34 (0.40)
	RFTmPSO-s2	1.94 (0.18)	2.23 (0.49)	2.59 (0.30)	2.82 (0.30)	3.04 (0.35)	3.16 (0.38)
	RFTmPSO-s3	2.05 (0.25)	2.13 (0.48)	2.40 (0.29)	2.77 (0.28)	3.08 (0.38)	3.00 (0.37)
	RFTmPSO-s4	2.07 (0.22)	2.23 (0.48)	2.48 (0.28)	2.85 (0.33)	3.11 (0.38)	3.23 (0.38)
50	RFTmPSO-s1	1.14 (0.10)	1.37 (0.24)	1.51 (0.19)	1.87 (0.22)	1.95 (0.29)	2.40 (0.35)
	RFTmPSO-s2	1.15 (0.10)	1.42 (0.24)	1.51 (0.18)	1.86 (0.20)	2.03 (0.29)	2.10 (0.25)
	RFTmPSO-s3	1.13 (0.11)	1.33 (0.24)	1.46 (0.17)	1.83 (0.20)	1.96 (0.32)	2.02 (0.24)
	RFTmPSO-s4	1.17 (0.10)	1.48 (0.24)	1.51 (0.18)	1.93 (0.21)	2.10 (0.32)	2.18 (0.27)

These three algorithms are called RFTmPSO-s4, RAmQSO-s4 and RmNAFSA-s4, and are compared against three existing methods. The first method is a TMO algorithm based on FTmPSO (Yazdani et al., 2013b) in which, when the current robust solution is not acceptable, the algorithm simply chooses the best found position as the next robust solution. Parameter settings of FTmPSO are the same as reported in Table 4.2 and parameter settings of AmQSO and mNAFSA are as proposed in their original references (Blackwell et al., 2008; Yazdani et al., 2014). As mentioned before, since the task of the multi-swarm methods in the proposed framework is the same as their original purpose, i.e, TMO, parameter settings suggested in the original papers can be used here as well. For RAmQSO-s4 and RmNAFSA-s4, the same exclusion mechanism as RFTmPSO-s4 with the same $excl_{\text{factor}}$ value is used. Additionally, both of them use the obtained value for shift severities in Eq. (4.1) instead of the actual value as an initial knowledge.

Table 4.17: Average fitness value (and standard error) on mMPBR with $m = \{20, 50, 100\}$ and evaluation between changes f, s randomized in [0.5,3] and D=5.

V	Algorithm	$m = 20$			$m = 50$			$m = 100$		
		f=500	f=1000	f=2500	f=500	f=1000	f=2500	f=500	f=1000	f=2500
40	RFTmPSO-s1	4.37	4.89	5.48	5.35	5.61	7.41	5.00	5.27	6.19
		(0.28)	(0.38)	(0.60)	(0.38)	(0.65)	(0.83)	(0.43)	(0.60)	(0.47)
	RFTmPSO-s2	4.36	5.10	5.60	5.55	5.86	7.84	4.95	5.52	5.73
		(0.30)	(0.40)	(0.82)	(0.39)	(0.70)	(0.99)	(0.51)	(0.68)	(0.47)
	RFTmPSO-s3	4.25	4.44	4.89	5.46	5.63	7.01	4.98	4.75	5.87
		(0.35)	(0.33)	(0.81)	(0.40)	(0.57)	(0.96)	(0.49)	(0.71)	(0.71)
	RFTmPSO-s4	4.42	5.00	6.14	5.50	5.94	8.23	5.36	5.63	6.51
		(0.27)	(0.41)	(0.85)	(0.41)	(0.70)	(0.98)	(0.56)	(0.70)	(0.46)
45	RFTmPSO-s1	3.28	3.51	3.90	3.87	4.27	5.20	3.60	3.68	4.98
		(0.23)	(0.30)	(0.34)	(0.30)	(0.50)	(0.47)	(0.34)	(0.40)	(0.60)
	RFTmPSO-s2	3.27	3.48	3.63	3.94	4.08	5.73	3.71	3.81	5.05
		(0.21)	(0.31)	(0.33)	(0.29)	(0.67)	(0.63)	(0.36)	(0.47)	(0.55)
	RFTmPSO-s3	3.23	3.35	3.44	3.92	4.33	5.95	3.59	4.07	4.36
		(0.24)	(0.29)	(0.34)	(0.30)	(0.78)	(0.76)	(0.31)	(0.49)	(0.55)
	RFTmPSO-s4	3.33	3.90	4.22	3.98	4.41	6.16	3.84	4.00	5.43
		(0.24)	(0.37)	(0.41)	(0.30)	(0.69)	(0.62)	(0.37)	(0.47)	(0.58)
50	RFTmPSO-s1	2.04	2.30	2.55	2.42	2.88	3.65	2.31	2.48	3.17
		(0.16)	(0.23)	(0.21)	(0.14)	(0.32)	(0.40)	(0.25)	(0.33)	(0.25)
	RFTmPSO-s2	2.04	2.33	2.43	2.42	2.57	3.94	2.40	2.37	3.27
		(0.17)	(0.19)	(0.26)	(0.15)	(0.25)	(0.46)	(0.26)	(0.35)	(0.40)
	RFTmPSO-s3	2.02	2.31	2.51	2.42	3.15	3.93	2.29	2.56	3.22
		(0.15)	(0.20)	(0.28)	(0.15)	(0.34)	(0.55)	(0.25)	(0.41)	(0.31)
	RFTmPSO-s4	2.09	2.44	2.77	2.44	3.00	4.10	2.44	2.55	3.39
		(0.17)	(0.21)	(0.31)	(0.16)	(0.28)	(0.51)	(0.26)	(0.37)	(0.37)

The other two methods are two reproduced versions of the method proposed by Fu et al. (2013), which are considered the state-of-the-art in the field of ROOT at the moment (Fu et al., 2013, 2015; Guo et al., 2014; Huang et al., 2017). The first version is exactly what was implemented in (Fu et al., 2013), utilizing the true values of previous environments instead of approximated values for training predictors, i.e., it assumes it does not need to use an approximator because it has access to the true values of previous environments. This method will be called ROOT with predicted values (ROOT-PV) and the parameter settings of PSO and AR are the same as those used in (Fu et al., 2013). The second version is reproduced from (Huang et al., 2017), in which the ROOT algorithm even had access to the future true values instead of having to approximate past fitness functions and predict future values. This method will be called ROOT with true future values (ROOT-TFV). Note that ROOT-TFV is the ROOT method proposed in (Fu et al., 2013) using the true future values, and was used in (Huang et al., 2017).

The reason behind choosing ROOT-TFV in the comparisons is to investigate the effect of prediction error on the performance of the ROOT algorithm. For PSO in ROOT-TFV, the same parameter setting as ROOT-PV is used. The obtained results of ROOT-PV will not be considered in environments for which the training datasets are not complete. Note that ROOT-PV and even more so ROOT-TFV have access to information that is

Table 4.18: Average survival time (and std. err) on test instances with different dimension D and peak number m, $f = 2500$ and s randomized in [0.5,3]. Best results based on Wilcoxon signed-rank test with Holm-Bonferroni p-value correction, $\alpha = 0.05$ are highlighted, ignoring ROOT-TFV due to its unrealistic assumption of knowing the true future fitness.

V	Algorithm	Survival time			
		D=2,P=5	D=2,P=20	D=5,P=5	D=5,P=20
40	ROOT-PV	3.10(0.39)	5.64(0.80)	0.83(0.16)	2.26(0.61)
	ROOT-TFV	~~5.74(0.47)~~	~~8.06(0.89)~~	~~1.26(0.19)~~	~~3.29(0.33)~~
	FTmPSO(TMO)	4.34(0.79)	6.18(1.00)	3.49(0.34)	4.22(0.33)
	RAmQSO-s4	5.40(0.91)	6.20(0.56)	4.15(0.87)	5.58(0.63)
	RmNAFSA-s4	4.72(0.83)	7.45(1.09)	3.71(0.52)	5.70(0.49)
	RFTmPSO-s4	4.84(0.86)	9.71(1.40)	3.86(0.45)	6.14(0.85)
45	ROOT-PV	2.71(0.26)	4.91(1.15)	0.13(0.05)	1.11(0.23)
	ROOT-TFV	~~3.93(0.36)~~	~~6.87(0.60)~~	~~0.29(0.08)~~	~~1.58(0.20)~~
	FTmPSO(TMO)	2.97(0.57)	3.71(0.42)	2.22(0.26)	3.26(0.22)
	RAmQSO-s4	3.39(0.67)	4.79(0.56)	2.64(0.59)	4.16(0.55)
	RmNAFSA-s4	3.28(0.59)	4.96(0.60)	2.40(0.33)	4.15(0.38)
	RFTmPSO-s4	3.30(0.51)	5.65(0.68)	2.48(0.28)	4.22(0.41)
50	ROOT-PV	1.68(0.22)	2.83(0.37)	0.04(0.06)	0.37(0.23)
	ROOT-TFV	~~2.48(0.34)~~	~~4.36(0.26)~~	~~0.19(0.05)~~	~~0.63(0.11)~~
	FTmPSO(TMO)	1.79(0.31)	2.41(0.19)	1.29(0.16)	2.09(0.18)
	RAmQSO-s4	2.16(0.38)	3.21(0.50)	1.63(0.46)	2.55(0.28)
	RmNAFSA-s4	1.95(0.35)	3.44(0.48)	1.52(0.19)	2.51(0.22)
	RFTmPSO-s4	1.95(0.33)	3.75(0.45)	1.51(0.18)	2.77(0.31)

not available in real-world optimization, and thus results can only be taken as an upper bound of what these methods are able to achieve in practice. As mentioned before, it is assumed that the algorithms are informed when environmental changes happen.

The experiments in this section are done on four different test instances of mMPBR on 2 and 5 dimensions with 5 and 20 peaks (all other parameters have default values). This combination shows how tested methods perform across different dimensions and numbers of peaks. The results of ROOT-PV, ROOT-TFV, FTmPSO, RFTmPSO-s4, RAmQSO-s4 and RmNAFSA-s4 are summarized in Tables 4.18 and 4.19.

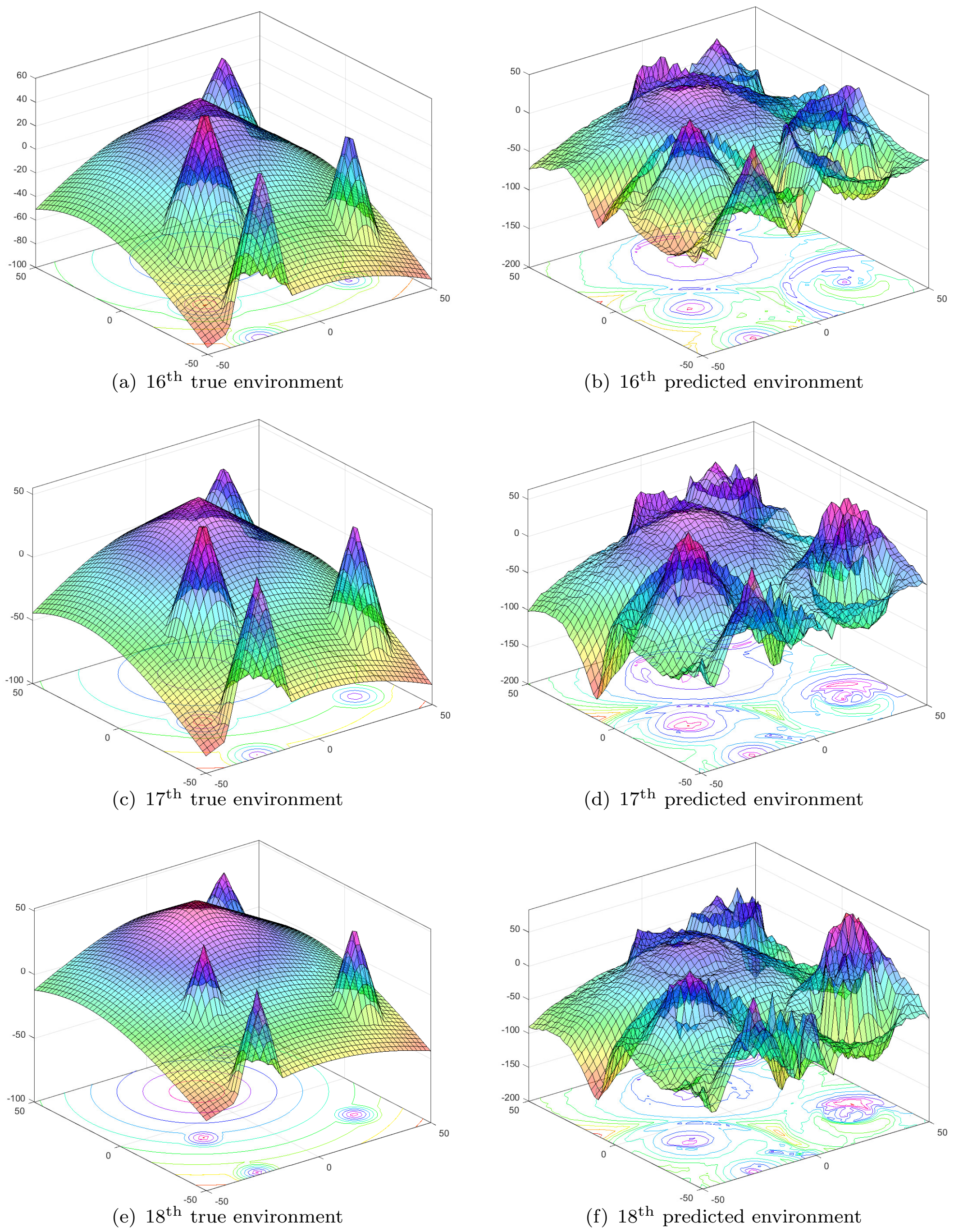
According to Table 4.18, not surprisingly, the average survival time of ROOT-TFV is better than that of ROOT-PV in all tests since ROOT-TFV eliminates predictor errors by assuming perfect knowledge of peak movements. Also, the autoregressive model, used by ROOT-PV as predictor (Fu et al., 2013), uses true values of solutions fitness values in previous environments for training. In a practical application where such information is not available, the performance of both algorithms will likely be worse.

Table 4.19: Average fitness values (and std. err) on test instances with different dimension D and peak number m, $f = 2500$ and s randomized in [0.5,3]. Best results based on Wilcoxon signed-rank test with Holm-Bonferroni p-value correction, $\alpha = 0.05$ are highlighted, ignoring ROOT-TFV due to its unrealistic assumption of knowing the true future fitness.

V	Algorithm	Survival time			
		D=2,P=5	D=2,P=20	D=5,P=5	D=5,P=20
40	ROOT-PV	48.35(0.32)	50.79(0.33)	-53.99(15.53)	20.49(3.04)
	ROOT-TFV	49.82(0.18)	51.64(0.21)	-54.37(13.45)	32.22(1.46)
	FTmPSO(TMO)	53.94(0.26)	55.53(0.20)	53.04(0.25)	54.68(0.16)
	RAmQSO-s4	52.19(0.32)	52.57(0.28)	51.45(0.32)	50.88(0.36)
	RmNAFSA-s4	52.60(0.35)	52.46(0.27)	51.48(0.29)	51.09(0.32)
	RFTmPSO-s4	52.52(0.36)	52.40(0.29)	51.02(0.31)	51.26(0.41)
45	ROOT-PV	50.70(0.39)	53.51(0.20)	-124.90(17.06)	2.92 (6.80)
	ROOT-TFV	51.22(0.24)	54.24(0.23)	-133.3(16.58)	16.19(3.92)
	FTmPSO(TMO)	56.47(0.18)	58.24(0.16)	56.02(0.21)	57.22(0.15)
	RAmQSO-s4	55.00(0.27)	55.93(0.23)	54.90(0.21)	54.23(0.27)
	RmNAFSA-s4	54.95(0.26)	55.52(0.25)	54.91(0.23)	54.44(0.29)
	RFTmPSO-s4	55.34(0.26)	55.14(0.23)	54.55(0.21)	54.53(0.26)
50	ROOT-PV	51.40(0.60)	56.45(0.12)	-190.50(19.76)	-37.01(7.55)
	ROOT-TFV	51.49(0.61)	56.66(0.09)	-116.73(14.48)	-11.98(6.18)
	FTmPSO(TMO)	58.79(0.19)	61.02(0.13)	58.56(0.23)	60.04(0.14)
	RAmQSO-s4	57.47(0.30)	59.15(0.13)	57.73(0.18)	57.94(0.16)
	RmNAFSA-s4	57.88(0.24)	58.47(0.17)	57.51(0.22)	57.81(0.19)
	RFTmPSO-s4	58.09(0.22)	58.36(0.19)	57.68(0.21)	58.14(0.18)

Figure 4.1 compares the true and predicted landscapes in D=2. Each environment is produced by 2,500 points, and the parameter setting of mMPBR is based on default values in Table 4.1 with m=5. The first 15 environments are used to train the predictor (Jin et al., 2013; Fu et al., 2013). Figure 4.1 shows that the error of the predictor is noticeable even though the true fitness values in previous environments are used to train it.

As can be seen in Table 4.18, ROOT-TFV has the highest average survival time in test instances with D=2 but loses its superiority in problems with D=5 and its performance, as well as that of ROOT-PV, experience a dramatic drop. To understand why these two methods struggle with even moderately dimensional problems, one has to note that they use Eq. (2.10) as fitness instead of the true fitness function Eq. (3.10). Figure 4.2 visualizes an example of the search space according to Eq. (2.10) in D=2. Figure 4.2(a) shows the true fitness landscape according to Eq. (3.10) and Fig. 4.2(b) shows the corresponding environment based on Eq. (2.10) with V=40 and its true five future environments. As can be seen, most of the problem landscapes defined by Eq. (2.10) are



(a) 16th true environment

(b) 16th predicted environment

(c) 17th true environment

(d) 17th predicted environment

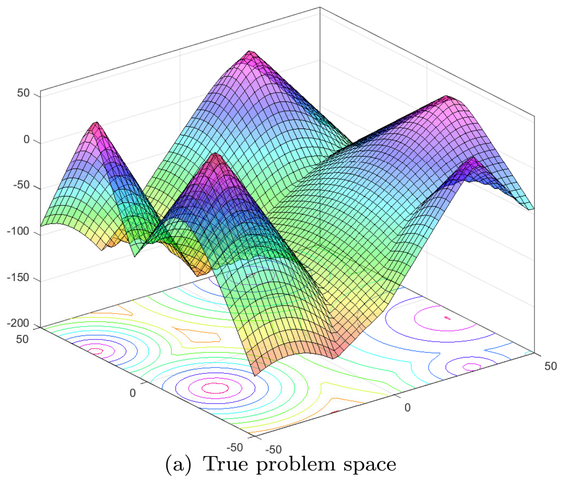
(e) 18th true environment

(f) 18th predicted environment

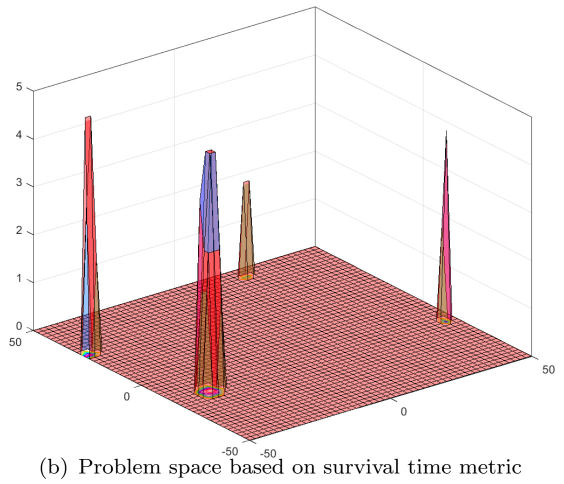
FIGURE 4.1: An example of mMPBR in dimension D=2 to show the error of the predictor.

flat with a few narrow peaks. This is really challenging for the optimizer, especially in higher dimensions.

To investigate the performance of PSO in this type of environments, PSO is used for optimizing the mMPBR with 5 peaks and 100 environmental changes. This experiment is done 50 times and at the end of each environment, the *Gbest* value of PSO based on the environment made by Eq. (2.10) is saved. The average *Gbest* values are reported in



(a) True problem space



(b) Problem space based on survival time metric

FIGURE 4.2: The search space made by Eq. (2.10) with a threshold V=40, dimension D=2 and peak number m=5 versus the true problem space.

Table 4.20. Experiments for Table 4.20 are done on mMPBR in 2, 5 and 10 dimensions and with the number of evaluations per change f of 2500 and 10000 and $V = 40$.

TABLE 4.20: Average Gbest value (standard error in parenthesis) of PSO in search space made by Eq. (2.10) with different dimension D.

Parameter settings	D=2	D=5	D=10
Population size=50, f=2500	5.02(0.17)	2.78(0.32)	0(0)
Population size=100, f=10000	5.31(0.16)	3.42(0.27)	0(0)

For f=2500, PSO with 50 particles is used and for f=10000, the population size is increased to 100. With D=2, although the second PSO benefits from a larger population size and more time to do exploration and exploitation in each environment, its performance is not so much better than the first PSO. With D=5, the performance of both PSOs decreases significantly and the results show that they are not able to find the best peak. Furthermore, the difference between the first and the second PSO increases relatively to their results in $D = 2$. This shows that the PSO needs more particles and time to deal with this type of environment. PSO fails to find peaks in $D = 10$.

Given the results in Table 4.20 and the fact that the search environment is shaped by the survival time metric Eq. (2.10) (an example is shown in Fig. 4.2(b)), it is concluded that with increasing dimension, the search space becomes very challenging for optimizers using the survival time metric Eq. (2.10). This was confirmed by Huang et al. (2017) where ROOT methods based on Eq. (2.10) have poor performance in higher dimensions. The provided analysis indicated an explanation for this behavior.

Table 4.18 shows that the performance of FTmPSO, designed for TMO but used as ROOT algorithm, is better than ROOT-PV in most test instances and works surprisingly well at finding robust solutions. The only other paper that has investigated the performance of population-based algorithms designed for TMO in the context of ROOT is (Jin et al., 2013), and according to the reported results and analysis in this chapter, some TMO algorithms also succeeded in finding robust solutions in ROOT. The reason behind the acceptable performance of some TMO based algorithms in ROOT is that in most research in the DOP domain, researchers have been working on DOPs with small changes, where the obtained knowledge from the current environment is useful for improving the optimization process in the next environment. In this type of environments which were also used in most ROOT papers, solutions around the peak centers can be robust solutions. Indeed, when comparing Fig. 4.2(a) and Fig. 4.2(b), it can be seen that robust solutions are around peak centers.

As can be seen in Table 4.19, the methods based on the proposed ROOT framework with strategy four can perform really well in maximizing the average survival time of robust solutions. All of RFTmPSO-s4, RAmQSO-s4 and RmNAFSA-s4 outperform ROOT-PV in all test instances in this section and only ROOT-TFV (Fu et al., 2013) (which, as mentioned before, is an unrealistic version of the ROOT algorithm due to its assumed knowledge of future environments) has better results in test problems with D=2. The average fitness value Eq. (2.13) of robust solutions obtained by the TMO algorithm is the best in all test instances because this algorithm chooses the best found solution in terms of fitness value.

For all three methods based on the proposed ROOT framework, the average fitness value of robust solutions in all test instances is better than that of ROOT-PV and ROOT-TFV because the proposed methods search the problem space with actual fitness values and choose one of the peaks as a robust solution. On the other hand, ROOT-PV and ROOT-TFV use the survival time metric and thus can get stuck in flat areas (Fig. 4.2(b)). For the same reason, their average fitness value can be very poor in problems with higher D and V (e.g. these two algorithms achieve negative average fitness values in D=5, m=5).

In this section, three different multi-swarm methods are embedded in the proposed ROOT framework. The reported results in Table 4.18 show that the proposed algorithms are able to perform better than previous state-of-the-art survival time metric Eq. (2.10) based methods especially on the environments with a higher number of dimensions.

By comparing results of RFTmPSO-s4, RAmQSO-s4 and RmNAFSA-s4, it is concluded that the quality of swarms in finding and tracking peaks can improve the proposed framework's performance noticeably. Specifically, better peak finding and tracking performance corresponds to more accurate information (gathered by (4.1), (4.2) and (4.3)) leading to more reliable decision making by (4.4), (4.5), (4.6) and (4.7).

4.4 Summary

A new framework for robust optimization over time (ROOT) was proposed. In the proposed framework, a multi-swarm/multi-population method is responsible for finding, tracking and monitoring peaks. Each sub-swarm gathers information about its covered peak. This information is used to predict the future behavior of the peak and pick the next robust solution in case the current solution becomes unacceptable. Three types of information based on shift severity, height variance and fitness variance of peaks were used and four different solution selection strategies based on this information were designed. The experimental results showed that the fourth strategy that uses the information about shift severity as well as height variance of peaks had the best performance overall and can be used for other problem instances.

A wide range of problem settings is used to investigate the performance of the proposed framework based algorithms versus the existing state-of-the-art framework based on a survival time metric. It is shown that the performance of previous methods that use the survival time metric is substantially worse in problems with higher dimensions. All previous state-of-the-art methods attempt to predict future fitness values of solutions based on previous fitness values of solutions. However, this is a difficult task and can become almost impossible for problems with higher dimensions, larger search space and higher change frequencies. In the experiments, the effect of predictor errors and approximation errors on the performance of previous methods are investigated. On the other hand, the proposed framework does not have to deal with the challenges of predicting future fitness values. The experimental results show that the performance of the proposed framework is significantly better than that of state-of-the-art methods especially in problems with higher dimensions.

The proposed framework based algorithms are tested on problem instances with different combinations of parameter settings of mMPBR and provided performance analysis based on them. The results showed that the problem becomes more challenging when shift severities of peaks, dimension of problem space, and change frequency are higher. However, the reported results showed that the proposed methods were capable of performing very well even in more challenging problems.

Although the experimental results in this chapter showed the effectiveness of using peaks' behavior information for choosing robust solutions, the proposed framework has several shortcomings that can limit its applicability in certain situations. In fact, the efficiency of the proposed framework is depended on the accuracy of the information gathered by trackers. Therefore, if the performance of the trackers drops, then the performance of the proposed framework will decrease significantly. One example is when the number of peaks are high and they do not have overlapping. As a result, the algorithm needs to create many trackers which leads to consume a large amount of computational resource. Therefore, trackers will fail to exploit and track efficiently before environmental changes which results to provide inaccurate information for the strategies of choosing next robust solution.

Chapter 5

Changing or keeping solutions in dynamic optimization problems with switching costs

As described previously, most previous research on DOPs focuses on TMO (Nguyen et al., 2012a). In TMO, the algorithm assumes that the solution can be changed for each environment without considering any switching cost and/or any resource limitation for changing solutions. Thus, lack of switching cost consideration in TMO algorithms makes them unsuitable for many real-world problems. For addressing this issue, ROOT (Yu et al., 2010) was proposed in which algorithms search for solutions that can remain acceptable after environmental changes. In fact, TMO and ROOT address two extreme cases. TMO is suitable for circumstances in which there is no switching cost or it is very low. On the other hand, ROOT is suitable for situations in which the switching cost is very high so the algorithm tries to keep each solution as long as it remains acceptable after environmental changes.

In this chapter, a new adaptive solution chooser (ASC) algorithm is proposed which acts in a similar way to TMO algorithms where switching cost is low and acts as ROOT algorithms when the switching cost is high. However, the main contribution of ASC is where the algorithm can decide about changing or keeping solutions based on their current fitness values, the fitness of other found solutions with better quality and their switching cost from the current solution. Indeed, although changing solutions in real-world problems is costly, there are situations in which the algorithm has found a solution whose quality is so high that the benefit of switching largely outweighs the cost.

5.1 Proposed algorithm

In this section ASC is described. ASC tries to maintain a trade-off between TMO and ROOT characteristics based on the switching cost and fitness values of the current solution and other peaks. In this chapter, the SC is defined based on the Euclidean distance as follows:

$$\mathrm{SC}(\mathbf{x}) = w \cdot \frac{\|\mathbf{x} - \mathbf{x}^*\|}{\sqrt{D}}, \tag{5.1}$$

where SC is switching cost, $\mathbf{x}$ is a design variable, $\mathbf{x}^*$ is the last chosen solution by the algorithm, D is dimension and $w \geq 0$ is a weight which controls the ratio between switching cost and fitness value. Moreover, by setting different values for w, higher or lower switching costs can be simulated. Note that increasing the number of dimensions results in increasing Euclidean distance values between random points in the search space which would lead to increased SC values. Therefore, the ratio between SC and fitness value changes depending on the number of dimensions that can be undesirable in experiments. Consequently, it is divided by $\sqrt{D}$ in Eq. (5.1) which makes SC values independent from the number of dimensions. On the other hand, w can be used for increasing SC values in higher dimensions if it is desired.

In ASC, a multi-swarm optimizer is responsible to find peaks, track them after environmental changes and calculate their fitness variance. This multi-swarm algorithm needs to continuously try to identify new peaks and tracks them after each environmental change. Knowledge about the problem such as number of peaks and their shift severities should not be necessary. Additionally, the algorithm should be able to adapt the number of populations as needed. For example, the proposed multi-swarm algorithms by Yazdani et al. (2014, 2017); Blackwell et al. (2008) have such characteristics. Each sub-swarm which is tracking a peak needs to store the Euclidean distance between best found positions (such as Gbest in PSO (Kennedy and Eberhart, 1995)) at the end of each successive pair of environments. The average of these distances indicates the peak's Shift Severity. Moreover, the differences between fitness values of its best found positions before and after each environmental change. The average of these values indicates the variance of fitness values of the best found position after environmental changes which is denoted fitness variance.

Like previous chapters, FTmPSO (Yazdani et al., 2013b) is chosen as the multi-swarm method embedded in ASC. The major reasons behind this choice are its simplicity, competitiveness and compatibility with ASC. To make FTmPSO simpler, the exploiter particle and awakening-sleeping mechanisms proposed in its original paper are not used hear. Additionally, to make it more realistic, the exclusion radius formula in Eq. (3.6) is used for it and the learned shift severities is used instead of the true shift that was used in the original paper.

At each environment, ASC determines the reliable peaks. Reliable peaks are peaks which are expected to remain acceptable after at least one environmental change. For determining reliable peaks, ASC uses the following formula:

$$\begin{cases} \text{if } f(\mathbf{x}_{\text{best},i}, \theta^{(t)}) - \gamma_i \geq V & \text{reliable} \\ \text{else} & \text{unreliable}, \end{cases} \tag{5.2}$$

where $\mathbf{x}_{\text{best},i}$ is the best position found by the i^{th} sub-swarm and γ_i is the fitness variance of the peak which is covered by i^{th} sub-swarm.

For each environment, after a predefined computational budget, ASC determines reliable peaks then there are three different possibilities:

1. If the last chosen solution is not acceptable in the current environment t, i.e. $f(\mathbf{x}^*, \theta^{(t)}) < V$ then a new solution must be chosen from the reliable peaks as follows:

   $$j = \text{argmin}_i \text{SC}(\mathbf{x}_{\text{best},i}), \tag{5.3}$$

   then

   $$\mathbf{x}^*_{\text{new}} = \mathbf{x}_{\text{best},j}, \tag{5.4}$$

   where $i \in \{ReliablePeaks\}$ which are determined by Eq. (5.2). If there is no reliable peak, ASC chooses the best found solution.

2. If the last chosen solution is still acceptable i.e. $f(\mathbf{x}^*) \geq V$ and if among peaks, there is at least one peak which has the following condition:

   $$f(\mathbf{x}^*, \theta^{(t)}) < (f(\mathbf{x}_{\text{best},i}, \theta^{(t)}) - SC(\mathbf{x}_{\text{best},i})) \tag{5.5}$$

   Then, the solution will be changed to $\mathbf{x}_{\text{best},i}$. If there is more than one reliable peak that has the condition in Eq. (5.5), the one with the lowest $SC(\mathbf{x}_{\text{best},i})$ will be chosen.

3. If $f(\mathbf{x}^*, \theta^{(t)}) \geq V$ and there is no peak that has the condition in Eq. (5.5), then the previous solution will be kept for at least another environment.

Algorithm 4: ASC

```
1  Initialize multi-swarm method;
2  repeat
3      if an environmental change happens then
4          forall sub-swarms do
5              Update database;
6              Calculating shift severity and fitness variance;
7              Introducing diversity;
8              Update memory;
9      if the computational budget has been used up then
10         Determine reliable peaks by Eq. (5.2);
11         if $f(\mathbf{x}^*, \theta^{(t)}) < V$ then
12             Choose the next solution by Eq. (5.4);
13         else
14             if there are peaks with condition of Eq. (5.5) then
15                 Choose the one with minimum $SC(\mathbf{x}_{\mathrm{best},i})$;
16             else
17                 Keep the current solution;
18     Execute an iteration of the multi-swarm method;
19 until stopping criterion is met;
```

5.2 Experiments

5.2.1 Performance indicator

For measuring performance, the following performance indicator is proposed

$$\text{Performance} = \frac{1}{T}\sum_{t=1}^{T}(F_t), \tag{5.6}$$

where

$$F_t = \begin{cases} f(\mathbf{x}^*, \theta^{(t)}) & \text{if previous solution is kept} \\ f(\mathbf{x}^*_{\text{new}}, \theta^{(t)}) - SC(\mathbf{x}^*_{\text{new}}) & \text{if a new solution is chosen,} \end{cases} \tag{5.7}$$

where $\mathbf{x}^*_{\text{new}}$ is a new chosen solution and T is the number of environments. Therefore, the value of SC is decreased from the fitness value where the solution is changed. From another point of view, its cost is decreased from profit. Moreover, it can be seen as a penalty value.

5.2.2 Benchmark

For experiments in this chapter, mMPBR from Chapter 4.2.2 is used. The parameter settings of mMPBR in this chapter is shown in Table 5.1.

TABLE 5.1: Parameter settings of mMPBR

Parameter	Value(s)
Number of peaks, m	5,20
Evaluations between changes, f	2500
Shift severity, s	Randomized in [0.5,3]
Height severity, α	Randomized in [1,15]
Width severity, β	Randomized in [0.1,1.5]
Peaks shape	Cone
Correlation coefficient, λ	0
Number of dimensions, D	2,5
Peaks location range, SR	[-50,50]
Peak height, h	[30,70]
Peak width, w	[1,12]
Initial height value	50
Initial width value	6
Number of environments, T	100

5.2.3 Algorithms and parameter settings

For comparison, FTmPSO is chosen as a TMO method (TFTmPSO) which changes solutions to the best found position in each environment. Additionally, a ROOT version of FTmPSO (RFTmPSO) proposed by Yazdani et al. (2017) is used as a ROOT method. Since ASC, TFTmPSO and RFTmPSO methods use the same multi-swarm method as core, the conclusion about performance of their different decision making procedure for choosing the next solution will not be affected by differences in the quality of finding and tracking peaks. Since in all of these three methods, the main task of the FTmPSO is finding and tracking peaks, it seemed appropriate to use the suggested parameter settings as in its original paper (Yazdani et al., 2013b). The parameter settings of FTmPSO in all three methods are shown in Table 5.2. All three algorithms choose solutions (if needed) at the end of each environment meaning the computational budget is f-1. Moreover, it is assumed that all algorithms will be informed about environmental changes happening. Change detection is another issue that can be dealt with separately, see e.g. Nguyen et al. (2012a).

Table 5.2: The parameter settings of FTmPSO inside the ASC, TFTmPSO and RFTmPSO

Parameter	Value
$C1, C2$	2.05
χ	0.729843788
$Trackers' \ population \ size$	5
$Finder's \ population \ size$	10
$Exclusion \ fatcor$	0.5
P	1
Q	1
$Convergence \ limit$	1
k	10
Stop criterion	Max fitness evaluation number

5.2.4 Experimental results

All experimental results are obtained by performing 30 independent runs and the best results based on Wilcoxon signed-rank test with significance level of 0.05 are highlighted in each table. All experiments are done for three different fitness acceptance thresholds $V \in \{40, 45, 50\}$ and five different values of $w \in \{0.1, 0.5, 1, 2, 3\}$ in Eq. (5.1) to simulate the impact of different levels of switching cost on the performance which is measured by Eq. (5.6). The median, mean and standard error of results are reported in Tables 5.3 to 5.6.

Tables 5.3 to 5.6 show the obtained results by algorithms on mMPB with 5 and 20 peaks in 2 and 5 dimensional search space. When w=0.1, the amount of switching cost obtained by Eq.(5.1) is smaller. Therefore, the problem is more suitable to be solved by TFTmPSO rather than RFTmPSO. As a result, the efficiency of the TFTmPSO which chooses the best solution for each environment is much better than that of RFTmPSO. Additionally, obtained results by TFTmPSO are independent of V. Indeed, RFTmPSO tries to keep solutions as long as they are larger than V which is not useful for problems with small switching costs. In this situation, with growing V value, the performance of RFTmPSO improves due to more frequent solution changing to better ones. In this situation ASC acts like a TFTmPSO because the possibility of having solutions with condition in Eq. (5.5) is high. Therefore, ASC's results are almost the same with different V values.

According to Tables 5.3 to 5.6, increasing w results in decreasing the performance of algorithms because of higher values of switching cost. However, TFTmPSO suffers more than the other two methods in this situation and its performance drops dramatically. The reason is that TFTmPSO changes solution every environment and this is detrimental if cost is large. In problems with higher switching cost, RFTmPSO outperforms

TABLE 5.3: Results obtained by Eq.(5.6) by TFTmPSO, RFTmPSO and ASC on 2-dimensional mMPBR with 5 peaks.

			2 Dimensional				
V	Alg.	Stats.	w=0.1	w=0.5	w=1	w=2	w=3
40	TFTmPSO	Median	61.32	53.97	46.57	31.11	14.89
		Mean	61.04	54.28	45.83	28.93	12.03
		StdErr	0.27	0.47	0.87	1.73	2.61
	RFTmPSO	Median	52.45	49.93	47.09	41.40	35.92
		Mean	52.63	49.99	46.69	40.09	33.49
		StdErr	0.23	0.33	0.52	0.97	1.43
	ASC	Median	61.29	57.38	54.31	48.79	45.05
		Mean	60.68	57.02	54.24	49.04	44.00
		StdErr	0.28	0.40	0.54	0.87	1.23
45	TFTmPSO	Median	61.32	53.97	46.57	31.11	14.89
		Mean	61.04	54.28	45.83	28.93	12.03
		StdErr	0.27	0.47	0.87	1.73	2.61
	RFTmPSO	Median	54.92	51.51	47.40	38.95	30.30
		Mean	54.89	51.49	47.24	38.74	30.24
		StdErr	0.19	0.31	0.56	1.11	1.66
	ASC	Median	60.69	57.28	54.32	47.77	39.42
		Mean	60.58	56.95	53.36	46.34	39.26
		StdErr	0.28	0.43	0.69	1.22	1.77
50	TFTmPSO	Median	61.32	53.97	46.57	31.11	14.89
		Mean	61.04	54.28	45.83	28.93	12.03
		StdErr	0.27	0.47	0.87	1.73	2.61
	RFTmPSO	Median	57.52	52.99	46.98	35.65	24.32
		Mean	57.25	52.90	47.46	36.59	25.72
		StdErr	0.20	0.30	0.55	1.10	1.66
	ASC	Median	61.01	56.52	52.78	42.97	33.09
		Mean	60.73	56.54	51.68	41.95	32.25
		StdErr	0.27	0.44	0.72	1.32	1.91

TFTmPSO and the gap between their performances become larger as w increases. In fact, in this situation, the problem become more suitable to be solved by ROOT based methods in which solutions are kept as much as they remain acceptable.

ASC obtains the best results in comparison with RFTmPSO and TFTmPSO when switching cost is higher. Indeed, ASC is an adaptive algorithm which with growing switching cost tries to act in a more similar way to ROOT based methods and less to TMO based algorithms. According to these tables, ASC outperforms TFTmPSO and RFTmPSO when $w \geq 0.5$. Surprisingly, ASC keeps its superiority over RFTmPSO even when w=3 in which ASC is expected to act in a similar way to RFTmPSO. The first reason is that even with large w, it is still possible to have solutions with condition in Eq. (5.5) which can increase the performance of ASC. The second reason is their different strategies for choosing a solution when the current one is not acceptable anymore. Both

TABLE 5.4: Results obtained by Eq.(5.6) by TFTmPSO, RFTmPSO and ASC on 5-dimensional mMPBR with 5 peaks.

			5 Dimensional				
V	Alg.	Stats.	w=0.1	w=0.5	w=1	w=2	w=3
40	TFTmPSO	Median	61.30	54.58	45.67	29.83	14.12
		Mean	61.00	54.41	46.18	29.71	13.25
		StdErr	0.31	0.48	0.80	1.51	2.23
	RFTmPSO	Median	51.92	48.77	45.87	38.09	31.05
		Mean	52.07	49.09	45.37	37.92	30.47
		StdErr	0.20	0.35	0.59	1.10	1.62
	ASC	Median	60.38	56.79	54.71	50.52	45.83
		Mean	60.43	56.61	54.17	49.30	44.08
		StdErr	0.34	0.50	0.69	1.15	1.64
45	TFTmPSO	Median	61.30	54.58	45.67	29.83	14.12
		Mean	61.00	54.41	46.18	29.71	13.25
		StdErr	0.31	0.48	0.80	1.51	2.23
	RFTmPSO	Median	54.44	50.76	46.76	38.87	30.10
		Mean	54.47	50.73	46.05	36.70	27.35
		StdErr	0.18	0.37	0.70	1.40	2.10
	ASC	Median	60.43	56.55	52.85	46.22	39.42
		Mean	60.47	56.50	52.69	45.12	37.38
		StdErr	0.35	0.52	0.82	1.47	2.11
50	TFTmPSO	Median	61.30	54.58	45.67	29.83	14.12
		Mean	61.00	54.41	46.18	29.71	13.25
		StdErr	0.31	0.48	0.80	1.51	2.23
	RFTmPSO	Median	57.11	52.23	46.66	35.73	25.33
		Mean	57.00	52.39	46.62	35.08	23.54
		StdErr	0.22	0.38	0.71	1.41	2.13
	ASC	Median	60.68	56.33	51.07	40.24	29.62
		Mean	60.63	56.31	51.22	40.82	30.25
		StdErr	0.32	0.56	0.91	1.63	2.33

RFTmPSO and ASC choose solutions from reliable peaks. RFTmPSO chooses the best reliable peak in terms of fitness function. However, ASC chooses the one with lowest switching cost which leads to improved performance of ASC under circumstances with large switching cost.

Different values of V affect the performance of ASC (except where w=0.1) and RFTmPSO. When w=0.1, ASC acts independent from V but RFTmPSO obtains better results when V is higher. The reason is that when V is higher, solutions become unacceptable more frequently and RFTmPSO needs to change solutions to better ones and since switching cost is low, changing to better solutions improves its performance. On the other hand, when w is larger, by increasing V the performance of ASC and RFTmPSO get worse because they need to change solutions more frequently which leads to higher switching costs.

v

Table 5.5: Results obtained by Eq.(5.6) by TFTmPSO , RFTmPSO and ASC on 2-dimensional mMPBR with 20 peaks.

			2 Dimensional				
V	Alg.	Stats.	w=0.1	w=0.5	w=1	w=2	w=3
40	TFTmPSO	Median	64.91	54.03	40.40	13.23	-13.95
		Mean	64.96	53.94	40.17	12.63	-14.91
		StdErr	0.09	0.31	0.62	1.24	1.85
	RFTmPSO	Median	54.36	52.49	50.33	46.01	41.39
		Mean	54.63	52.78	50.46	45.83	41.21
		StdErr	0.22	0.28	0.42	0.73	1.06
	ASC	Median	64.72	60.69	58.25	54.92	52.18
		Mean	64.73	60.86	58.05	54.83	52.17
		StdErr	0.14	0.27	0.37	0.40	0.47
45	TFTmPSO	Median	64.91	54.03	40.40	13.23	-13.95
		Mean	64.96	53.94	40.17	12.63	-14.91
		StdErr	0.09	0.31	0.62	1.24	1.85
	RFTmPSO	Median	56.91	54.19	51.42	45.05	38.44
		Mean	56.82	54.26	51.06	44.65	38.25
		StdErr	0.14	0.22	0.41	0.84	1.28
	ASC	Median	64.68	61.15	58.79	55.86	52.01
		Mean	64.63	61.03	58.83	55.20	51.67
		StdErr	0.15	0.22	0.25	0.40	0.57
50	TFTmPSO	Median	64.91	54.03	40.40	13.23	-13.95
		Mean	64.96	53.94	40.17	12.63	-14.91
		StdErr	0.09	0.31	0.62	1.24	1.85
	RFTmPSO	Median	59.40	55.70	51.34	42.70	33.60
		Mean	59.23	55.51	50.85	41.54	32.23
		StdErr	0.13	0.24	0.47	0.95	1.44
	ASC	Median	64.76	60.85	58.42	52.47	47.88
		Mean	64.69	61.00	58.19	52.76	47.26
		StdErr	0.12	0.25	0.43	0.79	1.15

As can be seen in Tables 5.3 to 5.6, the results of all of the algorithms are better in mMPB with 20 peaks in comparison with 5 peaks. When the number of peaks is higher in mMPB, the possibility of having taller peaks is higher which leads to improve the performance when w is smaller, especially for ASC and TFTmPSO . Moreover, when w is higher, the performance of ASC and RFTmPSO are more dependent on the robustness of solutions. Therefore, having more peaks increases the possibility of having more reliable peaks by Eq. (5.2). In addition, when the number of peaks is low, there are large areas of low fitness because there are few peaks to cover these areas. As a result, the average solution quality is lower, and robust solutions can lose their quality more quickly. By increasing the number of peaks, the average survival time of solutions increases because peaks are likely to overlap and support robust solutions. For ASC, when the number of peaks is higher, the number of reliable peaks is higher as well. Therefore, the density of peaks in the landscape is higher, so the possibility of having reliable peaks closer to the current solution is higher which leads to decrease in switching cost when ASC chooses a solution by Eq. (5.4).

According to Tables 5.3 to 5.6, all of the algorithms obtain better results in 2-dimensional problems than in 5-dimensional ones. In fact, in higher dimensions, the problem becomes more challenging for the optimizer, so the efficiency of finding and tracking peaks is decreased which leads to having worse results.

TABLE 5.6: Results obtained by Eq.(5.6) by TFTmPSO , RFTmPSO and ASC on 5-dimensional mMPBR with 20 peaks.

			5 Dimensional				
V	Alg.	Stats.	w=0.1	w=0.5	w=1	w=2	w=3
		Median	63.49	53.26	40.10	14.11	-11.85
	TFTmPSO	Mean	63.65	53.50	40.81	15.44	-9.94
		StdErr	0.13	0.36	0.68	1.35	2.02
		Median	53.86	51.75	49.04	43.70	38.53
40	RFTmPSO	Mean	53.76	51.59	48.88	43.46	38.05
		StdErr	0.17	0.20	0.34	0.68	1.03
		Median	62.66	56.86	55.27	53.40	49.45
	ASC	Mean	62.94	57.09	55.44	52.54	49.16
		StdErr	0.20	0.39	0.44	0.57	0.83
		Median	63.49	53.26	40.10	14.11	-11.85
	TFTmPSO	Mean	63.65	53.50	40.81	15.44	-9.94
		StdErr	0.13	0.36	0.68	1.35	2.02
		Median	56.23	52.92	49.29	41.55	33.97
45	RFTmPSO	Mean	56.13	53.19	49.51	42.15	34.80
		StdErr	0.16	0.23	0.41	0.82	1.23
		Median	62.71	58.69	56.39	52.15	48.46
	ASC	Mean	63.00	58.34	56.30	52.14	47.58
		StdErr	0.20	0.37	0.47	0.73	1.00
		Median	63.49	53.26	40.10	14.11	-11.85
	TFTmPSO	Mean	63.65	53.50	40.81	15.44	-9.94
		StdErr	0.13	0.36	0.68	1.35	2.02
		Median	58.24	54.44	50.02	41.15	31.94
50	RFTmPSO	Mean	58.23	54.33	49.46	39.71	29.96
		StdErr	0.08	0.26	0.58	1.24	1.89
		Median	62.97	59.29	56.16	50.44	43.56
	ASC	Mean	63.08	59.27	56.10	49.53	42.60
		StdErr	0.19	0.36	0.60	1.09	1.61

5.3 Summary

SC is an important aspect of dynamic optimization problems (DOPs); however, there are few works considering the switching cost during the optimization process. Most investigations in dynamic optimization literature have been focused on tracking moving optima (TMO), which is usually pursued irrespective of the switching cost. Robust optimization over time (ROOT) addresses this shortcoming by keeping solutions as long as they remain acceptable. However, ROOT methods are not suitable for problems with smaller switching cost.

In this chapter, an adaptive solution chooser (ASC) algorithm for dynamic optimization problems with switching costs was proposed. ASC behaves in a similar way to TMO based algorithms where the switching cost is low and similar to ROOT based algorithms when the switching cost is high. ASC's core is a multi-swarm method which tracks peaks and calculates fitness variance of peaks that is used for determining reliable peaks in terms of robustness of solutions. ASC decides if a new solution is to be chosen or the previous one can be kept based on the current solution's fitness values, the fitness of other found solutions with better quality and their switching cost from the current solution. The experimental results obtained by a proposed performance indicator on modified moving peaks benchmark showed that ASC performed significantly better than two state-of-the-art methods for TMO and ROOT in problems with different levels of switching cost.

In fact, by proposing ASC, it is tried to bridge a gap between academic research and real-world problems in the field of DOPs. In contrary to TMO and ROOT which are addressing two extreme cases i.e. when switching cost is very small or very large, ASC makes a decision about changing or keeping solutions according to the switching cost at any range.

Chapter 6

Scaling up dynamic optimization problems: a divide-and-conquer approach

Motivated by rapid technological advancements, large-scale optimization has gained popularity in recent years. However, the exponential growth in the size of the search space, with respect to an increase in the number of the decision variables, has made large-scale optimization a challenging task. For DOPs, however, the challenge is twofold. For such problems, not only should an algorithm be capable of finding the global optimum in the vastness of the search space but should also be able to track it over time. For multi-modal DOPs, where several optima have the potential to turn into the global optimum after environmental changes, the cost of tracking multiple moving optima also adds to the complexity.

In this chapter, the large-scale global optimization in dynamic optimization problems is investigated. Moreover, a decomposition-based algorithm for large-scale dynamic optimization problems is proposed. The idea is to first discover and exploit the underlying structure of a given problem by decomposing it into several components of smaller size, and then to tackle the sub-problems simultaneously. The former can be achieved by a wide range of variable interaction analysis algorithms capable of identifying the underlying structure of a black-box problem with high efficiency and accuracy (Omidvar et al., 2014a, 2017; Mei et al., 2016; Sun et al., 2017), and the latter can be achieved by means of cooperative coevolution (Potter and Jong, 2000; Yang et al., 2008a; Li and Yao, 2012).

As described previously, MPB (Branke, 1999) is the most popular benchmark in the field of DOPs. In this chapter, first, the standard MPB is formally analyzed and it is shown that its lack of modularity limits its applicability for the study of large-scale DOPs. Then a new benchmark by composing several MPBs is proposed. The proposed benchmark is suitable for generating problem instances in which the components are heterogeneous in terms of dimension and their contribution to the fitness function value. In addition to the new benchmark, a new algorithm for large-scale DOPs is proposed. The proposed algorithm utilizes the state-of-the-art DG2 proposed by Omidvar et al. (2017) as the decomposition method. After determining variable interactions and components by DG2, the proposed algorithm uses a swarm for each detected component that works in a CC manner with other components' swarm. Each swarm consists of several sub-swarms to track multiple moving optima in the component's search space. Finally, the proposed method benefits from a new resource allocation approach in which it tries to prevent over exploitation by sub-swarms, and allocates more computational resource to the best sub-swarm of each component and the swarm of the component with the highest progress. All four algorithms are empirically evaluated on a wide range of problem settings to show the individual impact of approaches such as CC, tracking multiple moving optima and resource allocation on improving performance.

6.1 The Proposed benchmark generator

MPB generates a landscape containing several peaks whose height, width, and location change over time. As a result, each peak can become the global optimum after an environmental change according to its current height and width. Although MPB can be scaled to any number of dimensions, its lack of modularity limits its capacity for large-scale DOPs. This limitation comes from the nonseparable nature of the benchmark's baseline shown by Eq. (3.10).

Proposition 6.1. *An n-dimensional MPB with $m > 1$ peaks is nonseparable.*

Proof. Let,

$$f^{(t)}(\mathbf{x}) = \max\left\{\xi_1^{(t)}(\mathbf{x}), \psi^{(t)}(\mathbf{x})\right\}, \tag{6.1}$$

where

$$\psi^{(t)}(\mathbf{x}) = \max\left\{\xi_2^{(t)}(\mathbf{x}), \dots, \xi_m^{(t)}(\mathbf{x})\right\}. \tag{6.2}$$

The $\max(\cdot)$ function can be rewritten as follows:

$$f^{(t)}(\mathbf{x}) = \frac{1}{2}\left(\xi_1^{(t)}(\mathbf{x}) + \psi^{(t)}(\mathbf{x}) + |\xi_1^{(t)}(\mathbf{x}) - \psi^{(t)}(\mathbf{x})|\right). \tag{6.3}$$

Now, the first and second order partial derivative of $f^{(t)}(\mathbf{x})$ is as follows:

$$\frac{\partial f^{(t)}(\mathbf{x})}{\partial x_i} = \frac{1}{2}\left[\frac{\partial \xi_1^{(t)}(\mathbf{x})}{\partial x_i} + \frac{\partial \psi^{(t)}(\mathbf{x})}{\partial x_i} + \left(\frac{\partial \xi_1^{(t)}(\mathbf{x})}{\partial x_i} - \frac{\partial \psi^{(t)}(\mathbf{x})}{\partial x_i}\right) \operatorname{sgn}\left(\xi_1^{(t)}(\mathbf{x}) - \psi^{(t)}(\mathbf{x})\right)\right], \quad (6.4)$$

$$\frac{\partial^2 f^{(t)}(\mathbf{x})}{\partial x_i \partial x_j} = \frac{1}{2}\left[\frac{\partial^2 \xi_1^{(t)}(\mathbf{x})}{\partial x_i \partial x_j} + \frac{\partial^2 \psi^{(t)}(\mathbf{x})}{\partial x_i \partial x_j} + \left(\frac{\partial^2 \xi_1^{(t)}(\mathbf{x})}{\partial x_i \partial x_j} - \frac{\partial^2 \psi^{(t)}(\mathbf{x})}{\partial x_i \partial x_j}\right) \operatorname{sgn}\left(\xi_1^{(t)}(\mathbf{x}) - \psi^{(t)}(\mathbf{x})\right)\right], \quad (6.5)$$

where $\operatorname{sgn}(x) = \frac{x}{|x|}$ is the sign function.

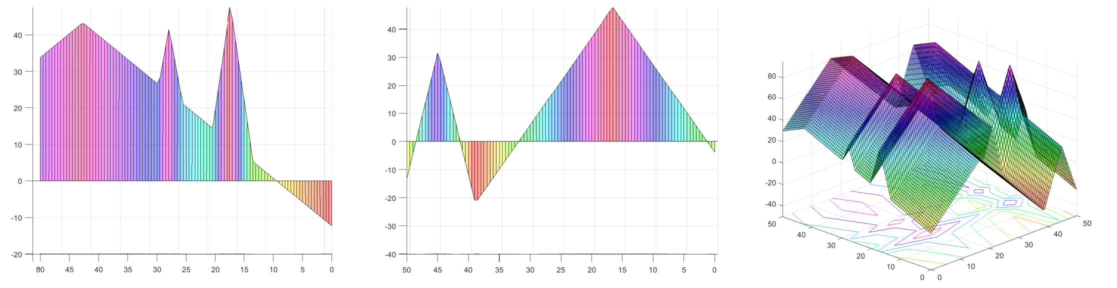
It is clear that $\frac{\partial f^{(t)}(\mathbf{x})}{\partial x_i}$ is either a function of $\frac{\partial \xi_1^{(t)}(\mathbf{x})}{\partial x_i}$ or $\frac{\partial \psi^{(t)}(\mathbf{x})}{\partial x_i}$ depending on whether $\xi_1^{(t)}(\mathbf{x}) > \psi^{(t)}(\mathbf{x})$ for a given value of x_i. In other words, for $\frac{\partial f^{(t)}(\mathbf{x})}{\partial x_i}$ to be consistently a function of $\frac{\partial \xi_1^{(t)}(\mathbf{x})}{\partial x_i}$ or $\frac{\partial \psi^{(t)}(\mathbf{x})}{\partial x_i}$, $\xi_1^{(t)}(\mathbf{x})$ must be strictly smaller or larger than $\psi^{(t)}(\mathbf{x})$ for every x_i. This essentially reduces $f^{(t)}(\mathbf{x})$ to a single peak MPB, which is clearly not the case simply because the height, the width, and the center of each peak is different. Therefore, the extremum with respect to the ith dimension cannot be uniquely determined by x_i. It is also clear that the second order partial derivative for arbitrary choices of i and j $(i \neq j)$ can be made nonzero for various choices of $\mathbf{x}$ due to the fact that the width, the height, and the center of each peak $(\xi_i^{(t)})$ is different. This makes every dimension interact with every other dimension (Def. 2.1). Therefore, a multi-modal MPB is fully nonseparable (Def. 2.4). □

Proposition 6.2. *An n-dimensional MPB with a single peak ($m = 1$) is additively nonseparable.*

Lemma 6.3 (necessary condition of additive separability)**.** *Given an additively separable function $f(\mathbf{x})$ (Def. 2.3), for arbitrary choices of x_i and x_j belonging to different component functions f_p and f_q, $\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$ is equal to zero.*

Proof. Assuming that x_i belongs to the component function f_p and x_j belongs to f_q, according to Def. 2.3, $\frac{\partial f}{\partial x_i} = \frac{f_p}{\partial x_i}$. Therefore, $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f_p}{\partial x_i \partial x_j} = 0$ because f_p is not a function of x_j. □

Proof of Proposition 6.2. Let the following be the definition of an n-dimensional single-peak MPB.



(a) 1-dimensional MPB with 3 peaks (b) 1-dimensional MPB with 2 peaks (c) 2-dimensional CMPB made by composing (a) and (b) resulting in a total of 6 peaks.

FIGURE 6.1: An example of exponentially growing number of peaks by composing MPBs.

$$f^{(t)}(\mathbf{x}) = h^{(t)} - w^{(t)}\|\mathbf{x} - \mathbf{c}^{(t)}\| \tag{6.6}$$

$$\frac{\partial f^{(t)}(\mathbf{x})}{\partial x_i} = \frac{-w^{(t)}\left(x_i - c_i^{(t)}\right)}{\|\mathbf{x} - \mathbf{c}^{(t)}\|} \tag{6.7}$$

$$\frac{\partial^2 f^{(t)}(\mathbf{x})}{\partial x_i \partial x_j} = \frac{w^{(t)}\left(x_i - c_i^{(t)}\right)\left(x_j - c_j^{(t)}\right)}{\|\mathbf{x} - \mathbf{c}^{(t)}\|^3} \tag{6.8}$$

It is clear that $\frac{\partial^2 f}{\partial x_i \partial x_j}$ is a function of both x_i and x_j and is nonzero as long as $x_i \neq c_i$ and $x_j \neq c_j$. Therefore, according to Lemma 6.3 the necessary condition for additive separability does not hold. Therefore, an n-dimensional MPB with a single peak is not additively separable. □

The following discussion clarifies why a single-peak MPB is easy to optimize despite its additive nonseparability feature. It is clear that Eq. (6.7) can be written as $g(x_i)h(\mathbf{x})$ where $g(x_i) = -w^{(t)}(x_i - c_i^{(t)})$, and $h(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}\|^{-1}$. To set Eq. (6.7) to zero, it is sufficient to force $g(x_i)$ to zero by forcing $x_i = c_i^{(t)}$. This is precisely why according to Def. 2.2 an n-dimensional MPB with a single peak is fully separable. Another way of looking at this problem is to realize that the square root function, implicit in the calculation of the Euclidean norm in the MPB formulation, is a monotonic function which does not change the location of the global optimum; however its presence removes additive separability. This observation is also empirically verified with DG2. It should be noted that this analysis is independent of environmental changes. In other words, MPB is additively nonseparable and remains so across all environments.

One way of modularizing MPB is through summation of several independent MPBs. This is customary in many large-scale global optimization benchmarks (Tang et al., 2009; Li et al., 2013) and has been recently used by Luo et al. (2017) to propose a modularized MPB. Three major shortcomings of this benchmark are: a lack of imbalance among components, uniform component sizes, and unrealistic homogeneous structures. Many real-world problems, however, are heterogeneous in nature which is caused by the coexistence of separable and nonseparable components, each having a different share in improving the objective function (Omidvar et al., 2015).

In this chapter, these shortcomings are addressed by proposing a new scalable benchmark, Composite MPB (CMPB), through heterogeneous composition of several MPBs. CMPB uses the standard MPB (Eq. (3.10)) as its component function and has the following general form:

$$F^{(t)}(\mathbf{x}) = \sum_{i=1}^{k} \left(\omega_i f_i^{(t)}(\mathbf{x}_i)\right) + \sum_{j=k+1}^{k+l} \left(\omega_j \gamma f_j^{(t)}(x_j)\right), \tag{6.9}$$

where the first summation term generates k nonseparable components, and the second summation term generates an l-dimensional separable component. Here f_i is the ith nonseparable subfunction which is a d_i-dimensional MPB ($d_i > 1$), f_j is the jth 1-dimensional MPB, $\mathbf{x}$ is the decision vector of D dimensions, $\mathbf{x}_i$ is a disjoint sub-vector of $\mathbf{x}$ with $d_i \geq 2$, x_j is a 1-dimensional scalar variable, ω_i and ω_j control the contribution of each component (for generating imbalance), and γ is a regulatory factor controlling the dominance of the separable component which is the reciprocal of the average dimensionality of the nonseparable components:

$$\gamma = \frac{k}{\sum_{i=1}^{k} d_i}. \tag{6.10}$$

According to Eq. (3.10), the contribution of various MPBs is almost identical. This is because the height and the width parameters are usually sampled from the same distribution for different instances of MPB and the use of the max function also dampens the contrasts between various instances of MPB. Therefore, in Eq. (6.9) a large number of separable variables can easily dominate the final function value, $F^{(t)}$, which limits the utility of the benchmark to study a wide range of scenarios. To alleviate this issue, γ is used to regulate the dominance of one component over another. As can be seen, γ is a function of k and d_i and is calculated automatically when the number of nonseparable components and their dimensions are chosen. It is only after this regularization that the

Table 6.1: Parameter settings of CMPB

Parameter	Symbol	Value
Number of peaks	m	Randomized between 1 to 10
Dimension	D	1-100
Evaluations between changes	f	$500D$
Shift severity	s	Randomized $\in [0.5, 3]$
Height severity	α	Randomized $\in [3, 10]$
Width severity	β	Randomized $\in [0.5, 1.5]$
Peaks shape	–	Cone
Peaks location range	SR	[-50,50]
Peak height	h	[30,70]
Peak width	w	[1,12]
Initial height value	–	50
Initial width value	–	6
Number of environments	–	100
Weight	ω	Randomized $\in [0.5, 2]$
Correlation coefficient	λ	0

imbalance coefficients (ω_i and ω_j) make intuitive sense and can be freely picked by the user to generate different imbalance patterns.

For each MPB in the CMPB, the height, width, and center of a peak change from one environment to the next similar to the MPB which is described in Subsection 4.2.2. The parameter settings of CMPB are shown in Table 6.1.

An interesting and natural consequence of CMPB's design is the exponential growth in the total number of peaks as the number of multi-modal components increases. This is a new challenge never addressed in either large-scale global optimization or dynamic optimization. In CMPB, when several MPBs are composed according to Eq. (6.9), the number of peaks is calculated as follows:

$$M = \prod_{i=1}^{k+l} m_i, \tag{6.11}$$

where m_i is the number of peaks in the ith MPB subfunction (represented by f_i and f_j in Eq. (6.9)). It should be noted that M is the maximum number of peaks that can exist in the landscape, which may change over time due to coverage of smaller peaks by larger ones. For the sake of clarity, an illustration is provided. In Fig. 6.1(a) and Fig. 6.1(b), two 1-dimensional MPBs with 2 and 3 peaks are shown. The 2-dimensional function constructed based on Eq. (6.9) with $\omega_1 = \omega_2 = 1$ results in a total of $2 \times 3 = 6$ peaks. A consequence of this is that even for low-dimensional functions of this form, variable interaction analysis and problem decomposition can significantly simplify the problem. Indeed, an ideal decomposition can reduce the maximum number of peaks to monitor down to $\sum_{i=1}^{k+l} m_i$ which is significantly smaller than Eq. (6.11) for problems with large number of peaks and components. In the next section a decomposition-based framework is proposed that has this feature.

6.2 The Proposed algorithm

In this section, a cooperative coevolutionary multi-swarm PSO (CCMPSO) is proposed for solving large-scale DOPs. First, an overview of the framework with an emphasis on its high-level structure and the resource allocation policy (Section 6.2.1) is provided. Then, the details of the multi-swarm optimizer is provided and dynamic issues such as convergence detection of swarms, avoiding mutual convergence of swarms onto the same peak, diversity control, and detection and handling of environmental changes (Section 6.2.2) are addressed.

6.2.1 The framework

Algorithm 5 shows the structure of the proposed CCMPSO. The algorithm has three major parts – decomposition, search and resource allocation, and change management – which are explained next.

6.2.1.1 Decomposition

The algorithm starts by decomposing a given dynamic optimization problem into its constituent independent components (Algorithm 5, line 1). This is done using a variable interaction analysis algorithm. In this chapter, the state-of-the-art DG2 algorithm (Omidvar et al., 2017) introduced in Subsection 2.9 is utilized. After problem decomposition, a multi-swarm dynamic optimizer is initialized for each of the identified components (Algorithm 5, lines 2-3). It should be noted that each component contains partial solutions which cannot be evaluated directly using the objective function. Due to the black-box nature of the objective function, these partial solutions can only be evaluated within the context of a complete solution. This complete solution is called a *context vector* (van den Bergh and Engelbrecht, 2004) which is randomly initialized on line 4.

Next, the algorithm enters its main loop and optimizes the identified lower-dimensional components in an iterative manner (Algorithm 5, lines 5-41). The algorithm has three major phases: 1) exploration, 2) exploitation, and 3) change management. In the first phase (Algorithm 5, lines 6-23), the algorithm cycles over all components with the aim of tracking optima, discovering any emerging optima, and estimating the contribution of each component in improving the overall objective function value. For this purpose, the algorithm maintains a free swarm and a set of tracker swarms for each component. The primary purpose of a free swarm is to find uncovered peaks. When a free swarm is converged to a peak, it will change to a tracker swarm whose primary purpose is to do exploitation and track it after each environmental change. For better use of the limited computational resources between successive environmental changes, the algorithm detects and deactivates the converged tracker swarms based on a mechanism which will be explained in the next section.

6.2.1.2 Search and resource allocation

The exploitation phase (Algorithm 5, lines 24-29) is a crucial step in improving the efficiency of the algorithm. First, the tracking of multiple moving optima is inherently expensive. Second, the contribution of components is not uniform, making a classic round-robin optimization policy very inefficient. The imbalance in the contribution happens for several reasons. Two major factors are nonuniform change severity of components after an environmental change, and discrepancy in the convergence behavior of swarms.

For the best use of the available resources, the exploitation phase occurs at two levels: component level, and swarm level. At the component level, the best contributing component is selected and all its active tracker swarms are executed for an extra iteration (Algorithm 5, lines 24-26). The amount by which each component improves the objective value at the end of the exploration phase is taken as its contribution. This often happens for the component experiencing the most intense environmental change. Therefore, by allocating more computational resource to such swarms, the algorithm accelerates the optimization process by prioritizing components with higher importance or higher change severities. Finally, at the swarm level, the best tracker of each component is executed for one more iteration (Algorithm 5, lines 27-29). This step not only gives more resources to the best performing tracker, but also keeps the information about the best partial solutions up-to-date for the purpose of updating the context vector.

6.2.1.3 Change management

Finally, the last phase deals with the environmental changes and updating of the context vector. Two events trigger the updating of the context vector. The first and the most obvious case is the detection of an environmental change (Algorithm 5, lines 30-37). The second is prior to the deployment of a solution (Algorithm 5, lines 38-40). In DOPs, the algorithm is given a predefined time frame within which it has to respond to an environmental change. This is denoted with η which is the maximum number of function evaluations available to the optimizer before providing a solution for deployment. It should be noted that in classic CC, the context vector is updated at every coevolutionary cycle. This is a costly operation because all solutions whose fitness were calculated with a previous version of the context vector have to be re-evaluated. However, owing to the grouping accuracy of DG2 and the independent nature of the components, this operation can be delayed until it becomes necessary (due to a dynamic change).

6.2.2 Dynamic considerations

The aim of the algorithm is to find all peaks and track them. However, due to the lack of information about the number of peaks, and also the coverage of some smaller peaks by larger ones in some of the environments, a free swarm needs to constantly search for possible uncovered peaks. Once a new optimum is found by a free swarm, it changes to a tracker swarm. To test the convergence of a free swarm, the procedure proposed by Blackwell et al. (2008) is used in which the Euclidean distances between all pairs of particles are calculated. If all calculated distances are smaller than a given threshold (r_{conv}), the algorithm assumes that the free swarm is converged. When a free swarm becomes a tracker swarm, a new free swarm will be initialized immediately in the search space in order to search for another uncovered peak. It is possible that a free swarm converges to a peak already covered by a tracker swarm. Tracking a peak by multiple swarms wastes a considerable amount of computational resource. Therefore, a mutual exclusion principle is enforced to avoid more than one swarm to cover the same peak. To establish the mutual exclusion, the mechanism proposed by Blackwell and Branke (2006) is utilized. According to the exclusion mechanism, when Euclidean distances between the global best of the free swarm and a tracker swarm is less than a threshold (r_{excl}), the algorithm assumes that the free swarm has converged to a covered peak. In this situation, the free swarm will be re-initialized. The value of r_{excl} is calculate as follows:

$$r_{\text{excl}} = 0.5\frac{\text{SR}}{\sqrt[D]{\text{TSN}}}, \tag{6.12}$$

where SR is the range of search space and TSN is the number of tracker swarms.

A similar conflict can also happen to two tracker swarms. This situation happens when a peak is covered by a larger peak. Therefore, its tracker swarm loses its own peak and starts converging to the larger peak's center. A similar situation happens when the free swarm convergence is detected before it enters into the mutual exclusion area of a covered peak. As a result, it becomes a tracker swarm and moves toward the peak's center. This

```
Algorithm 5: CCMPSO
1  $\mathcal{G} = \texttt{Grouping}(f)$;
2  forall $\mathcal{G}$ do
3      $\mathbf{P}_{\text{free}} \leftarrow$ Initialize the free swarm's population;
4  $\mathbf{c} \leftarrow$ Randomly initialize the context vector;
5  repeat
6      forall $\mathcal{G}$ do
7          $(\mathbf{P}_{\text{free}}, \mathbf{g}^{\star}_{\text{free}}, \mathbf{p}^{\star}_{\text{free}}) = \texttt{PSO}(\mathbf{P}_{\text{free}}, \mathbf{g}^{\star}_{\text{free}}, \mathbf{p}^{\star}_{\text{free}})$;
8          if diversity of the free swarm is $< r_{\text{conv}}$ then
9              Change its status to tracker swarm;
10             $\mathbf{P}_{\text{free}} \leftarrow$ Initialize a new free swarm;
11         foreach tracker swarm $i$ do
12             if $\|\mathbf{g}^{\star}_{\text{free}} - \mathbf{g}^{\star}_{i}\| < r_{\text{excl}}$ then
13                 $\mathbf{P}_{\text{free}} \leftarrow$ Reinitialize the free swarm;
14             if $i^{th}$ tracker swarm is active then
15                 $(\mathbf{P}_i, \mathbf{g}^{\star}_i, \mathbf{p}^{\star}_i) = \texttt{PSO}(\mathbf{P}_i, \mathbf{g}^{\star}_i, \mathbf{p}^{\star}_i)$;
16                 if the diversity is $< r_{\text{deact}}$ then
17                     Deactivate the tracker swarm;
18             foreach tracker swarms $j$ do
19                 if $\|\mathbf{g}^{\star}_i - \mathbf{g}^{\star}_j\| < r_{\text{excl}}$ then
20                     if $f(\mathbf{g}^{\star}_i) < f(\mathbf{g}^{\star}_j)$ then
21                         Remove $i^{\text{th}}$ tracker swarm;
22                     else if $f(\mathbf{g}^{\star}_i) > f(\mathbf{g}^{\star}_j)$ then
23                         Remove $j^{\text{th}}$ tracker swarm ;
24     Determine the component $H$ with the highest progress;
25     forall active tracker swarms $i$ in $H$ do
26         $(\mathbf{P}_i, \mathbf{g}^{\star}_i, \mathbf{p}^{\star}_i) = \texttt{PSO}(\mathbf{P}_i, \mathbf{g}^{\star}_i, \mathbf{p}^{\star}_i)$;
27     foreach $\mathcal{G}$ do
28         Determine the best tracker swarm $b$;
29         $(\mathbf{P}_b, \mathbf{g}^{\star}_b, \mathbf{p}^{\star}_b) = \texttt{PSO}(\mathbf{P}_b, \mathbf{g}^{\star}_b, \mathbf{p}^{\star}_b)$;
30     if an environmental change is happened then
31         $\mathbf{c} \leftarrow$ Update context vector using best found position in each swarm $\mathbf{g}^{\star}$;
32         forall $\mathcal{G}$ do
33             Re-evaluate all $\mathbf{p}^{\star}$ of free swarm;
34             forall tracker swarms do
35                 Update estimated shift severity by Eq. (4.1);
36                 Activate if is deactivated;
37                 Increase diversity by Eq. (6.13);
38     if computational budget $\eta$ is finished then
39         $\mathbf{c} \leftarrow$ Update context vector using best found position in each swarm $\mathbf{g}^{\star}$;
40         Re-evaluate all $\mathbf{p}^{\star}$ in all swarms;
41 until stopping criterion is met;
```

is another case where the exclusion principle is enforced to control the computational overhead. To do so, the tracker swarm with worse global best fitness value $f(\mathbf{g}^{\star})$ will be removed. For determining tracker swarms which are under the exclusion condition, the Euclidean distance between all pairs of tracker swarms' $\mathbf{g}^{\star}$ position is calculated and compared with r_{excl} based on Eq. (6.12).

Another critical challenge of the population-based optimization algorithms in DOPs is diversity loss. According to (Nguyen et al., 2012a), there are two main groups of methods to address this challenge. First is the reaction methods which introduce diversity after each environmental change, and second is diversity maintenance methods which try to keep the diversity of population above a certain level over time.

Nguyen et al. (2012a) categorized diversification mechanisms into two groups i.e. diversity maintenance and diversity introducing. Nguyen (2011) showed diversity maintenance methods decrease the exploitation ability and use additional computational resource for maintaining diversity. Moreover, diversity maintenance methods are suitable for DOPs in which the environmental changes are not detectable which is not in the scope of this thesis as described in 1.1. On the other hand, diversity introducing methods consume less computational resource and are suitable for DOPs in which changes are detectable (Blackwell et al., 2008). Therefore, the proposed multi-swarm PSO is a reaction type method in which the tracker swarms' diversities are increased at the beginning of each environment (diversity introducing). When a change is detected, for each tracker swarm, one of the particles is located on the $\mathbf{g}^{\star}$ position from the previous environment and other particles are randomized around the $\mathbf{g}^{\star}$ position with the radius of shift severity of the peak by Eq. (6.13):

$$\mathbf{P}_{i,j} = (s_i \cdot \mathbf{r}) + \mathbf{g}_i^{\star(t-1),\mathrm{end}}, \tag{6.13}$$

where $\mathbf{P}_{i,j}$ is the position of the jth particle of the ith tracker swarm and $\mathbf{g}_i^{\star(t-1),\mathrm{end}}$ is its global best position at the end of the previous environment, s_i is the shift severity of the peak which is under cover of the ith tracker swarm, and $\mathbf{r}$ is a uniformly distributed random number vector in range $[-1, 1]$. The reason for using s_i in Eq. (6.13) is that the new location of the peak after environmental change is expected to be inside that radius from the previous peak center. In Eq. (6.13), the $\mathbf{g}^{*}$ from the end of the previous environment is used instead of the previous peak center position. Therefore, the diversity is introduced to the population of each tracker swarm as much as needed. The shift severity of each peak is estimated by Eq. (4.1).

Another diversity related issue is detection and deactivation of converged tracker swarms to save computational resources. When a tracker swarm gets sufficiently close to the center of a peak, it should be deactivated until the next environment. A tracker swarm is deactivated when its diversity drops below a certain threshold. To measure the diversity of a tracker swarm the infinity norm distance between all pairs of particles is calculated. If all distances fall below a predefined value (r_{deact}), the algorithm deactivates the tracker swarm which means that its particles freeze until another environmental change is detected. r_{deact} is a positive constant number. A positive attribute of using infinity norm distance here is that it is independent from dimension number.

Another challenge of DOPs is outdated memory which happens after each environmental change due to the outdated stored fitness values of positions such as $\mathbf{g}^\star$ and personal best $\mathbf{p}^\star$. For addressing this issue, after each environmental change, the fitness values of all $\mathbf{p}^\star$ positions of the free swarm will be re-evaluated. For tracker swarms, after re-diversification, the fitness values of particle positions are evaluated and the $\mathbf{p}^\star$ positions are set to particle positions.

Finally, the last main challenge is change detection. Since detecting a change is a separate issue and in most real-world dynamic environments the occurrence of a change is obvious (e.g., arrival of new order, change in temperature) (Nguyen, 2011), like previous chapters in this thesis, it is assumed that the algorithm will be informed when an environmental change happens. However, it should be noted that environmental changes can be detected easily in many problems including the ones that is investigated in this research by re-evaluating some beacons (Nguyen et al., 2012a).

6.3 Experiments and analysis

The experiments in this section are based on different scenarios of CMPB framework described in Section 6.1. The statistical results are based on 31 independent runs and their median, mean, and standard error are reported for comparison. To test the statistical significance of the reported results, a multiple comparison test is performed based on a series of pairwise Wilcoxon signed-rank tests with Holm-Bonferroni p-value correction with $\alpha = 0.05$. Highlighted entries denote statistically significant results.

6.3.1 Comparison algorithms

The performance of the proposed Cooperative Coevolutionary Multi-swarm PSO (CCMPSO) is compared with three other methods. The main features of these algorithms are summarized in Table 6.2. As can be seen, each algorithm allows us to test the hypotheses about the efficacy of the proposed decomposition approach, resource allocation policy, and the dynamic multi-swarm strategy. The first algorithm uses a single multi-swarm PSO (SMsPSO) without any decomposition. The multi-swarm PSO approach in SMsPSO is the same as the one used in CCMPSO described in Section 6.2.2. The second algorithm is a CC-based method which uses DG2 to decompose the problem into its constituent components. This method uses a single-swarm PSO for each component resulting in multiple single swarms (MSsPSO). In MSsPSO, after each environmental change, each swarm is re-initialized except $\mathbf{g}^\star$ which is kept from the last environment. The third method is a multi multi-swarm PSO (MMsPSO), a CC-based algorithm which

TABLE 6.2: Summery of utilized approaches in the algorithms

Algorithm	Cooperative coevolutionary	Tracking multiple moving optima	Resource allocation
CCMPSO	✓	✓	✓
MMsPSO	✓	✓	×
MSsPSO	✓	×	×
SMsPSO	×	✓	×

uses a multi-swarm PSO for each component. MMsPSO is identical to CCMPSO except that it does not have any resource allocation mechanism and uses the round-robin allocation policy of the standard CC.

It should be noted that MMsPSO resembles the state-of-the-art GCM-PSO Luo et al. (2017). In fact, MMsPSO replicates major features of GCM-PSO but unifies the underlying decomposition methods and the multi-population peak tracking mechanism for a fair comparison with CCMPSO.

6.3.2 Performance indicator

To measure the efficiency of algorithms, the average error of the updated context vector at the time of deployment (determined by η) after each environmental change is used as the measure of performance:

$$P = \frac{1}{T}\sum_{t=1}^{T}\left(f^{(t)}\left(\text{Optimum}^{(t)}\right) - f^{(t)}\left(\mathbf{c}^{(t),\eta}\right)\right), \tag{6.14}$$

where $\mathbf{c}^{(t),\eta}$ is the context vector at the tth environment which is updated after η fitness evaluations since the beginning of the new environment.

6.3.3 Parameter settings

All algorithms use PSO with a constriction factor (Eberhart and Shi, 2001) as their core optimizer where $C_1 = C_2 = 2.05$ and $\chi = 0.729843788$. The value of r_{conv} for CCMPSO, SMsPSO, and MMsPSO is equal to r_{excl} as calculated by Eq. (6.12) based on (Blackwell et al., 2008). Additionally, the context vector in all algorithms is updated only after environmental changes and when the computational budget η is used. The default value of η is $f - 1$ which means the solution is fetched at the end of each environment.

To determine the appropriate population size for tracker and free swarms in MMsPSO, SMsPSO and CCMPSO, and the population size of swarms in MSsPSO, a sensitivity analysis is conducted, the result of which can be found in the supplementary document. According to Table 6.3 of the supplementary document, an appropriate swarm size for CCMPSO, SMsPSO, and MMsPSO is 5 for $d \leq 5$, 7 for $5 < d \leq 10$, and 10 for $d > 10$. For MSsPSO, a population size of 50 across all dimensions seems appropriate according to Table 6.4. Finally, according to Table 6.5, r_{deact} in CCMPSO is set to 0.1.

For determining appropriate population size for tracker and free swarms, the performance of the multi-swarm PSO described in Section 6.2.2 is measured for a single MPB with 5 peaks and different dimensions (all other parameters are set based on Table 6.1) and results are shown in Table 6.3. According to this table, it seems that choosing population size of 5 for $d \leq 5$, 7 for $5 < d \leq 10$ and 10 for $d > 10$ is good.

For MSsPSO, the same sensitivity analysis as Table 6.1 has been done and the results are shown in Table 6.4 for different population size of MSsPSO. According to the statistical analysis, results obtained by all the population sizes between 30 to 70 are not significantly different. However, according to median values, it seems population size of 50 and 60 are the best. 50 is chosen for population size of MSsPSO for the rest of experiments.

Finally, the sensitivity analysis for r_{deact} in CCMPSO is illustrated in Table 6.5 which is done on a single MPB with 5 peaks and different dimensions (all other parameters are set based on Table 6.1). According to this table, CCMPSO's performance is not sensitive to different values of this parameter. For the rest of experiments, since $r_{\text{deact}} = 0.1$ is in the middle of the highlighted values for almost all dimensions, it is chosen for the experiments.

6.3.4 Empirical analysis

To compare the performance of the four algorithms,they are tested on 20 instance functions with various characteristics created using the CMPB benchmark generator. The suite contains functions with five different variable interaction structures tested in 25-, 50-, 100-, and 200-dimensional spaces (Table 6.6). This section contains two sets of experiments. The first set is concerned with investigating the efficacy of decomposition and resource allocation in CCMPSO (Section 6.3.4.1), and the second set is concerned with investigating the robustness of the algorithm with respect to various aspects of DOPs, such as the number of peaks, shift severities, and change frequencies (Section 6.3.4.2).

6.3.4.1 The overall comparison

For the experiments in this section, the dynamic parameters of the functions listed in Table 6.6 are set according to the default values reported in Table 6.1. The results

TABLE 6.3: Sensitivity analysis on different sub-swarm's population size of the multi-swarm PSO from Section 6.2.2 (which is used in CCMPSO, MMsPSO and SMsPSO) for optimizing MPBs with $m = 5$, different dimensions and other parameters from Table 6.1

Dimension	stats.	Population Size							
		3	4	5	7	10	15	20	25
1	Median	5.55e-02	5.96e-02	5.53e-02	7.77e-02	1.24e-01	1.63e-01	2.36e-01	3.22e-01
	Mean	7.27e-02	6.27e-02	6.87e-02	8.55e-02	1.42e-01	1.99e-01	3.65e-01	3.76e-01
	StErr	8.54e-03	4.93e-03	6.94e-03	1.09e-02	1.62e-02	2.88e-02	7.25e-02	5.05e-02
2	Median	2.07e-01	1.33e-01	1.48e-01	2.45e-01	3.86e-01	5.02e-01	7.26e-01	8.54e-01
	Mean	3.22e-01	2.04e-01	2.84e-01	3.61e-01	4.81e-01	6.15e-01	8.66e-01	9.15e-01
	StErr	4.76e-02	3.56e-02	7.88e-02	5.84e-02	4.89e-02	5.62e-02	1.03e-01	8.69e-02
3	Median	3.69e-01	1.56e-01	1.48e-01	3.02e-01	3.79e-01	6.56e-01	1.05e+00	1.26e+00
	Mean	4.83e-01	3.01e-01	4.32e-01	5.10e-01	6.83e-01	8.05e-01	1.24e+00	1.41e+00
	StErr	6.49e-02	7.28e-02	1.24e-01	9.57e-02	1.55e-01	8.33e-02	1.30e-01	1.55e-01
5	Median	1.21e+00	3.45e-01	1.56e-01	2.27e-01	4.20e-01	7.51e-01	1.24e+00	1.61e+00
	Mean	1.38e+00	5.72e-01	3.03e-01	4.21e-01	5.59e-01	9.16e-01	1.53e+00	1.64e+00
	StErr	1.04e-01	1.32e-01	6.91e-02	8.30e-02	8.23e-02	8.28e-02	1.57e-01	9.52e-02
7	Median	2.96e+00	8.08e-01	3.63e-01	2.20e-01	4.76e-01	8.04e-01	1.17e+00	1.78e+00
	Mean	2.80e+00	9.22e-01	4.24e-01	4.77e-01	5.10e-01	9.35e-01	1.37e+00	1.89e+00
	StErr	1.49e-01	6.88e-02	3.66e-02	1.87e-01	5.15e-02	8.38e-02	1.15e-01	1.51e-01
10	Median	6.01e+00	2.24e+00	8.83e-01	6.12e-01	6.24e-01	1.11e+00	1.62e+00	1.79e+00
	Mean	5.89e+00	2.30e+00	1.04e+00	6.90e-01	7.43e-01	1.51e+00	1.74e+00	1.96e+00
	StErr	2.35e-01	1.20e-01	8.06e-02	7.02e-02	9.15e-02	2.48e-01	1.25e-01	1.64e-01
15	Median	1.29e+01	6.04e+00	3.16e+00	1.50e+00	8.72e-01	1.37e+00	1.66e+00	2.21e+00
	Mean	1.34e+01	6.39e+00	3.55e+00	1.82e+00	1.36e+00	1.45e+00	1.89e+00	2.70e+00
	StErr	5.53e-01	3.03e-01	2.59e-01	2.95e-01	2.90e-01	1.51e-01	1.55e-01	2.71e-01
20	Median	2.23e+01	1.12e+01	7.00e+00	3.61e+00	1.67e+00	2.05e+00	2.27e+00	2.92e+00
	Mean	2.27e+01	1.12e+01	7.42e+00	3.77e+00	2.15e+00	2.33e+00	3.25e+00	3.44e+00
	StErr	6.72e-01	3.73e-01	5.43e-01	2.89e-01	2.13e-01	2.68e-01	4.49e-01	3.15e-01

Table 6.4: Sensitivity analysis on different swarm's population size of the MSsPSO for optimizing MPBs with $m = 5$, different dimensions and other parameters from Table 6.1

Dimension	stats.	Population Size						
		10	20	30	40	50	60	70
1	Median	7.78e+00	1.01e+01	7.78e+00	6.84e+00	6.44e+00	6.56e+00	6.99e+00
	Mean	8.38e+00	9.06e+00	9.25e+00	7.78e+00	7.40e+00	7.14e+00	7.80e+00
	StErr	5.35e-01	5.11e-01	9.52e-01	7.50e-01	7.21e-01	5.77e-01	6.12e-01
2	Median	1.20e+01	1.12e+01	1.22e+01	1.23e+01	1.34e+01	1.18e+01	1.33e+01
	Mean	1.24e+01	1.26e+01	1.23e+01	1.25e+01	1.40e+01	1.22e+01	1.22e+01
	StErr	8.86e-01	1.09e+00	6.94e-01	7.74e-01	8.81e-01	5.27e-01	8.05e-01
3	Median	1.67e+01	1.31e+01	1.40e+01	1.36e+01	1.20e+01	1.31e+01	1.12e+01
	Mean	2.01e+01	1.30e+01	1.51e+01	1.35e+01	1.23e+01	1.32e+01	1.12e+01
	StErr	1.87e+00	5.01e-01	1.13e+00	8.29e-01	6.21e-01	7.02e-01	5.44e-01
5	Median	3.57e+01	1.24e+01	1.44e+01	1.37e+01	1.20e+01	1.34e+01	1.52e+01
	Mean	3.76e+01	1.44e+01	1.44e+01	1.58e+01	1.28e+01	1.40e+01	1.49e+01
	StErr	1.94e+00	1.35e+00	6.75e-01	1.26e+00	6.84e-01	5.42e-01	8.68e-01
7	Median	4.40e+01	1.64e+01	1.20e+01	1.29e+01	1.44e+01	1.25e+01	1.28e+01
	Mean	6.24e+01	1.75e+01	1.38e+01	1.33e+01	1.43e+01	1.40e+01	1.34e+01
	StErr	8.26e+00	1.24e+00	1.13e+00	8.22e-01	8.03e-01	9.27e-01	8.15e-01
10	Median	4.73e+01	1.60e+01	1.58e+01	1.42e+01	1.41e+01	1.49e+01	1.46e+01
	Mean	5.34e+01	1.70e+01	2.17e+01	1.76e+01	1.47e+01	1.56e+01	1.62e+01
	StErr	3.61e+00	1.30e+00	3.95e+00	2.41e+00	8.41e-01	8.54e-01	1.41e+00
15	Median	5.09e+01	1.49e+01	1.51e+01	1.62e+01	1.35e+01	1.71e+01	1.39e+01
	Mean	6.31e+01	2.16e+01	1.73e+01	1.91e+01	1.69e+01	1.65e+01	2.17e+01
	StErr	8.94e+00	5.15e+00	2.09e+00	2.20e+00	2.91e+00	1.18e+00	5.57e+00
20	Median	5.63e+01	1.92e+01	1.91e+01	1.58e+01	1.64e+01	1.40e+01	1.47e+01
	Mean	7.61e+01	4.71e+01	4.43e+01	2.46e+01	2.86e+01	1.67e+01	2.34e+01
	StErr	1.14e+01	1.26e+01	8.54e+00	4.15e+00	8.62e+00	2.14e+00	4.63e+00

Table 6.5: Sensitivity analysis on different r_{deact} in CCMPSO for optimizing MPBs with $m = 5$, different dimensions and other parameters from Table 6.1.

Dimension	stats.	r_{deact}							
		0.01	0.02	0.05	0.1	0.2	0.5	1	2
1	Median	6.21e-02	8.67e-02	1.09e-01	8.51e-02	1.09e-01	1.43e-01	3.03e-01	5.02e-01
	Mean	1.19e-01	1.63e-01	1.61e-01	1.43e-01	1.53e-01	1.82e-01	3.57e-01	5.75e-01
	StErr	2.45e-02	3.27e-02	2.97e-02	3.12e-02	2.98e-02	3.27e-02	4.94e-02	7.41e-02
2	Median	3.98e-01	2.96e-01	3.80e-01	2.21e-01	2.60e-01	2.20e-01	3.73e-01	7.43e-01
	Mean	6.29e-01	4.85e-01	5.50e-01	5.42e-01	4.82e-01	4.83e-01	5.82e-01	1.04e+00
	StErr	1.21e-01	9.76e-02	1.04e-01	1.47e-01	9.77e-02	1.19e-01	1.07e-01	1.85e-01
3	Median	2.45e-01	2.09e-01	1.31e-01	1.86e-01	1.99e-01	2.81e-01	4.78e-01	7.17e-01
	Mean	6.22e-01	1.21e+00	4.85e-01	6.76e-01	5.63e-01	7.10e-01	8.16e-01	1.02e+00
	StErr	2.02e-01	6.91e-01	1.71e-01	1.63e-01	1.44e-01	2.02e-01	2.52e-01	1.91e-01
5	Median	1.33e-01	1.48e-01	9.95e-02	1.22e-01	1.69e-01	1.83e-01	2.93e-01	9.46e-01
	Mean	4.82e-01	3.47e-01	3.44e-01	3.32e-01	3.28e-01	3.40e-01	4.41e-01	9.66e-01
	StErr	1.39e-01	9.10e-02	9.28e-02	9.75e-02	7.48e-02	7.76e-02	7.34e-02	1.29e-01
7	Median	2.48e-01	2.25e-01	2.40e-01	2.05e-01	2.04e-01	1.77e-01	2.64e-01	6.54e-01
	Mean	3.71e-01	4.16e-01	4.16e-01	3.53e-01	3.97e-01	2.73e-01	4.39e-01	7.65e-01
	StErr	7.44e-02	8.27e-02	8.17e-02	8.49e-02	8.05e-02	5.62e-02	7.72e-02	9.21e-02
10	Median	3.90e-01	3.38e-01	4.89e-01	3.33e-01	3.58e-01	4.01e-01	4.38e-01	1.03e+00
	Mean	8.26e-01	1.02e+00	6.41e-01	6.67e-01	7.35e-01	5.20e-01	5.97e-01	1.07e+00
	StErr	3.45e-01	3.46e-01	1.24e-01	1.49e-01	1.94e-01	9.73e-02	8.61e-02	1.29e-01
15	Median	9.96e-01	7.64e-01	8.79e-01	8.43e-01	9.38e-01	7.22e-01	7.95e-01	1.24e+00
	Mean	1.54e+00	1.37e+00	1.40e+00	1.49e+00	1.26e+00	9.14e-01	9.81e-01	1.74e+00
	StErr	2.51e-01	2.47e-01	2.79e-01	2.87e-01	1.84e-01	1.34e-01	1.27e-01	2.61e-01
20	Median	1.75e+00	1.52e+00	1.58e+00	1.68e+00	1.68e+00	1.37e+00	1.68e+00	2.10e+00
	Mean	2.41e+00	1.81e+00	2.03e+00	1.95e+00	2.61e+00	2.21e+00	1.94e+00	2.23e+00
	StErr	4.71e-01	2.48e-01	3.21e-01	2.36e-01	4.02e-01	4.43e-01	3.13e-01	2.37e-01

TABLE 6.6: Benchmark Scenarios Based on CMPB.

Function	D	Dimensionality of Nonseparable Components	Separable
f_1	25	$\{2, 4, 6, 8\}$	5
f_2	25	$\{2, 5\}$	18
f_3	25	$\{2, 4, 5, 6, 8\}$	0
f_4	25	—	25
f_5	25	$\{25\}$	0
f_6	50	$\{2, 3, 5, 6, 7, 8, 10\}$	10
f_7	50	$\{2, 3, 5, 5\}$	35
f_8	50	$\{2, 2, 3, 5, 5, 5, 5, 5, 8, 10\}$	0
f_9	50	—	50
f_{10}	50	$\{100\}$	0
f_{11}	100	$\{2, 2, 3, 5, 5, 6, 6, 8, 8, 10, 10, 15\}$	20
f_{12}	100	$\{2, 2, 3, 3, 5, 5, 10\}$	70
f_{13}	100	$\{2, 2, 2, 2, 3, 3, 5, 5, 5, 5, 5, 5, 8, 8, 10, 10, 20\}$	0
f_{14}	100	—	100
f_{15}	100	$\{100\}$	0
f_{16}	200	$\{2, 2, 3, 5, 5, 6, 6, 8, 8, 10, 10, 15, 20, 20, 30\}$	50
f_{17}	200	$\{2, 3, 5, 10, 20, 30\}$	130
f_{18}	200	$\{2, 2, 2, 3, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 20, 20, 30, 50\}$	0
f_{19}	200	—	200
f_{20}	200	$\{200\}$	0

obtained by CCMPSO, MMsPSO, SMsPSO, and MSsPSO on f_1 to f_{20} are summarized in Tables 6.7 and 6.8. These tables clearly show that CCMPSO performs significantly better than all other algorithms in the majority of the functions. Exceptions are the fully nonseparable functions (f_5, f_{10}, f_{15}, and f_{20}) for which no decomposition happens. It is notable that other decomposition-based algorithms, MMsPSO and MSsPSO, also perform better than SMsPSO which does not benefit from a decomposition mechanism. This clearly shows the benefit of problem decomposition for solving large-scale DOPs. Two reasons can be attributed to the poor performance of SMsPSO in the majority of the functions. First is the scalability issue. It is clear that in the absence of problem decomposition, the dimensionality of a given problem can easily exceed the capacity of the optimizer. Second, there is the exponential growth in the number of peaks when no decomposition is used (see Fig. 6.1).

In addition to problem decomposition, resource allocation is another major feature of CCMPSO. The effectiveness of CCMPSO's resource allocation mechanism can be checked by comparing it with MMsPSO whose only difference lies within its resource allocation policy. Tables 6.7 and 6.8 clearly show the superiority of CCMPSO over MMsPSO. Although problem decomposition plays a crucial role in simplifying a large-scale problem, the existence of numerous components can impose a computational overhead on

Table 6.7: Comparative results of CCMPSO, MMsPSO, MSsPSO, and SMsPSO on f_1 to f_{10}. The highlighted entries are significantly better using pair-wise Wilcoxon signed-rank test with Holm p-value adjustment ($\alpha = 0.05$).

Function	Stats.	CCMPSO	MMsPSO	MSsPSO	SMsPSO
	Median	2.35e+00	3.95e+00	5.93e+01	6.43e+01
f_1	Mean	2.59e+00	4.33e+00	5.94e+01	6.15e+01
	StErr	2.74e-01	3.56e-01	2.99e+00	3.18e+00
	Median	2.23e+00	3.24e+00	3.17e+01	9.55e+01
f_2	Mean	2.38e+00	4.00e+00	3.20e+01	9.72e+01
	StErr	1.95e-01	4.27e-01	1.90e+00	2.02e+00
	Median	1.19e+00	2.87e+00	5.71e+01	4.30e+01
f_3	Mean	1.67e+00	2.96e+00	5.89e+01	4.52e+01
	StErr	2.33e-01	2.40e-01	2.82e+00	2.67e+00
	Median	3.00e+00	3.19e+00	1.10e+01	3.21e+02
f_4	Mean	3.05e+00	3.23e+00	1.10e+01	3.17e+02
	StErr	2.13e-01	1.07e-01	2.43e-01	5.75e+00
	Median	1.84e+00	2.94e+00	1.38e+01	1.24e+00
f_5	Mean	2.34e+00	3.56e+00	1.49e+01	2.13e+00
	StErr	3.62e-01	3.87e-01	1.39e+00	4.14e-01
	Median	4.56e+00	8.77e+00	1.14e+02	1.77e+02
f_6	Mean	5.19e+00	9.09e+00	1.18e+02	1.79e+02
	StErr	4.33e-01	4.69e-01	3.96e+00	4.84e+00
	Median	4.42e+00	6.53e+00	6.68e+01	2.47e+02
f_7	Mean	4.44e+00	6.58e+00	6.62e+01	2.46e+02
	StErr	2.51e-01	3.43e-01	2.66e+00	3.63e+00
	Median	3.64e+00	5.95e+00	1.14e+02	2.07e+02
f_8	Mean	4.20e+00	6.22e+00	1.17e+02	1.96e+02
	StErr	3.47e-01	4.86e-01	4.23e+00	6.15e+00
	Median	6.08e+00	6.73e+00	2.17e+01	8.16e+02
f_9	Mean	6.03e+00	6.48e+00	2.18e+01	8.19e+02
	StErr	1.61e-01	1.26e-01	2.73e-01	1.11e+01
	Median	7.76e+00	8.38e+00	1.66e+01	8.05e+00
f_{10}	Mean	7.77e+00	9.67e+00	2.01e+01	8.22e+00
	StErr	6.11e-01	7.92e-01	2.66e+00	8.00e-01

the algorithm. Additionally, use of a multi-population algorithm to optimize the components also adds to the computational complexity. The component-level and swarm-level resource allocation policies of CCMPSO allow for an economical use of resources while preserving the simplifying effects of problem decomposition. The swarm-level mechanism prevents over-exploitation of tracker swarms and releases more resources to be used by the best trackers to improve the overall solution quality. The component-level mechanism accelerates the convergence by allocating more resources to the component

Table 6.8: Comparative results of CCMPSO, MMsPSO, MSsPSO, and SMsPSO on f_{11} to f_{20}. The highlighted entries are significantly better using pair-wise Wilcoxon signed-rank test with Holm p-value adjustment ($\alpha = 0.05$).

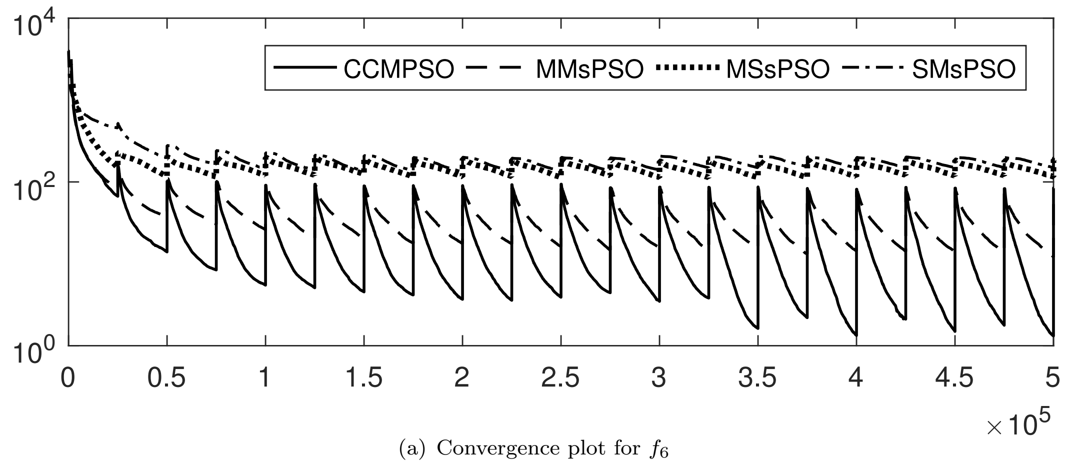
Function	Stats.	CCMPSO	MMsPSO	MSsPSO	SMsPSO
f_{11}	Median	1.05e+01	2.02e+01	2.13e+02	6.45e+01
	Mean	1.46e+01	2.06e+01	2.11e+02	6.14e+01
	StErr	3.10e+00	6.99e-01	5.06e+00	3.18e+00
f_{12}	Median	1.19e+01	1.51e+01	1.31e+02	5.71e+02
	Mean	1.44e+01	1.63e+01	1.30e+02	5.66e+02
	StErr	1.44e+00	1.34e+00	4.90e+00	5.46e+00
f_{13}	Median	1.05e+01	1.70e+01	1.98e+02	5.00e+02
	Mean	1.15e+01	1.79e+01	2.02e+02	4.96e+02
	StErr	1.04e+00	8.35e-01	4.67e+00	1.06e+01
f_{14}	Median	1.18e+01	1.32e+01	4.35e+01	2.14e+03
	Mean	1.19e+01	1.31e+01	4.38e+01	2.14e+03
	StErr	2.94e-01	1.64e-01	4.57e-01	2.26e+01
f_{15}	Median	3.61e+01	4.43e+01	3.47e+01	4.80e+01
	Mean	3.99e+01	4.38e+01	4.19e+01	4.81e+01
	StErr	2.76e+00	3.43e+00	4.99e+00	3.24e+00
f_{16}	Median	3.63e+01	5.20e+01	3.39e+02	9.78e+02
	Mean	3.64e+01	5.12e+01	3.38e+02	9.84e+02
	StErr	1.22e+00	1.44e+00	9.44e+00	1.98e+01
f_{17}	Median	6.05e+01	5.60e+01	2.92e+02	5.79e+02
	Mean	3.64e+01	8.88e+01	3.49e+02	5.67e+02
	StErr	1.55e+00	2.25e+01	3.53e+01	1.09e+01
f_{18}	Median	2.79e+01	3.77e+01	2.27e+02	1.17e+03
	Mean	3.13e+01	4.05e+01	2.25e+02	1.16e+03
	StErr	1.63e+00	1.79e+00	1.65e+01	3.54e+01
f_{19}	Median	2.38e+01	2.29e+01	8.14e+01	5.01e+03
	Mean	2.62e+01	2.62e+01	8.96e+01	4.98e+03
	StErr	2.51e+00	3.11e+00	7.48e+00	2.24e+01
f_{20}	Median	1.49e+02	1.97e+02	8.92e+01	1.75e+02
	Mean	1.70e+02	1.88e+02	1.15e+02	1.71e+02
	StErr	5.00e+01	1.27e+01	1.54e+00	1.48e+01

with maximum impact on the overall solution quality. On the fully nonseparable functions however (f_5, f_{10}, f_{15}, and f_{20}), the only active resources allocation mechanism is the swarm level. The relative high dimensionality of the only available component causes the swarm-level mechanism to lose its efficiency because of slow convergence and existence of many active swarms.

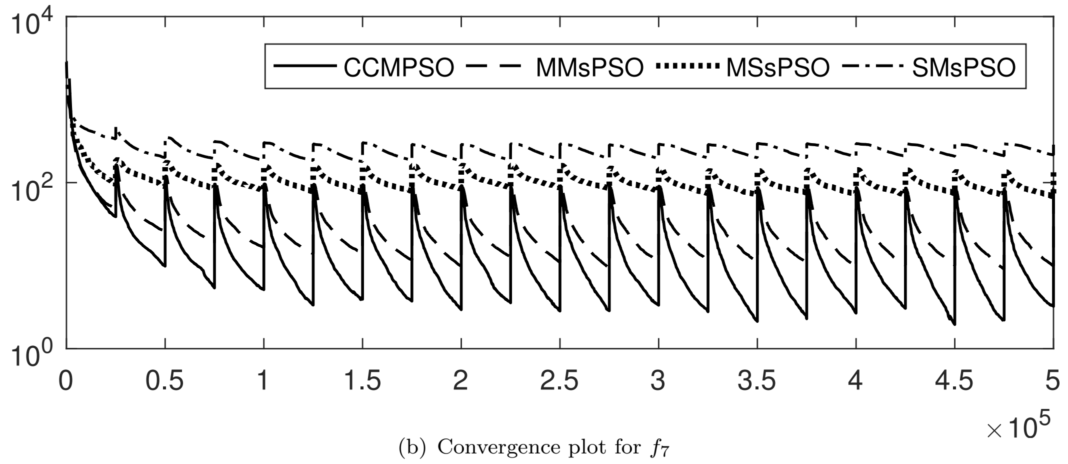
Another interesting observation is a sharp contrast between the performance of multi-swarm methods (CCMPSO and MMsPSO) and the only single-swarm method (MSsPSO).

TABLE 6.9: Obtained results by algorithms on f_6 to f_{10} with different number of peaks m for each component randomized in the following ranges $\{1,\ldots,5\}$, $\{1,\ldots,10\}$, and $\{1,\ldots,20\}$. Other parameters of CMPB are set as shown in Table 6.1.

		$m \in \{1,\ldots,5\}$				$m \in \{1,\ldots,10\}$				$m \in \{1,\ldots,20\}$			
$F(\mathbf{x})$	Stats.	CCMPSO	MMsPSO	MSsPSO	SMsPSO	CCMPSO	MMsPSO	MSsPSO	SMsPSO	CCMPSO	MMsPSO	MSsPSO	SMsPSO
f_6	Median	2.54e+00	5.57e+00	1.03e+02	1.90e+02	4.56e+00	8.77e+00	1.14e+02	1.77e+02	9.52e+00	1.33e+01	1.12e+02	2.02e+02
	Mean	2.68e+00	6.21e+00	1.02e+02	1.93e+02	5.19e+00	9.09e+00	1.18e+02	1.79e+02	1.01e+01	1.33e+01	1.16e+02	2.08e+02
	StErr	2.06e-01	4.94e-01	3.33e+00	4.59e+00	4.33e-01	4.69e-01	3.96e+00	4.84e+00	6.36e-01	6.97e-01	3.97e+00	5.80e+00
f_7	Median	2.36e+00	4.61e+00	6.01e+01	2.63e+02	4.42e+00	6.53e+00	6.68e+01	2.47e+02	7.04e+00	9.32e+00	6.73e+01	2.62e+02
	Mean	2.75e+00	5.22e+00	6.13e+01	2.62e+02	4.44e+00	6.58e+00	6.62e+01	2.46e+02	7.60e+00	9.10e+00	6.56e+01	2.56e+02
	StErr	1.95e-01	5.27e-01	2.71e+00	3.59e+00	2.51e-01	3.43e-01	2.66e+00	3.63e+00	3.82e-01	3.29e-01	2.31e+00	4.61e+00
f_8	Median	2.14e+00	3.01e+00	1.04e+02	2.08e+02	3.64e+00	5.95e+00	1.14e+02	2.07e+02	8.07e+00	1.21e+01	1.29e+02	2.55e+02
	Mean	2.15e+00	3.16e+00	1.01e+02	2.11e+02	4.20e+00	6.22e+00	1.17e+02	1.96e+02	8.53e+00	1.28e+01	1.26e+02	2.57e+02
	StErr	1.77e-01	2.08e-01	2.63e+00	4.33e+00	3.47e-01	4.86e-01	4.23e+00	6.15e+00	6.03e-01	6.71e-01	3.80e+00	6.53e+00
f_9	Median	3.41e+00	3.64e+00	1.92e+01	8.61e+02	6.08e+00	6.73e+00	2.17e+01	8.16e+02	9.55e+00	1.11e+01	2.47e+01	8.38e+02
	Mean	3.39e+00	3.67e+00	1.91e+01	8.69e+02	6.03e+00	6.48e+00	2.18e+01	8.19e+02	1.19e+01	1.15e+01	2.51e+01	8.29e+02
	StErr	1.39e-01	7.36e-02	3.22e-01	1.06e+01	1.61e-01	1.26e-01	2.73e-01	1.11e+01	2.17e+00	4.01e-01	8.22e-01	4.70e+00
f_{10}	Median	6.17e+00	8.08e+00	1.45e+01	7.75e+00	7.76e+00	8.38e+00	1.66e+01	8.05e+00	6.99e+00	8.42e+00	1.89e+01	8.36e+00
	Mean	7.30e+00	8.63e+00	1.61e+01	8.51e+00	7.77e+00	9.67e+00	2.01e+01	8.22e+00	7.76e+00	9.19e+00	2.11e+01	9.19e+00
	StErr	7.52e-01	7.24e-01	1.64e+00	7.27e-01	6.11e-01	7.92e-01	2.66e+00	8.00e-01	6.24e-01	6.90e-01	1.88e+00	7.09e-01



(a) Convergence plot for f_6



(b) Convergence plot for f_7

FIGURE 6.2: Convergence plot of CCMPSO, MMsPSO, MSsPSO and SMsPSO based on the average current error of 31 runs on f_6 and f_7 for the first 20 environments.

These are all decomposition based where each component is optimized independently. CCMPSO and MMsPSO use multiple swarms for each component whereas MSsPSO uses a single swarm for each component. All these methods benefit from an ideal decomposition which eliminates the issue of an exponentially growing number of peaks. However, the comparison clearly shows that a special mechanism for tracking multiple moving optima should be in place to obtain acceptable results. In other words, simple mechanism such as re-initialization and injection of the best found solution into the swarm are not sufficient for efficient handling of environmental changes.

Figure 6.2 shows the convergence plot of the four algorithms on f_6 and f_7. The convergence plots are based on current error of the context vector after each function evaluation for the first 20 environments. The figure shows that the algorithms try to find better solutions until the end of an environment where the error jumps due to an environmental change. CCMPSO and MMsPSO which are decomposition-based and track multiple optima outperform SMsPSO and MSsPSO across all environments. CCMPSO has a clear advantage over MMsPSO due to its efficient resource allocation mechanism. As can be seen in Fig. 6.2, for the first environment, the algorithms try to find uncovered peaks to track them after environmental changes. That is why the results obtained in the first environment are worse. This circumstance is more obvious for MMsPSO and CCMPSO which suffer from uncovered peaks until the fifth environment. After this phase, the algorithms are more stable and their results are improved because most peaks are identified and the tracker swarms can converge faster to the new optimum after each environmental change.

6.3.4.2 Robustness to dynamic changes

Table 6.9 shows the results obtained by the four algorithms on f_6-f_{10} with different numbers of peaks. The results show that the performance of all algorithms deteriorates as the number of peaks increases. However, CCMPSO maintains the best performance across all three cases. For the multi-swarm algorithms (CCMPSO, MMsPSO, and SMsPSO) the increase in the number of peaks results in more tracker swarms, which increases the computational overhead of these algorithms. Among these methods, SMsPSO has the worst performance and experiences an exponential growth in the number of peaks due to its lack of decomposition (see Fig. 6.1). CCMPSO performs better than MMsPSO thanks to its resource allocation mechanism, which makes it less susceptible to an increase in the number of peaks (hence more tracker swarms). MSsPSO, which maintains a single population, also suffers from an increase in the number of peaks. The reason is that the increased number of peaks adds to the complexity of the landscape and increases the likelihood of a premature convergence.

Table 6.10 shows the results obtained by the four algorithms on f_6-f_{10} with different shift severities. It is clear that stronger shift severities, i.e. large displacement in the location of a peak, makes tracking more difficult and time consuming. Table 6.10 shows that CCMPSO has the best overall performance across all three severity levels. The results clearly show that CCMPSO has a better competitive advantage on problems with stronger shift severities. The resource allocation mechanism of CCMPSO allows it to prioritize its limited computational resources for tracking of important peaks. On simpler problems with a smaller shift magnitude, other algorithms with no resource allocation mechanism such MMsPSO can also track the peaks with a relatively good efficiency and accuracy. This is because the number of available function evaluations between successive environmental changes is large enough to track all the peaks accurately.

Table 6.11 shows the results obtained by the four algorithms on f_6-f_{10} with different change frequencies^0. The results show that the performance of all methods declines

TABLE 6.10: Results obtained by algorithms on f_6 to f_{10} with different shift severity values for each peak in each component. The values are randomized in the following ranges $[0.5, 1]$, $[0.5, 3]$, and $[0.5, 5]$. Other parameters of CMPB are set as shown in Table 6.1.

		$S \in [0.5, 1]$				$S \in [0.5, 3]$				$S \in [0.5, 5]$			
$F(\mathbf{x})$	Stats.	CCMPSO	MMsPSO	MSsPSO	SMsPSO	CCMPSO	MMsPSO	MSsPSO	SMsPSO	CCMPSO	MMsPSO	MSsPSO	SMsPSO
f_6	Median	3.67e+00	6.74e+00	1.10e+02	1.50e+02	4.56e+00	8.77e+00	1.14e+02	1.77e+02	6.20e+00	1.14e+01	1.13e+02	2.49e+02
	Mean	3.82e+00	7.27e+00	1.15e+02	1.53e+02	5.19e+00	9.09e+00	1.18e+02	1.79e+02	7.15e+00	1.18e+01	1.19e+02	2.47e+02
	StErr	2.85e-01	5.27e-01	4.30e+00	3.12e+00	4.33e-01	4.69e-01	3.96e+00	4.84e+00	5.51e-01	5.72e-01	3.94e+00	5.66e+00
f_7	Median	3.79e+00	4.26e+00	6.26e+01	2.12e+02	4.42e+00	6.53e+00	6.68e+01	2.47e+02	5.84e+00	7.75e+00	7.01e+01	3.12e+02
	Mean	4.39e+00	4.35e+00	6.24e+01	2.13e+02	4.44e+00	6.58e+00	6.62e+01	2.46e+02	6.01e+00	7.98e+00	6.80e+01	3.13e+02
	StErr	3.02e-01	2.21e-01	2.64e+00	2.39e+00	2.51e-01	3.43e-01	2.66e+00	3.63e+00	4.12e-01	4.05e-01	2.58e+00	3.96e+00
f_8	Median	3.53e+00	4.02e+00	1.20e+02	1.73e+02	3.64e+00	5.95e+00	1.14e+02	2.07e+02	5.05e+00	7.60e+00	1.24e+02	2.89e+02
	Mean	3.43e+00	4.68e+00	1.19e+02	1.74e+02	4.20e+00	6.22e+00	1.17e+02	1.96e+02	5.30e+00	7.80e+00	1.18e+02	2.86e+02
	StErr	3.33e-01	4.49e-01	4.08e+00	4.35e+00	3.47e-01	4.86e-01	4.23e+00	6.15e+00	3.78e-01	4.89e-01	4.14e+00	7.14e+00
f_9	Median	7.63e+00	5.04e+00	1.87e+01	7.02e+02	6.08e+00	6.73e+00	2.17e+01	8.16e+02	5.93e+00	7.97e+00	2.25e+01	9.53e+02
	Mean	7.95e+00	5.04e+00	1.87e+01	7.00e+02	6.03e+00	6.48e+00	2.18e+01	8.19e+02	6.07e+00	7.76e+00	2.24e+01	9.61e+02
	StErr	2.28e-01	1.27e-01	2.03e-01	5.61e+00	1.61e-01	1.26e-01	2.73e-01	1.11e+01	1.29e-01	1.76e-01	1.92e-01	8.56e+00
f_{10}	Median	6.05e+00	6.43e+00	1.65e+01	6.53e+00	7.76e+00	8.38e+00	1.66e+01	8.05e+00	9.48e+00	1.03e+01	1.66e+01	9.48e+00
	Mean	6.72e+00	7.36e+00	2.05e+01	7.59e+00	7.77e+00	9.67e+00	2.01e+01	8.22e+00	9.55e+00	1.15e+01	1.99e+01	1.03e+01
	StErr	4.82e-01	5.44e-01	2.66e+00	7.53e-01	6.11e-01	7.92e-01	2.66e+00	8.00e-01	7.85e-01	9.29e-01	2.10e+00	8.11e-01

TABLE 6.11: Results obtained by algorithms on f_6 to f_{10} with different change frequencies: $200D$, $500D$, and $1000D$. Other parameters of CMPB are set as shown in Table 6.1.

		$f = 200D$				$f = 500D$				$f = 1000D$			
$F(\mathbf{x})$	Stats.	CCMPSO	MMsPSO	MSsPSO	SMsPSO	CCMPSO	MMsPSO	MSsPSO	SMsPSO	CCMPSO	MMsPSO	MSsPSO	SMsPSO
f_6	Median	1.83e+01	2.90e+01	1.93e+02	2.76e+02	4.56e+00	8.77e+00	1.14e+02	1.77e+02	1.90e+00	2.84e+00	9.08e+01	1.62e+02
	Mean	1.98e+01	2.91e+01	2.03e+02	2.76e+02	5.19e+00	9.09e+00	1.18e+02	1.79e+02	2.25e+00	3.00e+00	8.84e+01	1.62e+02
	StErr	8.62e-01	1.17e+00	6.59e+00	5.65e+00	4.33e-01	4.69e-01	3.96e+00	4.84e+00	2.59e-01	2.44e-01	2.90e+00	3.80e+00
f_7	Median	1.61e+01	2.00e+01	1.05e+02	3.32e+02	4.42e+00	6.53e+00	6.68e+01	2.47e+02	2.16e+00	1.91e+00	4.55e+01	2.30e+02
	Mean	1.60e+01	1.96e+01	1.03e+02	3.34e+02	4.44e+00	6.58e+00	6.62e+01	2.46e+02	2.39e+00	2.19e+00	4.70e+01	2.25e+02
	StErr	6.01e-01	7.79e-01	3.47e+00	3.88e+00	2.51e-01	3.43e-01	2.66e+00	3.63e+00	2.03e-01	1.79e-01	2.15e+00	2.86e+00
f_8	Median	1.33e+01	2.34e+01	1.73e+02	3.18e+02	3.64e+00	5.95e+00	1.14e+02	2.07e+02	2.24e+00	2.13e+00	9.89e+01	1.84e+02
	Mean	1.44e+01	2.39e+01	1.76e+02	3.20e+02	4.20e+00	6.22e+00	1.17e+02	1.96e+02	2.47e+00	2.52e+00	1.03e+02	1.89e+02
	StErr	6.50e-01	1.18e+00	5.29e+00	8.26e+00	3.47e-01	4.86e-01	4.23e+00	6.15e+00	2.78e-01	3.00e-01	3.88e+00	4.73e+00
f_9	Median	1.74e+01	2.79e+01	5.82e+01	1.05e+03	6.08e+00	6.73e+00	2.17e+01	8.16e+02	3.54e+00	1.33e+00	1.16e+01	7.30e+02
	Mean	1.75e+01	2.79e+01	5.83e+01	1.04e+03	6.03e+00	6.48e+00	2.18e+01	8.19e+02	3.65e+00	1.38e+00	1.14e+01	7.18e+02
	StErr	2.53e-01	4.85e-01	5.54e-01	9.73e+00	1.61e-01	1.26e-01	2.73e-01	1.11e+01	1.32e-01	6.45e-02	1.98e-01	4.42e+00
f_{10}	Median	1.20e+01	1.51e+01	1.97e+01	1.36e+01	7.76e+00	8.38e+00	1.66e+01	8.05e+00	6.03e+00	6.12e+00	1.77e+01	6.75e+00
	Mean	1.21e+01	1.56e+01	2.21e+01	1.44e+01	7.77e+00	9.67e+00	2.01e+01	8.22e+00	5.95e+00	6.25e+00	1.85e+01	7.58e+00
	StErr	8.67e-01	1.15e+00	2.14e+00	1.12e+00	6.11e-01	7.92e-01	2.66e+00	8.00e-01	4.14e-01	4.23e-01	2.31e+00	6.88e-01

when the change frequency is high (i.e., when the number of fitness evaluations between successive environmental changes is lower). A high change frequency means that the algorithm has limited time to do an accurate global and local search, which leads to degraded performance. Despite this, a desired property of CCMPSO is its good performance on problems with a high change frequency. The results clearly show that the CCMPSO gains a significant competitive edge over other algorithms on such problems. This can be attributed to its resource allocation mechanism which allows it to benefit from the saved resources to respond to rapid environmental changes more efficiently. This property is less crucial for problems with low change frequencies, due to the availability of sufficient time between environmental changes for accurate tracking of all the peaks.

6.4 Summary

In this chapter, a thorough investigation of large-scale dynamic optimization problems (DOPs) was presented. A formal analysis of the moving peaks benchmark showed that its lack of modularity limits its applicability for the study of large-scale DOPs. A new benchmark generator based on the moving peaks benchmark was proposed for large-scale DOPs. The proposed benchmark was made by composing several weighted MPBs in which an automated weight regulates the equilibrium between fully separable components and the nonseparable ones. Additionally, a manual weight is used to artificially create imbalance between contribution of different components. Moreover, a cooperative coevolutionary multi-swarm PSO (CCMPSO) was proposed which benefits from a computational resource allocation mechanism capable of saving resources on both component level and swarm level. A wide range of problem settings were used to investigate the performance of the proposed algorithm against two other divide-and-conquer methods. The algorithms were tested on problem instances with different dimensions, variable interaction structures, shift severities, number of peaks, and change frequencies. The results showed that the problem becomes more challenging when shift severities of peaks, dimension of problem space, number of peaks, and change frequency are higher. However, the reported results showed that the proposed CCMPSO algorithm outperforms other algorithms on more challenging problems with a wider margin.

Despite improving the baseline with up to two orders of magnitude, the proposed framework has several shortcomings that can limit its applicability in certain situations. The decomposition algorithm used in this chapter cannot exploit the structure of problems with overlapping components. Many overlapping problems have sparse interaction matrices (Omidvar et al., 2015); however, the proposed framework does not have the necessary mechanisms to exploit this sparsity; therefore, treating them as fully nonseparable. Optimal decomposition of overlapping functions is an open question even in static optimization, which becomes a greater challenge in dynamic environments. Another decomposition related issue is optimal grouping of separable variables. Although one may consider a full decomposition of separable variables into a series of 1-dimensional subproblems an obvious choice, empirical evidence suggests that such decomposition is suboptimal and increases the computational overhead of cooperative coevolution (Omidvar et al., 2014b). Grouping of separable variables may decrease this computational overhead; however, imbalance considerations and the phenomenon of exponentially growing "pseudo" peaks discussed in Section 6.1 makes finding an effective decomposition a nontrivial task. Finally, dealing with fully nonseparable problems with no apparent exploitable modularity is another important issue missing form the current study.

Chapter 7

Conclusion

The thesis has tried to investigate two missing links between academic DOPs research and real-world problems i.e. DOPs with switching cost and large-scale DOPs. These two classes of DOPs are commonly found in real-world scenarios but have rarely been studied in the literature. The results of this thesis in continuous large-scale DOPs and DOPs with switching cost, which are among the first in these areas, provide a deeper understanding of the unknown characteristics and the solvability of these problems, and suggest some promising ways to solve these challenging problems using multi-swarm PSO methods.

7.1 Summary of major contributions

Details of the contributions in this thesis have been described at the end of each chapter. Here the most significant contributions are summarized as follows:

1. DOPs with PSDRs were investigated and their characteristics were identified. These problems have two objectives i.e. the optimality objective and the displacement between the consecutive solutions. Therefore, these problems are categorized as multi-objective problems with dynamic search space. On the other hand, the displacement objective has time-linkage feature because choosing a solution for each environment affects the future problem space. After identifying the characteristics of these problems, a novel multi-objective and time-linkage based method was developed for solving them. The algorithm used a simple linear weighted sum of objectives for handling multi-objective characteristic. Moreover, it used the information gathered by sub-swarms for anticipating the future situations of the chosen solutions. The performance of the proposed method were compare with

of the current methods. Experimental results indicated the poor effectiveness of the current methods on these problems and superiority of the proposed algorithm. However, a weakness of the proposed algorithm is its multi-objective handling mechanism which is not efficient in many problems.

2. The shortcomings of the current ROOT methods were investigated. It was shown that using predictors and approximators is not efficient for large problems i.e. problems with larger number of dimension and the ones with wider boundaries. Additionally, the drawbacks of the state-of-the-art survival time metric were indicated for ROOT. In fact the produced search space by this metric is very hard for optimization methods, especially in higher dimensions. A new framework was proposed in which the robust solutions were chosen by a decision making strategy. This strategy used the gathered information by sub-populations for choosing the next solution. Four different strategies were proposed which used different types of information. The experimental results showed that choosing robust solutions by learning the environmental behavior is significantly more efficient than the state-of-the-art methods. However, the proposed framework was a reaction based method which means that it is not capable of working in DOPs in which the changes are not detectable. Additionally, if peaks change very severely, the multi-population method loses its efficiency which deteriorates the performance of the framework.

3. An adaptive solution chooser (ASC) algorithm was proposed for DOPs with switching costs. ASC decided if a new solution is to be chosen or the previous one can be kept based on the current solution's fitness values, the fitness of other found solutions with better quality and their switching cost from the current solution. ASC is the first method that considered switching cost in the decision making process for changing or keeping solutions. In addition, a new performance indicator was proposed in which the switching cost was considered as a penalty value. Therefore, ASC monitored covered peaks and checked if there was any peak that the benefit from changing solution to it outweighed the penalty related to the switching cost. ASC used the same approach as the proposed framework in Chapter 4 i.e. finding robust solutions based on the learned behavior of peaks. The experimental results on a wide range of problem instances showed that ASC outperforms ROOT and TMO methods with the same multi-population approach.

4. large-scale DOPs were investigated in this thesis which rarely have been studied before. This thesis investigated this class of problems and identified its particular challenges. Scalability issues and exponentially growing number of peaks were the most important identified challenges. Additionally, in the experiments, the

results indicated the poor performance of the current DOPs algorithms for optimizing large-scale DOPs. Moreover, it was shown that although the previous DOP benchmarks are scalable, lack of modularity makes them unsuitable for large-scale studies. A formal analysis for the moving peaks benchmark was provided to prove its lack of modularity. Then, a new benchmark generator based on the MPB was proposed for large-scale DOPs. The proposed benchmark was capable of generating problem instances with real-world characteristics including modularity, heterogeneity, imbalance and high dimensionality. Furthermore, a cooperative co-evolutionary multi-swarm PSO (CCMPSO) was proposed which benefited from a bi-level computational resource allocation mechanism capable of saving resources on both component level and swarm level. The experimental results showed the effectiveness of the proposed algorithm.

7.2 Future work

There are many related research topics in DOP domain that can be pursued in the future. Among these topics, some possible interesting future research directions are:

- Focusing on dynamic constrained optimization problems (DCOPs). Many real-world DOPs are constrained. In the DCOPs, there are still significant gaps between academic research and real-world DOPs which need to be addressed.
- Focusing on dynamic multi-objective optimization problems (DMOOPs). Although this class of DOPs has been studied considerably, from the dynamic aspect of problems, there are still noticeable gaps between academic research and real-world DOPs which needs to be considered in the future works.
- Focusing on incremental optimization problems (IOPs). IOPs are a class of DOPs in which decision variables will be added to the problem over time. Therefore, the number of dimensions and variable interactions change over time. Additionally, the shape of problem landscape for previous dimensions may be changed after adding new variables to the problem.

As to the particular research topics that have been studied in this thesis, because the works that have been done in the thesis are only among the first steps in these topics, there are a lot of future works to be addressed. Some possible directions for future extensions are discussed below:

- In the ROOT area, future work will include a study of other peak's behavioral information and design of new strategies for choosing robust solutions.
- Working on ROOT in dynamic constrained optimization problems will be an important topic since many real-world problems are constrained. Therefore, the algorithm needs to consider robustness based on feasibility of solutions along with their fitness.
- In the large-scale domain, future work will include a study of large-scale dynamic optimization problems whose number of components, dimensions, and strength of interactions change over time. Additionally, working on the grouping fully separable dimensions and decomposing fully nonseparable subfunctions are other important research topics.

Appendices

Appendix A

Publications resulting from this thesis

A.1 Refereed or submitted journal papers

1. D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," IEEE Transactions on Evolutionary Computation, (Accepted), 2018
2. D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," IEEE Transactions on Evolutionary Computation, Under review (Past the first round), 2018

A.2 Refereed conference papers

3. D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, and X. Yao, "Changing or keeping solutions in dynamic optimization problems with switching costs," in The Genetic and Evolutionary Computation Conference (GECCO). ACM, , pp. 1095-1102, 2018
4. D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A multi-objective time-linkage approach for dynamic optimization problems with previous-solution displacement restriction," in Applications of Evolutionary Computation, K. Sim and P. Kaufmann, Eds. Lecture Notes in Computer Science, 2018, vol. 10784, pp. 864-878 (Nominated for the best paper award)

5. D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, “A new multi-swarm particle swarm optimization for robust optimization over time,” in Applications of Evolutionary Computation, G. Squillero and K. Sim, Eds. Springer Lecture Notes in Computer Science, 2017, vol. 10200, pp. 99-109

The following lists materials (or part) of the publications presented in the thesis:

- Chapter 2 : publications [1,2,3,4,5]
- Chapter 3 : publication [4]
- Chapter 4 : publications [1,5]
- Chapter 5 : publication [3]
- Chapter 6 : publication [2]

Bibliography

J. A. D. Atkin, E. K. Burke, J. S. Greenwood, and D. Reeson. On-line decision support for take-off runway scheduling with uncertain taxi times at london heathrow airport. *Journal of Scheduling*, 11(5):323–346, 2008. doi: 10.1007/s10951-008-0065-9.

G. Avigad, E. Eisenstadt, and O. Schuetze. Handling changes of performance requirements in multi-objective problems. *Journal of Engineering Design*, 23(8):597–617, 2010. doi: 10.1080/09544828.2011.630656.

R. Azzouz, S. Bechikh, and L. B. Said. Dynamic multi-objective optimization using evolutionary algorithms: A survey. *Recent Advances in Evolutionary Multi-objective Optimization*, 20:31–70, 2017.

H. G. Beyer and B. Sendhoff. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, 2007.

S. Bird and X. Li. Using regression to improve local convergence. In *IEEE Congress on Evolutionary Computation*, pages 592–599. IEEE, 2007. doi: 10.1109/CEC.2007.4424524.

T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006. doi: 10.1109/TEVC.2005.857074.

T. Blackwell, J. Branke, and X. Li. Particle swarms for dynamic optimization problems. In C. Blum and D. Merkle, editors, *Swarm Intelligence: Introduction and Applications*, pages 193–217. Springer, 2008.

P. A. N. Bosman. Learning, anticipation and time-deception in evolutionary online dynamic optimization. In *Annual Workshop on Genetic and Evolutionary Computation*, GECCO '05, pages 39–47. ACM, 2005. doi: 10.1145/1102256.1102264.

P. A. N. Bosman and H. L. Poutré. Learning and anticipation in online dynamic optimization with evolutionary algorithms: The stochastic case. In *Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 1165–1172. ACM, 2007. doi: 10.1145/1276958.1277187.

G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control.* Wiley, 2015.

J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 1875–1882. IEEE, 1999. doi: 10.1109/CEC.1999.785502.

J. Branke. *Evolutionary Optimization in Dynamic Environments.* Springer US, 2002.

J. Branke and D. C. Mattfeld. Anticipation in dynamic optimization: The scheduling case. In M. S. et al, editor, *Parallel Problem Solving from Nature*, volume 1917, pages 253–262. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2000.

J. Branke and D. C. Mattfeld. Anticipation and flexibility in dynamic scheduling. *International Journal of Production Research*, 43(15):3103–3129, 2005.

J. Branke, T. Kaussler, C. Smidt, and H. Schmeck. A multipopulation approach to dynamic optimization problems. In *Evolutionary Design and Manufacture*, pages 299–307, 2000. doi: 10.1007/978-1-4471-0519-0_24.

C. Bu, W. Luo, T. Zhu, and L. Yue. Solving online dynamic time-linkage problems under unreliable prediction. *Applied Soft Computing*, 56:702–716, 2017. doi: 10.1016/j.asoc.2016.11.005.

L. Bui, H. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2349–2356. IEEE, 2005. doi: 10.1109/CEC.2005.1554987.

L. T. Bui, Z. Michalewicz, E. Parkinson, and M. B. Abello. Adaptation in dynamic environments: A case study in mission planning. *IEEE Transactions on Evolutionary Computation*, 16(2):190–209, 2012.

M. Camara, J. Ortega, and F. J. Toro. Parallel processing for multi-objective optimization in dynamic environments. In *IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2007.

C. Caramanis, S. Mannor, and H. Xu. Robust optimization in machine learning. In S. S. et al., editor, *Optimization in machine learning*, page 369402. MIT Press, Cambridge, MA, USA, 2011.

V. Chankong and Y. Y. Haimes. *Multiobjective decision making theory and methodology.* New York: North-Holland, 1983.

H. Chen, M. Li, and X. Chen. Using diversity as an additional-objective in dynamic multi-objective optimization algorithms. In *International Symposium on Electronic Commerce and Security*, volume 1, pages 484–487, 2009.

W. Chen, T. Weise, Z. Yang, and K. Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *International Conference on Parallel Problem Solving from Nature*, pages 300–309. Springer, 2010.

H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. In *International Conference on Genetic Algorithms*, pages 523–530. Morgan Kaufmann Publishers Inc., 1993.

C. Cruz, J. R. Gonzlez, and D. A. Pelta. Optimization in dynamic environments– a survey on problems, methods and measures. *Soft Computing*, 15(7):14271448, 2011.

K. Deb, U. B. R. N., and S. Karthik. Dynamic multi-objective optimization and decision-making using modified nsga-ii: A case study on hydro-thermal power scheduling. In S. O. et al., editor, *Evolutionary Multi-Criterion Optimization*, volume 4403, pages 803–817. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007.

W. Du and B. Li. Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information Sciences*, 178(15):3096–3109, 2008. doi: 10.1016/j.ins.2008.01.020.

W. Du, Y. Tang, S. Y. S. Leung, L. Tong, A. V. Vasilakos, and F. Qian. Robust order scheduling in the discrete manufacturing industry: A multiobjective optimization approach. *IEEE Transactions on Industrial Informatics*, 14(1):253–264, 2018.

M. C. du Plessis and A. P. Engelbrecht. Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, 218(1):7–20, 2012. doi: 10.1016/j.ejor.2011.08.031.

R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 84–88. IEEE, 2001. doi: 10.1109/CEC.2000.870279.

D. E.Wilkins, S. F.Smith, L. A.Kramer, T. J.Lee, and T. W.Rauenbusch. Airlift mission monitoring and dynamic rescheduling. *Engineering Applications of Artificial Intelligence*, 21(2):141–155, 2008. doi: 10.1016/j.engappai.2007.04.001.

M. Farina, K. Deb, and P. Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transaction on Evolutionary Computation*, 8(5):425–442, 2004. doi: 10.1109/TEVC.2004.831456.

A. Fertis. *A robust optimization approach to statistical estimation problems.* PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., 2009.

H. Fu, B. Sendhoff, K. Tang, and X. Yao. Finding robust solutions to dynamic optimization problems. In *Applications of Evolutionary Computation*, volume 7835, pages 616–625. Lecture Notes in Computer Science, 2013. doi: 10.1007/978-3-642-37192-9_62.

H. Fu, B. Sendhoff, K. Tang, and X. Yao. Robust optimization over time: problem difficulties and benchmark problems. *IEEE Transactions on Evolutionary Computation*, 19(5):731–745, 2015. doi: 10.1109/TEVC.2014.2377125.

V. Gabrel, C. Murat, and A. Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483, 2014.

C.-K. Goh and K. C. Tan. *Evolutionary Multi-objective Optimization in Uncertain Environments*. Springer: Studies in Computational Intelligence, 2004.

J. J. Grefenstette. Evolvability in dynamic fitness landscapes: a genetic algorithm approach. In *Congress on Evolutionary Computation-CEC99*, volume 3, pages 2031–2038, 1999.

H. Greiner. Robust filter design by stochastic optimization. In *SPIE 2253, Optical Interference Coatings*, volume 2253, 1994.

Y. Guo, M. Chen, H. Fu, and Y. Liu. Find robust solutions over time by two-layer multi-objective optimization method. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1528–1535. IEEE, 2014. doi: 10.1109/CEC.2014.6900241.

I. Hatzakis and D. Wallace. Dynamic multi-objective optimization with evolutionary algorithms: A forward-looking approach. In *Annual Conference on Genetic and Evolutionary Computation*, pages 1201–1208, 2006.

J. Helton. Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty. *Journal of Statistical Computation and Simulation*, 57(1-4):3–76, 1997.

Y. Huang, Y. Ding, K. Hao, and Y. Jin. A multi-objective approach to robust optimization over time considering switching cost. *Information Sciences*, 394-395:183–197, 2017. doi: 10.1016/j.ins.2017.02.029.

Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

Y. Jin, K. Tang, X. Yu, B. Sendhoff, and X. Yao. A framework for finding robust optimal solutions over time. *Memetic Computing*, 5(01):3–18, 2013. doi: 10.1007/s12293-012-0090-2.

Y. jun Shi, H. fei Teng, and Z. qiang Li. Cooperative co-evolutionary differential evolution for function optimization. In L. W. et al., editor, *Advances in Natural Computation*, volume 3611, pages 1080–1088. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks*, volume 04, pages 1942–1948. IEEE, 1995. doi: 10.1109/ICNN.1995.488968.

W. T. Koo, C. K. Goh, and K. C. Tan. A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. *Memetic Computing*, 2(2): 87110, 2010.

C. Li and S. Yang. Optimization in dynamic environments utilizing a novel method based on particle swarm optimization. In *4th International Conference on Natural Computation*, pages 624–628. IEEE, 2008. doi: 10.1109/ICNC.2008.313.

C. Li and S. Yang. A clustering particle swarm optimizer for dynamic optimization. In *IEEE Congress on Evolutionary Computation*, pages 439–446. IEEE, 2009. doi: 10.1109/CEC.2009.4982979.

C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan. Benchmark generator for cec'2009 competition on dynamic optimization. Technical report, Center for Computational Intelligence, 2008.

C. Li, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, and H. G. Beyer. Benchmark generator for cec 2009 competition on dynamic optimization. Technical report, Department of Computer Science, University of Leicester, UK, 2009.

C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang. An adaptive multi-population framework for locating and tracking multiple optima. *IEEE Transactions on Evolutionary Computation*, 20(05):590–605, 2016. doi: 10.1109/TEVC.2015.2504383.

X. Li and X. Yao. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 16(2):210–224, 2012.

X. Li, K. Tang, M. N. Omidvar, Z. Yang, , and K. Qin. Benchmark functions for the CEC2013 special session and competition on large-scale global optimization. Technical report, RMIT University, 2013.

D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, and B. S. Lee. Inverse multi-objective robust evolutionary design. *Genetic Programming and Evolvable Machines*, 7(4):383–404, 2006.

C. Liu. New dynamic multiobjective evolutionary algorithm with core estimation of distribution. In *2010 International Conference on Electrical and Control Engineering*, pages 1345–1348, 2010.

J. Liu and K. Tang. Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In H. Y. et al., editor, *Intelligent Data Engineering and Automated Learning*, volume 8206, pages 350–357. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2013.

W. Luo, B. Yang, C. Bu, and X. Lin. A hybrid particle swarm optimization for high-dimensional dynamic optimization. In Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, editors, *Simulated Evolution and Learning*, volume 10593, pages 981–993. Springer Lecture Notes in Computer Science, 2017. doi: 10.1007/978-3-319-68759-9_81.

S. Mahadevan and R. Rebba. Inclusion of model errors in reliability-based optimization. *Journal of Mechanical Design*, 128(4):936–944, 2006.

S. Mahdavi, S. Rahnamayan, and M. E. Shiri. Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization. *Applied Intelligence*, 47(3):888–913, 2017a.

S. Mahdavi, S. Rahnamayan, and M. E. Shiri. Multilevel framework for large-scale global optimization. *Soft Computing*, 21(14):4111–4140, 2017b.

M. Mavrovouniotis, C. Li, and S. Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17, 2017. doi: 10.1016/j.swevo.2016.12.005.

Y. Mei, M. N. Omidvar, X. Li, and X. Yao. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software (TOMS)*, 42(2):13, 2016.

R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments.* Springer: Natural Computing Series, 2004.

R. W. Morrison and K. A. D. Jong. A test problem generator for non-stationary environments. In *Congress on Evolutionary Computation-CEC99*, volume 3, pages 2047–2053, 1999.

A. Muruganantham, K. C. Tan, and P. Vadakkepat. Solving the ieee cec 2015 dynamic benchmark problems using kalman filter based dynamic multiobjective evolutionary algorithm. In K. L. et al., editor, *Adaptation, Learning and Optimization*, volume 5, pages 239–252. Intelligent and Evolutionary Systems, Springer, 2016.

T.-D. Nguyen and A. W. Lo. Robust ranking and portfolio optimization. *European Journal of Operational Research*, 221(2):407–416, 2012.

T. T. Nguyen. *Continuous dynamic optimisation using evolutionary algorithms.* PhD thesis, University of Birmingham, Birmingham, UK, 2011.

T. T. Nguyen and X. Yao. Dynamic time-linkage problems revisited. In M. G. et al., editor, *European Conference on the Applications of Evolutionary Computation*, volume 5484, pages 735–744. Lecture Notes in Computer Science, 2009. doi: 10.1007/978-3-642-01129-0_83.

T. T. Nguyen and X. Yao. Dynamic time-linkage evolutionary optimization: Definitions and potential solutions. In E. Alba, A. Nakib, and P. Siarry, editors, *Metaheuristics for Dynamic Optimization*, volume 433, pages 371–395. Studies in Computational Intelligence, 2013.

T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012a. doi: 10.1016/j.swevo.2012.05.001.

T. T. Nguyen, Z. Yang, and S. Bonsall. Dynamic time-linkage problems - the challenges. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–6. IEEE, 2012b. doi: 10.1109/rivf.2012.6169823.

M. N. Omidvar, X. Li, Z. Yang, and X. Yao. Cooperative co-evolution for large scale optimization through more frequent random grouping. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

M. N. Omidvar, X. Li, and X. Yao. Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In *Annual conference on Genetic and evolutionary computation*, pages 1115–1122. ACM, 2011.

M. N. Omidvar, X. Li, Y. Mei, and X. Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):378–393, 2014a.

M. N. Omidvar, Y. Mei, and X. Li. Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1305–1312. IEEE, 2014b.

M. N. Omidvar, X. Li, and K. Tang. Designing benchmark problems for large-scale continuous optimization. *Information Sciences*, 316:419–436, 2015. doi: 10.1016/j.ins.2014.12.062.

M. N. Omidvar, B. Kazimipour, X. Li, and X. Yao. CBCC3 – a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 3541–3548. IEEE, 2016.

M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao. DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 21(6):929–942, 2017.

D. Parrott and X. Li. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, 10 (04):440–458, 2006. doi: 10.1109/TEVC.2005.859468.

C. Peng, P. Xie, L. Pan, and R. Yu. Flexible robust optimization dispatch for hybrid wind/photovoltaic/hydro/thermal power system. *IEEE Transactions on Smart Grid*, 7(2):751–762, 2016.

O. V. Pictet, M. M. Dacorogna, B. Chopard, M. Oussaidene, R. Schirru, and M. Tomassini. Using genetic algorithms for robust optimization in financial applications. available at ssrn. Technical report, Center for Computational Intelligence, 1998.

M. A. Potter and K. A. D. Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29, 2000.

M. Preuss. Niching the CMA-ES via nearest-better clustering. In *12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1711–1718. ACM, 2010. doi: 10.1145/1830761.1830793.

T. Ray and X. Yao. A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In *IEEE Congress on Evolutionary Computation*, pages 983–989, 2009.

S. Salomon, G. Avigad, P. J. Fleming, and R. C. Purshouse. Optimization of adaptation - a multi-objective approach for optimizing changes to design parameters. In R. Purshouse, P. Fleming, C. Fonseca, and J. S. S. Greco, editors, *Evolutionary Multi-Criterion Optimization*, volume 7811, pages 21–35. Springer Lecture Notes in Computer Science, 2013.

A. Sharifi, J. K. Kordestani, M. Mahdaviani, and M. R. Meybodi. A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems. *Applied Soft Computing*, 32:432–448, 2015. doi: 10.1016/j.asoc.2015.04.001.

L. Sun, S. Yoshida, X. Cheng, and Y. Liang. A cooperative particle swarm optimizer with statistical variable interdependence learning. *Information Sciences*, 186(1):20–39, 2012.

Y. Sun, M. Kirley, and S. K. Halgamuge. Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In *Annual Conference on Genetic and Evolutionary Computation*, GECCO 2015, pages 313–320, New York, NY, USA, 2015. ACM.

Y. Sun, M. Kirley, and S. K. Halgamuge. A recursive decomposition method for large scale continuous optimization. *IEEE Transactions on Evolutionary Computation*, 2017.

K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark functions for the CEC2010 special session and competition on large-scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, 2009.

R. K. Ursem, T. Krink, M. T. Jensen, and Z. Michalewicz. Analysis and modeling of control tasks in dynamic systems. *IEEE Transactions on Evolutionary Computation*, 6(4):378–389, 2002.

F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transaction on Evolutionary Computation*, 8(3):225–239, 2004. doi: 10.1109/TEVC.2004.826069.

Y. Wang and B. Li. Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment. In *2009 IEEE Congress on Evolutionary Computation*, pages 630–637, 2009. doi: 10.1109/CEC.2009.4983004.

J. Wei and L. Jia. A novel particle swarm optimization algorithm with local search for dynamic constrained multi-objective optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 2436–2443, 2013. doi: 10.1109/CEC.2013.6557861.

J. Wei and Y. Wang. Hyper rectangle search based particle swarm algorithm for dynamic constrained multi-objective optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2012. doi: 10.1109/CEC.2012.6256137.

K. Weicker. *Evolutionary algorithms and dynamic optimization problems.* PhD thesis, University of Stuttgart, Stuttgart, Germany, 2003.

K. Weicker and N. Weicker. Dynamic rotation and partial visibility. In *Congress on Evolutionary Computation*, page 11251131, 2000.

R. P. Wiegand, W. C. Liles, and K. A. D. Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Annual Conference on Genetic and Evolutionary Computation*, pages 1235–1242. Morgan Kaufmann Publishers Inc., 2001.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83, 1945.

H. Xu. *Robust Decision Making and Its Applications in Machine Learning*. PhD thesis, Montreal, Que., Canada, Canada, 2009. AAINR61804.

H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

M. Yang, M. N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, and X. Yao. Efficient resource allocation in cooperative co-evolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation*, 21(4):493–505, 2017.

S. Yang and C. Li. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 14(06):959–974, 2010. doi: 10.1109/TEVC.2010.2046667.

S. Yang and X. Yao, editors. *Evolutionary Computation for Dynamic Optimization Problems*. Springer-Verlag Berlin Heidelberg, 2013.

Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, 2008a.

Z. Yang, K. Tang, and X. Yao. Multilevel cooperative coevolution for large scale optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1663–1670, 2008b.

D. Yazdani, B. Nasiri, R. Azizi, A. Sepas-Moghaddam, and M. R. Meybodi. Optimization in dynamic environments utilizing a novel method based on particle swarm optimization. *International Journal of Artificial Intelligence*, 11:170–192, 2013a.

D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi. A novel multiswarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing*, 13(04):2144–2158, 2013b. doi: 10.1016/j.asoc.2012.12.020.

D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. R. Meybodi, and M. Akbarzadeh-Totonchi. mNAFSA: A novel approach for optimization in dynamic environments with global changes. *Swarm and Evolutionary Computation*, 18:38–53, 2014. doi: 10.1016/j.swevo.2014.05.002.

D. Yazdani, A. Sepas-Moghaddam, A. Dehban, and N. Horta. A novel approach for optimization in dynamic environments based on modified artificial fish swarm algorithm. *International Journal of Computational Intelligence and Applications*, 15(02): 1650010–1650034, 2016. doi: 10.1142/S1469026816500103.

D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang. A new multi swarm particle swarm optimization for robust optimization over time. In G. Squillero and K. Sim, editors, *Applications of Evolutionary Computation*, volume 10200, pages 99–109. Springer Lecture Notes in Computer Science, 2017.

D. Yazdani, T. T. Nguyen, and J. Branke. Robust optimization over time by learning problem space characteristics. *IEEE Transactions on Evolutionary Computation*, 2018a.

D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang. A multi-objective time-linkage approach for dynamic optimization problems with previous-solution displacement restriction. In *European Conference on the Applications of Evolutionary Computation.* Lecture Notes in Computer Science, 2018b.

A. Younes. *Adapting Evolutionary Approaches for Optimization in Dynamic Environments*. PhD thesis, University of Waterloo, Waterloo, Canada, 2006.

X. Yu, Y. Jin, K. Tang, and X. Yao. Robust optimization over time - a new perspective on dynamic optimization problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6. IEEE, 2010. doi: 10.1109/CEC.2010.5586024.

A. E. M. Zavala. A comparison study of pso neighborhoods. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation)*, volume 175, pages 251–265. Advances in Intelligent Systems and Computing, 2013.

B. Zheng. A new dynamic multi-objective optimization evolutionary algorithm. In *International Conference on Natural Computation (ICNC 2007)*, volume 5, pages 565–570, 2007.

A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. In S. O. et al., editor, *Evolutionary Multi-Criterion Optimization*, volume 4403, pages 832–846. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007.

A. Zhou, Y. Jin, and Q. Zhang. A population prediction strategy for evolutionary dynamic multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(1):40–53, 2014.