Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften

(Dr. rer. nat.)

Evolutionary Multi-Objective Optimization for Computation Offloading in Collaborative Edge-Cloud Computing

eingereicht
am Institut für Informatik
des Fachbereichs Mathematik und Informatik
der Freien Universität Berlin
von

Guang Peng

Berlin, 2021

Gutachter:

Prof. Dr. Katinka Wolter
Department of Computer Science
Freie Universität Berlin, Germany

Associate Prof. Dr. Huaming Wu
Center for Applied Mathematics
Tianjin University, China

Disputation: 25.10.2021

Selbständigkeitserklärung

Ich versichere, dass ich die Doktorarbeit selbständig verfasst, und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit hat keiner anderen Prüfungsbehörde vorgelegen.

Guang Peng

Berlin, den 20. Dezember 2021

Abstract

Many real-world problems involve multiple and conflicting objectives to be optimized simultaneously, which are formulated as multi-objective optimization problems (MOPs). In general, problems with more than three objectives or a large number of decision variables are referred to as many-objective or large-scale optimization problems (MaOPs or LSMOPs), respectively. Recently, a variety of multi-objective evolutionary algorithms (MOEAs) inspired by nature have been developed to solve MOPs. However, the performance of MOEAs often deteriorates when faced with many-objective or large-scale optimization. In addition, these MOEAs using the classical Pareto dominance principle face no strong selection pressure towards the Pareto front with regard to many-objective optimization. Furthermore, it is difficult for a limited population size to explore the high-dimensional search space of large-scale optimization.

With the development of the Internet of things (IoT) and mobile networks, more and more computation-intensive and latency-critical applications are deployed to mobile devices. Due to some limitations inherent to mobile devices including limited computing capability, storage space, and battery lifetime, these applications cannot run efficiently on mobile devices. To address this issue, computation offloading in mobile cloud and edge computing (MCC and MEC) provides a promising paradigm to migrate computation-heavy parts of mobile applications to the cloud and edge servers. Hence, MCC and MEC computation offloading may involve different objectives such as reducing time and saving energy. Offloading decision making can be described as multi-criteria optimization and we develop efficient MOEAs to solve different computation offloading models.

This work covers evolutionary multi-objective optimization for computation offloading in collaborative MCC and MEC. Its main content includes two parts: (i) evolutionary multi-objective optimization and (ii) optimization in offloading. Specifically, the contributions of this thesis can be summarized as follows:

- Proposing a multi-objective artificial bee colony algorithm based on decomposition to improve convergence and diversity for solving normalized and scaled MOPs.

- Developing a many-objective evolutionary algorithm with adaptive weight vectors for dealing with normalized and scaled MaOPs.
- Designing a novel archive maintenance for adapting weight vectors in decomposition-based evolutionary algorithms for handling MOPs and MaOPs with irregular Pareto fronts.
- Proposing three constrained MOEAs to deal with constrained MOPs as well as offloading problems in IoT-edge-cloud computing networks.
- Exploring and comparing two evolutionary large-scale sparse multi-objective optimization algorithms for tackling collaborative edge-cloud offloading problems.
- Studying a novel multi-objective computation offloading algorithm to solve offloading problems with the consideration of compression, security and mobility.

Zusammenfassung

Bei vielen realen Problemen müssen mehrere und widersprüchliche Ziele gleichzeitig optimiert werden, die als Multi-Objective Optimierungsprobleme (MOPs) formuliert werden. Im Allgemeinen werden Probleme mit mehr als drei Zielen oder einer großen Anzahl von Entscheidungsvariablen als Optimierungsprobleme mit vielen Zielen oder großen Optimierungen (MaOPs bzw. LS-MOPs) bezeichnet. In jüngster Zeit wurde eine Vielzahl von von der Natur inspirierten Multi-Objective-Evolutionary-Algorithmen (MOEAs) entwickelt, um MOPs zu lösen. Die Leistung von MOEAs verschlechtert sich jedoch häufig, wenn sie mit einer Optimierung mit vielen Zielen oder in großem Maßstab konfrontiert werden. Darüber hinaus sind diese MOEAs, die das klassische Pareto-Dominanzprinzip verwenden, keinem starken Selektionsdruck gegenüber der Pareto-Front im Hinblick auf eine Optimierung mit vielen Zielen ausgesetzt. Darüber hinaus ist es für eine begrenzte Populationsgröße schwierig, den hochdimensionalen Suchraum einer groß angelegten Optimierung zu erkunden.

Mit der Entwicklung des Internet der Dinge (IoT) und mobiler Netzwerke werden immer mehr rechenintensive und latenzkritische Anwendungen auf mobilen Geräten bereitgestellt. Aufgrund einiger Einschränkungen, die mobilen Geräten inhärent sind, einschließlich eingeschränkter Rechenkapazität, Speicherplatz und Akkulaufzeit, können diese Anwendungen auf mobilen Geräten nicht effizient ausgeführt werden. Um dieses Problem zu beheben, bietet das Auslagern von Berechnungen in Mobile Cloud und Edge Computing (MCC und MEC) ein vielversprechendes Paradigma für die Migration rechenintensiver Teile mobiler Anwendungen in die Cloud und auf Edgeserver. Daher kann das Auslagern von MCC- und MEC-Berechnungen unterschiedliche Ziele beinhalten, wie z. B. Zeitersparnis, Energieeinsparung und Verbesserung der Sicherheit. Die Entscheidungsfindung beim offloading kann als Optimierung mehrerer Kriterien beschrieben werden. Wir entwickeln effiziente MOEAs, um verschiedene Modelle für das offloading von Berechnungen zu lösen.

Diese Arbeit befasst sich mit der evolutionären Optimierung mehrerer Ziele für das Auslagern von Berechnungen in kollaborativen MCC und MEC. Sein Hauptinhalt besteht aus zwei Teilen:

(i) evolutionäre Multiobjektive Optimierung und (ii) Optimierung beim Offloading. Insbesondere können die Beiträge dieser Arbeit wie folgt zusammengefasst werden:

- Vorschlag eines auf Zersetzung basierenden Algorithmus für künstliche Bienenkolonien mit mehreren Zielsetzungen zur Verbesserung der Konvergenz und Diversität zur Lösung von normalisierten und skalierten MOPs.
- Entwicklung eines evolutionären Algorithmus mit vielen Zielen und adaptiven Gewichtsvektoren für den Umgang mit normalisierten und skalierten MaOPs.
- Entwurf einer neuartigen Archivpiesing zur Anpassung von Gewichtsvektoren in zerlegungsbasierten evolutionären Algorithmen zur Behandlung von MOPs und MaOPs mit unregelmäßigen Pareto-Fronten.
- Vorschlag von drei eingeschränkten MOEAs zur Bewältigung eingeschränkter MOPs sowie zum Auslagern von Problemen in IoT-Edge-Cloud-Computing-Netzwerken.
- Untersuchen und Vergleichen von zwei evolutionären, spärlichen Optimierungsalgorithmen mit mehreren Zielen, um kollaborative Edge-Cloud-Offloading-Probleme anzugehen.
- Untersuchung eines neuartigen Multi-Objective-Computation-Offloading-Algorithmus zur Lösung von Offloading-Problemen unter Berücksichtigung von Komprimierung, Sicherheit und Mobilität.

Acknowledgements

My study is under the financial support by Xi'an Jiaotong University and has been carried out at Dependable Distributed Systems (DDS) group at Freie Universität Berlin, Germany. I am very thankful to them for providing me with such a valuable working opportunity. I would like to express my sincere gratitude to everyone who contributed to the completion of this thesis.

First and foremost, I would like to thank my supervisor Prof. Dr. Katinka Wolter. I really appreciate her invaluable advice and guidance throughout my research in FUB. She taught me how to write good papers and prepare presentations. She carefully revised my papers sentence by sentence. I deeply admire her work attitude and professional knowledge. She has been more than a perfect supervisor during my life in FUB. When I came to Berlin, she helped and encouraged me to adapt to a new culture and start a new life. She provided me with a lot of help and suggestion for both life and work. Without her advice and patience, this thesis would have never been possible. I really appreciate everything she has done for me.

Thanks to my colleagues Xiao Jia, Dr. Zhihao Shang and Dr. Han Wu. They created an ever nice and friendly working atmosphere in the institute. I enjoyed working with them very much. They shared their interesting ideas and discussed the problems I faced. Thanks for their company and memorable time.

Thanks to my friends Hanxing Lin, who gave me a lot of support and helpful suggestions during the research.

Last but not least, I am very grateful for the love and support of my parents, who have provided continuous motivation for my education.

Contents

Part I

Introduction

Chapter 1

Basic Problems

In this chapter the problems considered in this thesis will be explained, the contributions are illustrated and an outline is given.

1.1 Problem Statement

Multi-objective optimization problems (MOPs) [116] exist in real-world applications. A decision-maker often needs to handle different conflicting objectives. For example, a hybrid electric vehicle controller design problem proposed in [1] consists of seven optimization objectives: fuel consumption, battery stress, internal combustion engine (ICE) operation changes, ICE emissions, ICE noise, urban operation, and average battery state of charge level. These seven optimization objectives together determine the performance of the hybrid electric vehicle controller. Generally, the MOPs with more than three objectives are named many-objective optimization problems (MaOPs) [69]. In addition, the MOPs with a set of equality and/or inequality constraints are denoted constrained MOPs (CMOPs) [31]. The improvement of one objective may lead to deterioration of other objectives. Different from a single solution in single-objective optimization, a set of non-dominated solutions are used to balance different objectives in MOPs. Another challenge of multi-objective optimization is to deal with the large-scale optimization [119], which means the MOPs are associated with a large number of decision variables.

There are a number of open problems in classical optimization algorithms. The classical optimization methods such as weighted sum cannot perform well for MOPs, sometimes it is also difficult to set the weights for different objectives. On the other hand, some MOPs like multi-objective travelling salesman problem are NP-hard, the traditional deterministic methods might

need a large computational budget. Multi-objective evolutionary algorithms (MOEAs) [14] have been popularly developed to solve these problems. Evolutionary algorithms (EAs) [82] belong to the metaheuristic algorithms, which are inspired by the natural evolution from biology. Most of the EAs are population-based algorithms and have the advantages of global optimization. However, some MOEAs can achieve good results for MOPs, while they may encounter difficulties in facing MaOPs. With the increasing number of objectives, the solutions with many objectives are usually non-dominated by each other. Pareto dominance has no strong selection pressure to make the non-dominated solutions approximate the true Pareto front. Furthermore, a huge number of solutions are needed to represent the entire Pareto front and how to choose a single final solution is difficult for the decision-maker. As for the large-scale optimization, with the growing number of decision variables, it becomes much more challenging for MOEAs to search in a high-dimensional search space. The initial population can only explore a limited area of the search space. Then the MOEAs can just obtain a small part of the Pareto front. With regard to CMOPs, it is challenging to handle both multiple objectives optimization and constraint satisfaction.

We apply MOEAs to optimization problems in the context of offloading. Mobile cloud computing (MCC) [99] is emerging as a computing paradigm that combines the strength of cloud computing and the convenience of mobile networks. With the progress of the Internet of Things (IoT) and 5G communications, mobile computing has undergone a shift from centralized MCC to mobile edge computing (MEC) [1]. More and more computation-intensive and time-sensitive applications have been developed for mobile devices, e.g., face recognition, augmented reality and interactive gaming. Due to the insufficient computing ability as well as the limited battery power of mobile devices, the quality of service (QoS) of these complicated applications cannot be satisfied if they are handled locally. Computation offloading in MCC and MEC has emerged as a potential method to solve these problems, where the complicated computing tasks can be chosen to be offloaded to a central cloud or an edge cloud. Fig. 1.1 presents a generic mobile edge-cloud offloading architecture. The applications can be offloaded to the cloud through 4G/5G or WiFi networks. The computation offloading in MCC and MEC aims to shorten response time, save energy consumption and decrease the monetary cost. Obviously, offloading decision making problems are MOPs. How to establish the multi-objective computation offloading models and apply the efficient MOEAs to solve the models will be studied.

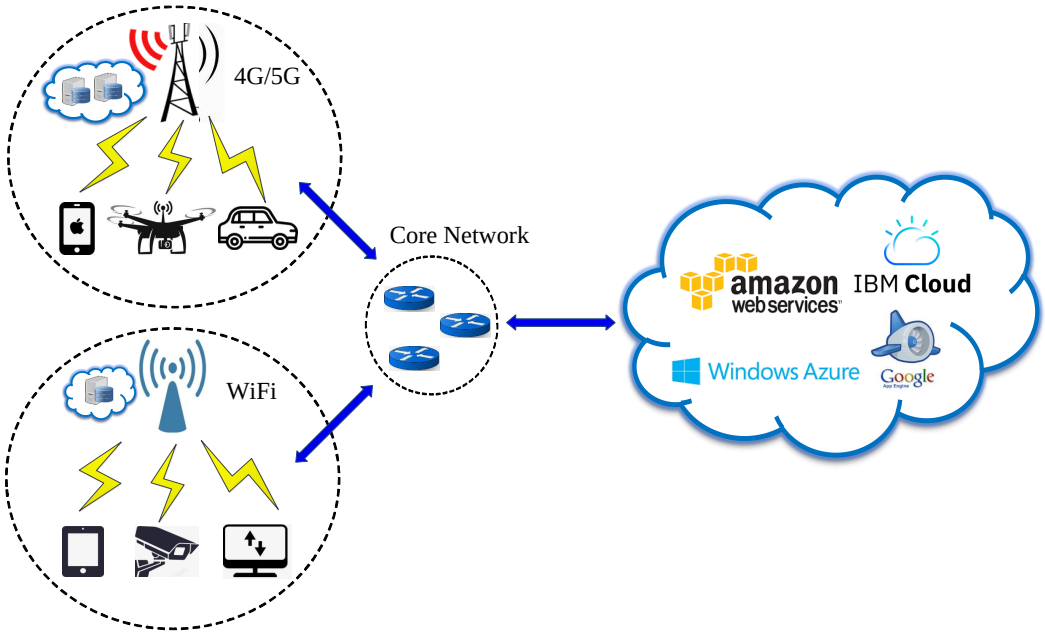


Figure 1.1: Mobile edge-cloud offloading architecture

1.2 Main Research Challenges

A variety of multi-objective evolutionary algorithms (MOEAs) have been developed for multi-objective optimization problems (MOPs), it still faces many challenges to improve convergence and diversity, especially for many-objective optimization, constrained optimization and large-scale optimization. From another perspective, existing MOEAs may encounter difficulties in solving different multi-objective computation offloading optimization problems in MCC and MEC. As shown in Fig. 1.2, several research challenges are summarized as follows:

- ***Multi-objective Optimization***: Although different MOEAs have been proposed to solve MOPs, how to further improve the convergence and diversity of algorithms for solving normalized and scaled MOPs needs to be investigated. In addition, traditional MOEAs may be suitable for MOPs with regular Pareto fronts, while they cannot solve the complex MOPs with irregular Pareto fronts. For instance, the performance of decomposition-based algorithms strongly depends on Pareto front shapes [45]. The decomposition-based algorithms utilize a decomposition method to decompose a MOP into several subproblems based on the weight vectors provided, resulting in the performance of the algorithms being highly dependent on the unifor-

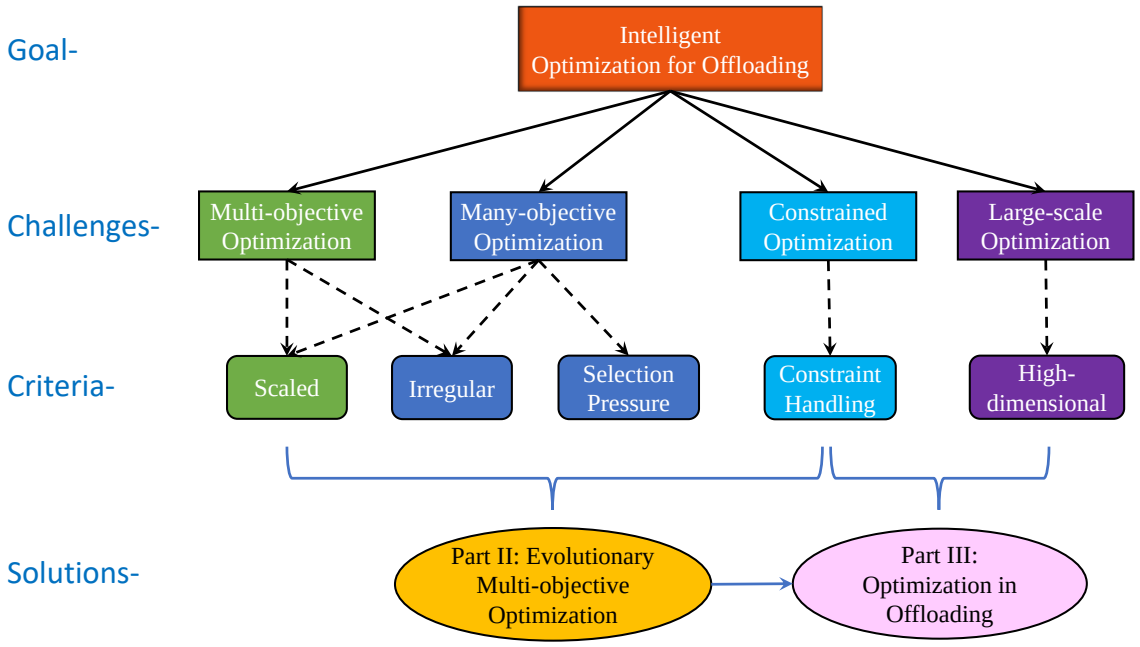


Figure 1.2: Main research challenges and solutions in this dissertation

mity between the problem's optimal Pareto front and the distribution of the specified weight vectors [65]. However, weight vector generation is generally based on a simplex lattice design [22], which is suitable for regular Pareto fronts (i.e., simplex-like fronts) but not for other irregular Pareto fronts (e.g., inverted, degenerated, disconnected, etc.) [63].

- ***Many-objective Optimization***: Many-objective optimization problems (MaOPs) consist of more than three objectives, more and more solutions are non-dominated by each other. The traditional Pareto dominance-based methods cannot work well for MaOPs, which face no strong selection pressure towards the Pareto front [22]. Then some MOEAs have good performance for solving normalized problems, but they may have drawbacks for dealing with problems with disparately scaled objectives. The entire Pareto front needs a huge number of solutions. We want to find an adaptive algorithm to solve normalized and scaled MaOPs for obtaining a set of solutions to represent the Pareto front. Furthermore, MaOPs with irregular Pareto fronts are also difficult to be solved [63].
- ***Constrained Optimization***: Constrained multi-objective optimization problems (CMOPs) have a set of equality and/or inequality constraints. It is challenging to solve CMOPs because of the difficulty in striking a good balance of optimizing objectives and satisfying constraints

simultaneously. In addition, classical constrained multi-objective evolutionary algorithms (CMOEAs) may face difficulties in solving complicated CMOPs, such as CMOPs with large infeasible regions [32], CMOPs with adjustable difficulties [34] and CMOPs with both decision and objective constraints [66]. We will design an efficient constrained algorithm to solve these different complicated CMOPs.

- ***Large-scale Optimization***: Large-scale multi-objective optimization problems (LSMOPs) include a large number of decision variables [12]. The search space increases exponentially with the growth of the number of decision variables. The first challenge for optimization algorithms is to search the whole decision space using a limited population size [119]. These limited algorithms can only search in a small portion of the high-dimensional search space, resulting in obtaining a small part of the Pareto front. The second challenge for MOEAs is that it is difficult to explore the high-dimensional space with limited computational resources. The algorithms need enough function evaluations to search the whole decision space.

We try to propose effective and efficient multi-objective evolutionary algorithms (MOEAs) to solve these challenges in different kinds of multi-objective optimization problems (MOPs). Based on these multi-objective optimization theories and algorithms, we apply and develop multi-objective optimization methods to solve different multi-objective offloading optimization problems, such as constrained and large-scale offloading optimization problems.

1.3 Contributions

The main contributions of this thesis are to analyze and improve the performance of MOEAs and adopt evolutionary multi-objective optimization methods for solving computation offloading problems in collaborative MCC and MEC. According to Fig. 1.2, the main contributions include two parts (**Part II** and **Part III**). In the second part of the thesis, we propose four efficient multi-objective optimization algorithms to solve a variety of MOPs and MaOPs coming from the widely used benchmark suites. Then in the third part we apply and develop three optimization methods for dealing with different multi-objective computation offloading problems in collaborative edge-cloud computing.

The major contributions of this thesis are summarized as follows:

Part II: Evolutionary Multi-objective Optimization

- **Multi-objective Optimization Algorithm**: In order to efficiently solve normalized and scaled MOPs, we design a multi-objective artificial bee colony algorithm based on decomposition. We use a novel reproduction operator inspired by the artificial bee colony algorithm to improve

the convergence. Then a modified Tchebycheff approach is adopted to achieve higher diversity of the solutions. Further, an adaptive normalization operator can be applied to solve differently scaled problems. The proposed algorithm can obtain a well-converging and well-diversified set of solutions for MOPs.

- **Many-objective Optimization Algorithm**: In order to efficiently solve normalized and scaled MaOPs, we develop a decomposition-based evolutionary algorithm with adaptive weight vectors. For dealing with disparately scaled problems, a strategy is adopted to tune weight vectors according to the range of each objective concerning candidate solutions. Based on the adaptive weight vectors, we compare the existing six popular decomposition approaches and choose the best suitable one. Even more, one novel replacement strategy is adopted to attain the balance between convergence and diversity for MOPs and MaOPs.
- **Adaptive Algorithm for Irregular Pareto Fronts**: In order to efficiently solve MOPs and MaOPs with irregular Pareto fronts, we propose a novel archive maintenance for adapting weight vectors to improve the performance of the decomposition-based evolutionary algorithms. An archive is used to store non-dominated solutions. A novel archive maintenance strategy is applied to avoid the dominance resistant solutions, as well as retain the good diversity of non-dominated solution set. Furthermore, guided from the information of the archive, an adaptive weight vector method is designed to solve problems with various Pareto fronts (the simplex-like, the inverted, the disconnected, the degenerated, the scaled, the mixed, the high dimensional).
- **Constrained Optimization Algorithm**: In order to efficiently solve different constrained MOPs (CMOPs), we put forward three tailored constrained multi-objective optimization algorithms. The search process of the proposed algorithms is divided into two stages. In the first stage, we use a multi-objective evolutionary algorithm to search the unconstrained solutions without considering any constraints. In the second stage, a constraint handling mechanism is used to search the constrained solutions. The two-stage search method helps solutions to jump across large infeasible regions. The proposed constrained algorithms can be used to deal with more complicated CMOPs.

Part III: Optimization in Offloading

- **Constrained Offloading Optimization**: For dealing with constrained computation offloading optimization problems in IoT-edge-cloud networks, the three tailored constrained MOEAs (CMOEAs) in **Chapter 6** of **Part II** are applied. First of all, a constrained multi-objective computation offloading model is established to satisfy time and energy consumption require-

ments. Then a multi-server multi-user multi-task computation offloading experimental scenario is used to evaluate the performance of three proposed algorithms. The experimental results demonstrate the effectiveness and superiority of the proposed algorithms.

- **Large-scale Offloading Optimization**: For solving large-scale computation offloading problems in collaborative edge-cloud computing, two evolutionary large-scale sparse multi-objective optimization algorithms are developed. To begin with, a large-scale multi-objective computation offloading model is established, where the offloading decision is represented as a binary encoding. Considering the large-scale and sparsity property of the computation offloading model, the restricted Boltzmann machine (RBM) is applied to reduce the dimensionality and learn the Pareto-optimal subspace. In addition, the contribution score of each decision variable is assumed to generate new offspring solutions. The proposed algorithms are compared with other representative algorithms and offloading strategies on a number of test problems with different scales.
- **Dynamic and Secure Offloading Optimization**: For tackling dynamic and secure computation offloading problems when considering compression, security and mobility in collaborative MCC and MEC, a novel multi-objective computation offloading evolutionary algorithm (MCOEA) is proposed. A dynamic and secure multi-objective computation offloading model is built with the consideration of compression, security and mobility. The designed binary crossover and mutation operators in the proposed MCOEA are applied to improve the convergence and diversity of the solutions.

1.4 Thesis Structure

This thesis consists of four parts. In the first part we generally introduce the research topics and background of the thesis as well as some related work. **Part I** has the following structure:

In **Chapter 1**, the main research topics and challenges are described. The major contributions are summarized and the structure of this thesis is also organized.

In **Chapter 2**, we present the basic principles and some related work that are needed for this work. First the basic concepts of multi- and many-objective, constrained and large-scale optimization are introduced. The general framework of the MOEA is described. The characteristics of benchmark suites for multi-objective optimization are summarized. Some of the most widely used evaluation metrics are adopted to test the performance of MOEAs. Then we discuss the principles of computation offloading optimization. What's more, we present and analyze some related work about evolutionary multi-objective optimization and computation offloading policies.

We propose four MOEAs for solving MOPs and MaOPs in the second part. **Part II** is structured as follows:

In **Chapter 3**, we design a multi-objective artificial bee colony algorithm using the decomposition approach for dealing with normalized and scaled MOPs.

In **Chapter 4**, we propose a decomposition-based evolutionary algorithm with adaptive weight vectors for solving the normalized and scaled MaOPs.

In **Chapter 5**, we develop a novel archive maintenance method for adapting weight vectors in the decomposition-based algorithms for MOPs and MaOPs with irregular Pareto fronts.

In **Chapter 6**, we propose three CMOEAs with better versatility for different kinds of CMOPs.

The third part utilizes three evolutionary multi-objective methods for dealing with computation offloading problems in collaborative edge-cloud computing. **Part III** is organized as follows:

In **Chapter 7**, the three CMOEAs illustrated in **Chapter 6** are adopted to tackle constrained computation offloading problems in IoT-edge-cloud computing networks.

In **Chapter 8**, two evolutionary large-scale sparse multi-objective optimization algorithms are developed to solve edge-cloud computation offloading problems.

In **Chapter 9**, a novel multi-objective computation offloading evolutionary algorithm is developed to a solve multi-objective computation offloading model to minimize time and energy consumption with the consideration of compression, security and mobility.

Part IV shows the concluding remarks.

In **Chapter 10**, this thesis is summarized and concluded. We also provide an outlook on some future research directions.

Chapter 2

Background and Related Work

In this chapter, some background knowledge and related work about evolutionary multi-objective optimization and computation offloading optimization are briefly described. The basic concepts of multi-objective optimization are introduced in Section 2.1. Section 2.2 presents the general principles of multi-objective evolutionary algorithms. Then the benchmark suites and evaluation metrics for multi-objective optimization are introduced in Section 2.3 and Section 2.4, respectively. Afterwards, the basic principles of computation offloading optimization in mobile cloud/edge computing are described in Section 2.5. Some related work and recent advances about evolutionary multi-objective optimization and computation offloading optimization are introduced in Section 2.6. The final section of this chapter gives a short summary.

2.1 Multi-objective Optimization

In this section, we introduce some basic concepts about multi- and many-objective optimization, large-scale optimization, and constrained multi-objective optimization, respectively.

Multi-objective Optimization

Many real-world engineering optimization problems involve the simultaneous satisfaction of multiple objectives which are often in conflict, which means the improvement of one objective value can result in the degradation of other objectives. For instance, the construction of a car is a typical example. The car can be designed to be fast but at a higher price, or cheap, which might decrease its maximum speed. Speed and price can be two conflicting objectives. Instead of finding a single optimal solution as in single-objective optimization, multi-objective optimization problems have a

set of tradeoff solutions to satisfy all conflicting objectives. A multi-objective optimization problem (MOP) can be defined as follows [76]:

$$\begin{aligned} \min \quad & F(\boldsymbol{x}) = \left(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})\right)^T \\ s.t. \quad & g_i(\boldsymbol{x}) \leq 0, \;\; i = 1, 2, \cdots, p \\ & h_j(\boldsymbol{x}) = 0, \;\; j = 1, 2, \cdots, q \\ & \boldsymbol{x} \in \Omega \subseteq R^n \end{aligned} \tag{2.1}$$

where Ω denotes the decision space and $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is an n-dimensional decision vector. $F : \Omega \rightarrow \Theta \subseteq R^m$ represents an m-dimensional objective vector and Θ is the objective space. $g_i(\boldsymbol{x}) \leq 0 \;\; (i = 1, 2, \cdots, p)$ refers to i-th inequality constraint and $h_j(\boldsymbol{x}) = 0 \;\; (j = 1, 2, \cdots, q)$ refers to the j-th equality constraint.

The MOP consists of m objective functions, sometimes also called fitness functions, which have to be minimized or maximized. Without loss of generality, we assume that all objective functions are minimized. It is noted that MOPs having two and three objective functions without considering constraints are called multi-objective optimization, while MOPs with more than three objective functions are often referred to as many-objective optimization problems (MaOPs). In particular, MOPs with a set of equality and/or inequality constraints are called constrained multi-objective optimization problems (CMOPs).

In the following, four important definitions for MOPs are given [76].

Definition 2.1.1. (Pareto dominance) A decision vector $\boldsymbol{x}^0 = \left(x_1^0, x_2^0, \cdots, x_n^0\right)$ is said to dominate a decision vector $\boldsymbol{x}^1 = \left(x_1^1, x_2^1, \cdots, x_n^1\right)$, denoted by $\boldsymbol{x}^0 \prec \boldsymbol{x}^1$, if and only if the following two conditions are satisfied: (1) for all objective functions, the values $f\left(\boldsymbol{x}^0\right)$ are not greater than $f\left(\boldsymbol{x}^1\right)$ and (2) there exists at least one objective function where $f\left(\boldsymbol{x}^0\right)$ is less than $f\left(\boldsymbol{x}^1\right)$.

$$\begin{cases} f_i\left(\boldsymbol{x}^0\right) \leq f_i\left(\boldsymbol{x}^1\right), \;\; \forall i \in \{1, 2, \cdots, m\} \\ f_j\left(\boldsymbol{x}^0\right) < f_j\left(\boldsymbol{x}^1\right), \;\; \exists j \in \{1, 2, \cdots, m\} \end{cases} \tag{2.2}$$

Definition 2.1.2. (Pareto optimal solution) A solution vector $\boldsymbol{x}^0 = \left(x_1^0, x_2^0, \cdots, x_n^0\right)$ is called a Pareto optimal solution, if and only if there exists no other solution $\boldsymbol{x}^1 \in \Omega$ which dominates solution $\boldsymbol{x}^0$.

$$\neg \exists \boldsymbol{x}^1 : \boldsymbol{x}^1 \prec \boldsymbol{x}^0 \tag{2.3}$$

Definition 2.1.3. (Pareto optimal solution set) The set of Pareto optimal solutions is defined as $PS = \left\{ \boldsymbol{x}^0 \left| \neg \exists \boldsymbol{x}^1 \prec \boldsymbol{x}^0 \right. \right\}$.

Definition 2.1.4. (Pareto front) The Pareto optimal solution set in the objective space is called Pareto front, denoted by $PF = \{F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})) | \boldsymbol{x} \in P_s\}$.

The solutions in the Pareto optimal set are non-dominated solutions, which are also not dominated by any other solutions in the decision space. The goal of MOP is to obtain final non-dominated solutions with better convergence and diversity, where the convergence means that the achieved non-dominated solutions in the objective space approximate the true PF closely, and diversity means that the corresponding non-dominated solutions are uniformly distributed over the PF. Fig. 2.1 shows an example of PF of a two-objective optimization problem. The orange circles represent the dominated solutions in the search process, the blue circles denote the non-dominated solutions, and the black line is assumed as an optimal front in the objective space.

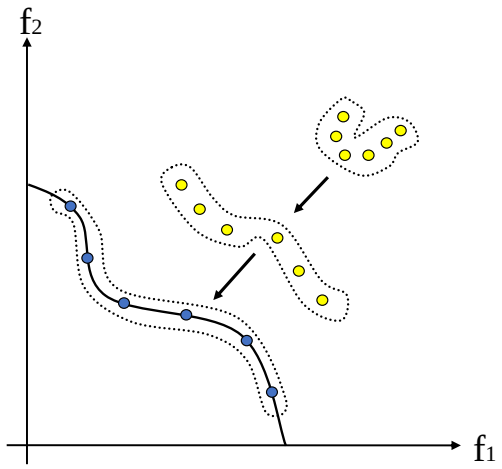


Figure 2.1: The PF of two-objective optimization problem

When solving CMOPs with equality constraints, we often relax the equality constraint with an infinitesimal positive value δ and convert the equality constraints into inequality constraints, which can be expressed as:

$$h_j(\boldsymbol{x})' \equiv \delta - |h_j(\boldsymbol{x})| \geq 0 \tag{2.4}$$

In order to deal with CMOPs with different inequality and quality constraints, the overall con-

straint violation of each solution $\boldsymbol{x}$ can be calculated as:

$$CV\left(\boldsymbol{x}\right)=\sum_{i=1}^{p}\left|\min\left\{g_i\left(\boldsymbol{x}\right),0\right\}\right|+\sum_{i=1}^{q}\left|\min\left\{h_j(\boldsymbol{x})',0\right\}\right| \tag{2.5}$$

where $\boldsymbol{x}$ is a feasible solution if $CV\left(\boldsymbol{x}\right)=0$, otherwise it is infeasible.

Many-objective Optimization

Multi-objective optimization problems (MOPs) with more than three objectives are called many-objective optimization problems (MaOPs). With the growth of the number of objective functions, more and more solutions are most likely non-dominated by each other, the traditional Pareto dominance concept encounters difficulties in selecting non-dominated solutions to improve the convergence [74]. To address this effect, some modified Pareto dominance concepts [106, 110] have been proposed to enhance the selection ability for solving MaOPs. The main idea of these modified Pareto dominance methods is to relax the requirements of traditional Pareto dominance. In this way, the non-dominated solutions can be selected easily under the new environment.

On the other hand, some other many-objective optimization algorithms [22, 60, 113] try to rely on the weight vectors (or reference directions) to decompose a MaOP into a series of scalar optimization subproblems. Each weight vector represents a search direction, we only need to optimize one subproblem under this weight vector. With the help of these search directions, different non-dominated solutions can be obtained to approach the Pareto front.

In addition, the number of objective functions in many-objective optimization research areas mainly focuses on 5, 10, 15.

Large-scale Optimization

In this work, large-scale optimization refers to MOPs and MaOPs with a large number of decision variables. Single-objective and multi-objective optimization problems may both have a large number of decision variables, here we analyze large-scale multi-objective optimization problems (LSMOPs) [12]. In the literature, large-scale may represent different numbers of decision variables. Any problem with more than 100 decision variables can be called large-scale.

When the number of decision variables increases, the search space increases exponentially. The first challenge for metaheuristic methods is to search the whole decision space using a limited population size [119]. Only a small portion of the high-dimensional search space can be explored by the algorithm, and also a small part of the Pareto front can be obtained. Thus the diversity of the final

obtained non-dominated solution set is poor with regard to the optimal PF. On the other hand, large scale properties may influence the function of common genetic operators. The mutation rate in the mutation operator is often set to $1/n$, where n is the number of decision variables. Considering that n is very large, i.e. $n = 1000$, the mutation rate is 0.001, in this way the solutions will easily fall into local optima.

The second challenge for MOEAs is that it is difficult to explore the high-dimensional space with limited computational resources. With increasing dimensionality of the decision variables, the decision space is becoming huge quickly. The algorithms not only need a certain population size but also need enough function evaluations to help the algorithms thoroughly search the whole decision space. For solving these two problems, most large-scale optimization algorithms try to reduce the number of decision variables for solving large-scale MOPs.

2.2 Multi-objective Evolutionary Algorithm

Simple optimization problems can be addressed by traditional methods such as integer programming, dynamic programming, and enumeration methods. However, some complex optimization problems are NP-hard or cannot be formulated mathematically directly. The traditional mathematical methods face difficulties in dealing with such problems. The metaheuristic optimization algorithms which have been inspired by biological or physical processes provide a way to obtain a suboptimal or optimal solution during the polynomial time.

A large variety of metaheuristic optimization algorithms has been proposed to solve different optimization problems over the years. Some representative algorithms are Hill Climbing [53], Simulated Annealing (SA) [52, 53], Genetic Algorithms (GA) [19], Particle Swarm Optimization (PSO) [50], and Ant Colony Optimization (ACO) [8]. While the former two methods seem to be local search mechanisms, the latter three methods are representative algorithms that belong to global optimization. A Genetic Algorithm (GA) is inspired by the evolution of species, the new offspring solutions will be generated by crossover and mutation operators, and the better solution (offspring) will survive through natural selection. The Particle Swarm Optimization (PSO) algorithm is inspired by bird predation behavior, which finds the optimal solution through collaboration and information sharing between individuals in the population. PSO uses a position in the decision space to represent a solution, and utilize concepts like velocity to guide the search. Ant Colony Optimization (ACO) algorithm is a probabilistic algorithm, which is inspired by the behavior of ants finding paths in the process of searching for food. Ants will release a substance called pheromone on the path, and they will walk along a path with a higher concentration of pheromone. GA, PSO and ACO

are all population-based metaheuristic algorithms, and they fall into the category of evolutionary computation [77].

An Evolutionary Algorithm (EA) [82, 94] applies a population to retain a set of solutions and approximates the optimal or Pareto optimal solution set by gradually improving the performance of current solutions in the population. The new candidate solutions are generated by existing solutions, and they are selected as better solutions by natural selection mechanism. The natural selection mechanism is also often called "survival of the fittest" based on the theory of evolution [16]. EAs consist of two categories, single-objective EAs and multi-objective EAs. In this work, we focus on the topic of multi-objective evolutionary algorithms (MOEAs) [14].

Algorithm 1 presents the basic framework of a MOEA. At first, a random population is initialized. The solutions in the population are evaluated by the objective functions. After that, a loop operation is carried out until the termination is satisfied. The existing solutions with better fitness function values are selected into a mating pool. Then the parents are selected from the mating pool by a tournament selection mechanism. The new solutions are generated by crossover and mutation operations of parents. After the new solutions are evaluated, an environmental selection operation is used to determine which solution can survive to the next generation. As the iteration progresses the algorithm approaches better non-dominated solutions until the Pareto front is found.

Algorithm 1: Basic framework of a multi-objective evolutionary algorithm

Input: Optimization problem
Output: Solution population P

$P \leftarrow$ initial random population;
$Evaluate(P)$;
while *the termination criterion is not satisfied* **do**
    $P' \leftarrow MatingSelection\,(P)$;
    $Q \leftarrow Crossover\,(P')$;
    $Q \leftarrow Mutation\,(Q)$;
    $Evaluate(Q)$;
    $P \leftarrow EnvironmentalSelection\,(P, Q)$;
end

In MOEAs there exist different kinds of operations to produce offspring solutions, and also a variety of environmental selection mechanisms to select better solutions in the evolutionary process. It is worth noting that two solutions that have similar objective functions share a similar representation of decision variables. In that way, the newly generated offspring solutions can inherit the characteristics of the parents.

2.3 Benchmark Suites

This section gives a brief overview of existing benchmark suites for multi- and many-objective optimization, constrained and large-scale multi-objective optimization, including the ones used in this thesis. Most of the benchmark suites have been widely used to evaluate different kinds of algorithms. The problems have different characteristics, which can test a variety of performance of algorithms for solving different problems. A series of scalable test problems have been designed and these benchmark instances are mathematical problems. For example, Zitzler, Deb and Thiele [122] designed ZDT test problems and used their names' acronyms to name the test suite ZDT. The ZDT problems aim to find x to minimize two functions $f_1(x)$ and $f_2(x)$ simultaneously. Each test problem in ZDT test suite involves a particular feature that can cause difficulty in the evolutionary optimization process. By investigating these different features, it is possible to know different techniques are suited for different kinds of problems. Five widely used two-objective ZDT test problems are formalized as:

- ZDT1

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left[1-\sqrt{f_1(x)/g(x)}\right] \\ g(x) &= 1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \end{aligned} \tag{2.6}$$

  where $x=(x_1,\ldots,x_n)^T \in [0,1]^n$, and $n=30$. Its Pareto front (PF) is convex.

- ZDT2

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left[1-(f_1(x)/g(x))^2\right] \\ g(x) &= 1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \end{aligned} \tag{2.7}$$

  where $n=30$. Its PF is concave. ZDT1 and ZDT2 can test an algorithm's performance for searching the Pareto fronts with convex and concave shapes, respectively.

- ZDT3

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left[1-\sqrt{f_1(x)/g(x)}-(f_1(x)/g(x))\sin(10\pi x_1)\right] \\ g(x) &= 1+9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1) \end{aligned} \tag{2.8}$$

  where $n=30$. Its PF is disconnected. This problem can test an algorithm's performance for

searching the disconnected Pareto fronts.

- ZDT4

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= g(x)\left[1-\sqrt{f_1(x)/g(x)}\right] \\ g(x) &= 1+10(n-1)+\sum_{i=2}^{n}\left[x_i^2-10\cos(4\pi x_i)\right] \end{aligned} \tag{2.9}$$

where $x = (x_1, \ldots, x_n)^T \in [0,1] \times [-5,5]^{n-1}$, and $n = 10$. It has many local PFs. This problem can test an algorithm's ability for searching the Pareto fronts with many local optima.

- ZDT6

$$\begin{aligned} f_1(x) &= 1-\exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(x) &= g(x)\left[1-(f_1(x)/g(x))^2\right] \\ g(x) &= 1+9\left[\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1)\right]^{0.25} \end{aligned} \tag{2.10}$$

where $n = 10$. Its PF is concave. The Pareto optimal solutions are distributed nonuniformly in the PF. This problem can test an algorithm's ability for searching for a set of solutions distributed uniformly in the Pareto fronts.

The true PFs of ZDT test problems are shown in Fig. 2.2. Similar to the ZDT test suite, the mathematical description and Pareto fronts of other test problems can be located in the original papers. Afterward we mainly analyze the characteristics and scalability of other benchmark suites.

The DTLZ (Deb, Thiele, Laumanns and Zitzler [25] designed and used their names' acronyms to denote) benchmark suite is designed in [25], which includes nine test problems DTLZ1-9, but the first seven problems DTLZ1-7 are most used. These problems are scalable in the number of decision variables and objective functions and have the ability to control the difficulties in both converging to the true PF and maintaining a widely distributed set of solutions in the PF. The DTLZ benchmark families are suitable for testing the performance of algorithms on multi- and many-objective optimization and large-scale optimization.

The SZDT (Scaled ZDT) and SDTLZ (Scaled DTLZ) benchmark suites are modifications of the ZDT and DTLZ test suites [22], which have disparately scaled objective functions. To illustrate, if the scaling factor is a, the objective value f_i of SZDT and SDTLZ is equal to the original objective value f_i of ZDT and DTLZ multiplied by a^{i-1}, that is:

$$f_i = a^{i-1} f_i, \quad i = 1, 2, \cdots, m, \quad a \neq 0 \tag{2.11}$$

where m is the number of objective functions. Each test problem of the ZDT and SZDT suites

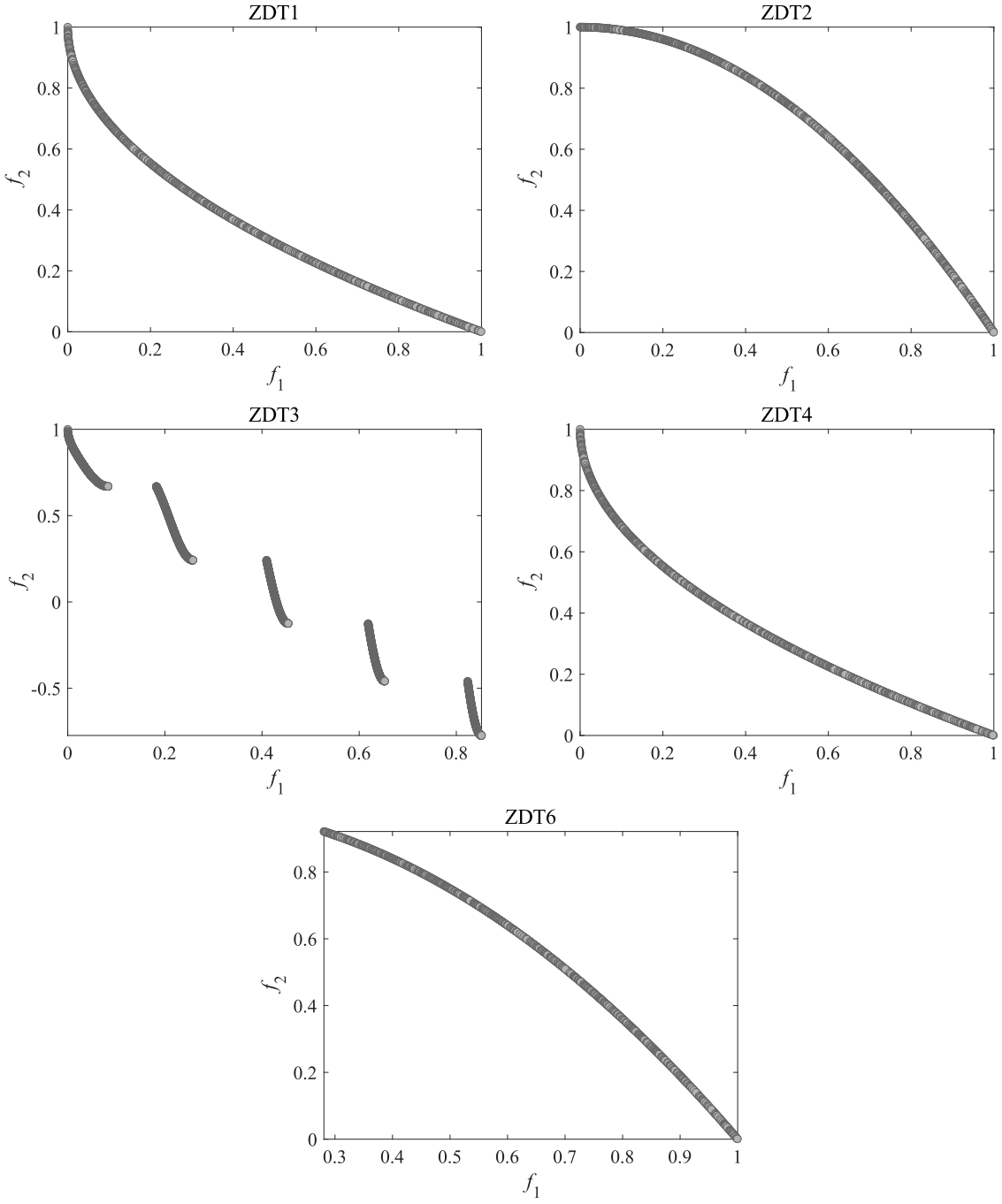


Figure 2.2: The true Pareto fronts of five ZDT test instances

only has 2 objective functions. DTLZ and SDTLZ test problems can be scaled to any number of objectives. The different number of objective functions corresponds to different scaling factors. The scaling factors used in SZDT and SDTLZ with the different number of objectives are listed in Table 2.1.

Table 2.1: The scaling factor

No. of objectives (m)	Scaling factor (a)
2	10
3	10
5	10
8	3
10	2

The IDTLZ (Inverted DTLZ) and CDTLZ (Convex DTLZ) benchmark suites are modifications of the DTLZ test suites. IDTLZ1-4 problems have inverted PF shapes compared with DTLZ1-4 test problems. CDTLZ2 and CDTLZ4 have convex PF shapes while the original PFs of DTLZ2 and DTLZ4 are concave.

The MaF benchmark suite has been developed for the IEEE Conference of Evolutionary Computation (CEC) Competition on Evolutionary Many-Objective Optimization [13]. The MaF test suite consists of fifteen test problems, which are mainly modified DTLZ [25], WFG [44], LSMOP [12] test problems.

The LIR-CMOP (Constrained Multi-objective Optimization Problem with Large Infeasible Regions) benchmark suite is proposed in [32], which has fourteen test problems LIR-CMOP1-14. All the test problems have large infeasible regions with respect to constraint functions. The objective functions have two components: shape functions and distance functions, where shape functions are used to set the shape of PFs and distance functions are applied to control the difficulty of convergence. In addition, the feasible regions of LIR-CMOP1-4 are very small. LIR-CMOP5 and LIR-CMOP6 have convex and concave PFs, respectively. The PFs of LIR-CMOP7 and LIR-CMOP8 are located on their constraint boundaries. The PFs of LIR-CMOP9-12 have a number of disconnected segments. LIR-CMOP13 and LIR-CMOP14 have three objectives.

The DAS-CMOP (Difficulty-adjustable and Scalable CMOP) benchmark suite has been developed in [34]. It has a set of nine CMOPs (DAS-CMOP1-9). DAS-CMOPs adopted three types of difficulty, feasibility-hardness, convergence-hardness and diversity-hardness, to characterize the constraint functions in CMOPs. The difficulty can be defined by the three primary constraint functions with different parameters. The number of objectives in DAS-CMOPs can be scaled. The PFs

of DAS-CMOPs include convex, concave and discrete PFs.

The DOC (Decision and Objective Constraints) benchmark suite is constructed in [66]. It consists of nine test problems DOC1-9. DOC considers both the decision and objective constraints simultaneously. Various decision constraints in DOC (e.g., inequality and equality, linear and nonlinear constraints) are collected, while the controllable objective constraints make the PFs have diverse characteristics (e.g., continuous, discrete, mixed and degenerate). DOC poses a great challenge for CMOEAs to find well-converged and well-distributed feasible solutions.

The LSMOP (Large-scale MOP) benchmark suite is proposed in [12]. It includes nine test problems, LSMOP1-9, for large-scale multi- and many-objective optimization. These test problems are scalable in terms of the number of decision variables and objective functions. The mixed separability between decision variables and the non-uniform correlation between decision variables and objective functions are also considered in the test problems.

The SMOP (Sparse MOP) benchmark suite is designed in [90]. It includes eight test problems SMOP1-8 for large-scale sparse MOPs. These test problems are designed to have various landscape functions for providing various difficulties for algorithms in obtaining Pareto optimal solutions.

Table 2.2 shows different properties of a list of benchmark suites.

Table 2.2: Different properties of a list of benchmark suites

Benchmark suites	Benchmark problems	Properties					
		multi	many	large	constrained	scaled	irregular
ZDT	ZDT1-4, and ZDT6	✓					✓
DTLZ	DTLZ1-7	✓	✓			✓	✓
SZDT	SZDT1-4, and SZDT6	✓				✓	✓
SDTLZ	SDTLZ1-7	✓	✓			✓	✓
IDTLZ	IDTLZ1-4	✓					✓
CDTLZ	CDTLZ2 and CDTLZ4	✓					✓
MaF	MaF1-15	✓	✓			✓	✓
LIR-CMOP	LIR-CMOP1-14	✓			✓		✓
DAS-CMOP	DAS-CMOP1-9	✓	✓		✓		✓
DOC	DOC1-9	✓			✓		✓
LSMOP	LSMOP1-9	✓	✓	✓			
SMOP	SMOP1-8	✓	✓	✓			

2.4 Evaluation Metrics

In single-objective optimization, the fitness function values are often used to compare the performance of achieved solutions. Different from single-objective optimization, the result of multi-objective optimization is to find a Pareto optimal solution set instead of one solution. A set of solutions represents a tradeoff between different objective functions. Therefore, certain metrics (also called performance indicators) are needed to evaluate the performance of multi-objective optimization algorithms by analyzing the obtained Pareto optimal solution set. Different evaluation metrics can be used to measure the convergence and diversity of a solution set. This section mainly introduces three performance indicators Generational Distance (GD) [93], Inverted Generational Distance (IGD) [5, 125] and Hypervolume (HV) [96, 125], which are also used in the experiments of this thesis.

The Generational Distance (GD) indicator can measure the convergence of a solution set. The GD indicator requires a reference set P^* which is assumed to be a sample of the true PF. The GD indicator calculates the average of the shortest Euclidean distance from each solution in the obtained solution set to its closest solution in P^*. GD is defined as follows:

$$\mathrm{GD}\left(S, P^*\right) = \frac{1}{|S|}\sqrt{\sum_{i=1}^{|S|} d_i^2} \tag{2.12}$$

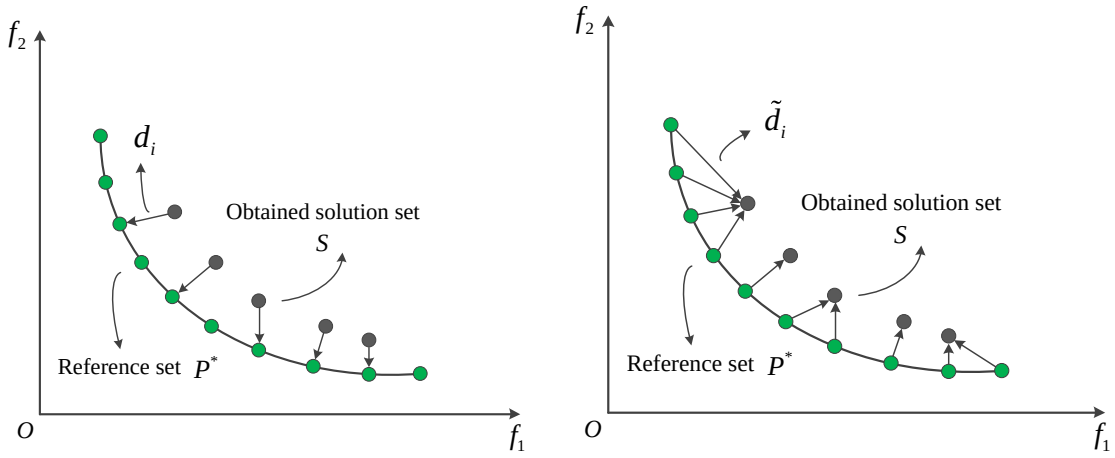
where S is the obtained solution set, $|S|$ denotes the number of solutions in S. d_i is the smallest Euclidean distance between the i-th point of set S and all the members in the set P^*. If the size of P^* is large enough, GD can reflect the information on how close the obtained solutions are to the optimal ones. For scaled problems, GD is often computed after the objective values are normalized.

The Inverted Generational Distance (IGD) indicator can convey information of both convergence and diversity of a solution set. IGD also needs the reference set P^* in the true PF. The IGD indicator calculates the average of the shortest Euclidean distance from each solution in P^* to its closest solution in the obtained solution set S. IGD is formulated as follows:

$$\mathrm{IGD}\left(P^*, S\right) = \frac{1}{|P^*|}\sqrt{\sum_{i=1}^{|P^*|} \tilde{d}_i^2} \tag{2.13}$$

where $\tilde{d}_i$ is the smallest Euclidean distance between the i-th point of set P^* and all the members in the set S. When the size of P^* is large enough, IGD can reflect the information on how close the obtained solutions are to the optimal ones as well as how diverse the obtained solutions are dis-

tributed in the PF. IGD can provide information about convergence and diversity, and it is often used as the performance indicator to compare the performance of multi- and many-objective optimization algorithms [2, 11, 46, 113]. Also for scaled problems, IGD is often calculated after normalizing the objective values. Fig. 2.3 gives an example of calculating GD and IGD on a two-objective optimization problem.



(a) GD calculation (b) IGD calculation

Figure 2.3: GD and IGD calculation

The Hypervolume (HV) indicator is one of the most frequently used metrics [96, 125]. It can reflect the information of convergence as well as diversity. HV measures the volume of the region dominated by the obtained solution set S and bounded by a reference point $r = (r_1, r_2, \cdots, r_m)$, the reference point is dominated by all the obtained solutions. HV is calculated as follows:

$$HV(S, r) = Lebesgue\left(\bigcup_{x \in S} [f_1(x), r_1] \times \cdots \times [f_m(x), r_m]\right) \tag{2.14}$$

where $Lebesgue(S)$ is the Lebesgue measure of a solution set S. Considering the differently scaled objectives in different optimization problems, the objective values of all test problems are normalized before calculating the HV indicator. The reference point can be set as an m-dimensional vector of ones. Fig. 2.4 shows how to calculate HV on a two-objective optimization problem.

Assuming that all objective functions need to be minimized, the obtained solution set is closer to the true PF and further away from the reference point, thus the HV increases. Compared with GD and IGD, the HV indicator does not need a sample of the true PF, and it only has one reference

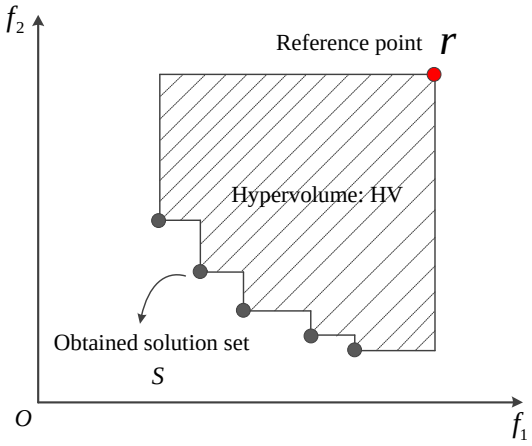


Figure 2.4: HV calculation

point. Therefore, the HV indicator can be used in some real applications whose true PFs are not available. On the other hand, the HV indicator has been applied to some indicator-based multi- and many-objective optimization algorithms [3,4]. Larger HV means better solutions with regard to both convergence and diversity.

2.5 Computation Offloading Optimization

Optimization of application offloading is an interesting real-world problem.

Cloud computing can provide powerful, elastic and scalable computing resources for users [70]. The computation offloading in mobile cloud computing (MCC) [78] takes advantage of cloud resources and the convenience of mobile networks, which helps the mobile devices to offload complicated computation tasks to remote data centers [68]. To reduce the transmission delay in MCC, mobile edge computing (MEC) [51, 72] has been developed as a new solution that performs computation at the edge of the network.

IoT devices have different complex applications and these applications can be divided into a series of tasks. The tasks can be migrated to MCC/MEC servers and then processed on the servers. Computation offloading in MCC/MEC is not always beneficial because of the extra cost of transmitting the tasks from IoT devices to MCC/MEC servers [98]. A suitable offloading decision is needed to determine whether a task is best executed locally or offloaded to different servers. We want to optimize the offloading decision to shorten execution time, reduce energy consumption or achieve a combination of the above two. Hence, the offloading decision making is a multi-objective

optimization problem.

2.6 Related Work

In this section, we introduce some related work of the topics covered in this thesis, including multi- and many-objective optimization, constrained and large-scale multi-objective optimization as well as computation offloading optimization in mobile cloud and edge computing.

2.6.1 Evolutionary Multi-objective Optimization

We present and analyze some related work on multi-objective evolutionary algorithms (MOEAs). Fig. 2.3 shows the classification of some representative optimization algorithms discussed as follows.

Table 2.3: Classification of different optimization algorithms

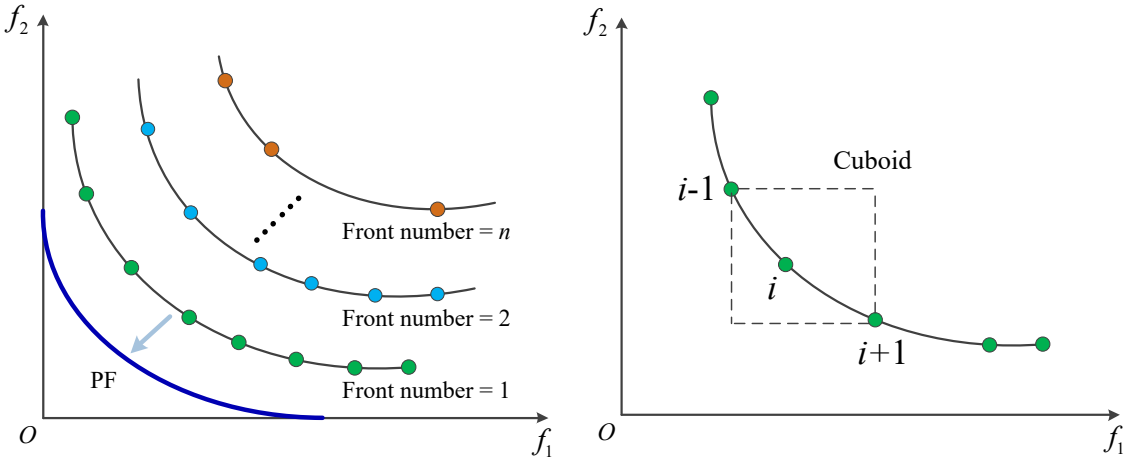
Multi- and many-objective algorithms	Constrained algorithms	Large-scale algorithms
Pareto dominance-based: NSGA-II [24], PESA-II [15], SPEA-II [124], GrEA [106]. Decomposition-based: MOEA/D [113], MOEA/D-M2M [64], RVEA [11]. Indicator-based: IBEA [123], HyPE [3], SMS-EMOA [4], SRA [56].	MOEA/D-IEpsilon [32], C-TAEA [59], CCMO [89], MOEA/D-DAE [118].	Decision variable decomposition: MOEA/DVA [67], LMEA [114]. Problem transformation: WOF [120], LCSA [121], LSMOF [40]. Sparse: MOEA/PSL [88], SparseEA [90].

Multi- and Many-objective Optimization

A diversity of MOEAs has been developed for solving multi-objective optimization problems (MOPs) and many-objective optimization problems (MaOPs) during the last two decades, which can be classified into three categories [55, 112].

The first category is the Pareto dominance-based MOEA, where the Pareto dominance criterion is the main feature to push the candidate solutions to approximate the PF. NSGA-II [24] is a typical Pareto dominance-based MOEA for MOPs, all non-dominated solutions are first identified by the fast non-dominated sorting approach and then the crowding distance strategy is used to preserve population diversity. Fig. 2.5 shows the non-dominated sorting and crowding distance calculating in NSGA-II. PESA-II [15] and SPEA-II [124] are two other similar representative Pareto dominance-based MOEAs with different diversity preservation methods. Due to the loss of selection pressure with the increasing number of objectives, the traditional Pareto dominance-based MOEAs

face the dominance resistance phenomenon with regard to many-objective optimization problems [74]. To address this issue, some modified Pareto dominance relationships have been put forward to solve MaOPs, such as grid dominance-based MOEA (GrEA) [106], fuzzy Pareto dominance-based MOEA (FD-NSGA-II) [41], knee point driven MOEA (KnEA) [115] and θ dominance-based MOEA (θ-DEA) [110]. These new modified Pareto dominance methods relax the restrictions of traditional Pareto dominance to some extent.



(a) Front number

(b) Crowding distance

Figure 2.5: Non-dominated sorting and crowding distance in NSGA-II

The second category is the decomposition-based MOEA, where a MOP can be decomposed into a series of scalar optimization subproblems. MOEA/D [113] is a typical MOEA of this category, the decomposition approaches can decompose a MOP into several subproblems and optimize them simultaneously based on a set of uniformly distributed weight vectors. Owing to the efficient framework, MOEA/D with differential evolution (DE) has been applied for solving complicated Pareto sets [58]. Some other novel decomposition-based MOEAs try to decompose a MOP into different subspaces, such as MOEA/D-M2M [64], RVEA [11], and SPEA/R [48].

The third category is the indicator-based MOEA, where the performance indicators are adopted as criteria to distinguish and select candidate solutions. IBEA [123] and SMS-EMOA [4] are two representative indicator-based MOEAs, where the indicators are designed as predefined binary indicator and hypervolume (HV) indicator. The computational complexity of HV grows exponentially with the increasing number of objectives. HypE [3] adopted the Monte Carlo simulation to estimate the HV value for accelerating the computing speed. Other performance indicators (such as Δ_p [79], IGD [83], and IGD-NS [84]) with lower computational complexity have also been proposed for

MOPs. In addition, combined multiple indicators are also a trend, SRA [56] is a typical MOEA selecting the stochastic ranking strategy to balance different indicators.

Constrained Multi-objective Optimization

A wide range of constrained multi-objective evolutionary algorithms (CMOEAs) have been designed to tackle constrained multi-objective optimization problems (CMOPs). A key issue in CMOEA is to deal with constraints. The penalty function approach is often used to balance objectives and constraints. It converts a CMOP into an unconstrained MOP by adding the overall constraint violation multiplied by a predefined penalty factor to each objective [97]. The constrained NSGA-II [24] adopts the constraint dominance principle to distinguish feasible and infeasible solutions. MOEA/D-IEpsilon [32] combines an improved epsilon constraint handling mechanism with the MOEA/D algorithm [113] to solve CMOPs. C-TAEA [59] maintains a convergence-oriented archive and a diversity-oriented archive simultaneously to retain the balance between the convergence and diversity of solutions. Push and pull search (PPS) [33] divides the search process into two stages: push and pull search, and embeds the MOEA/D algorithm [113] into the PPS framework for tackling CMOPs. CCMO [89] uses a coevolutionary framework of two populations to share information with each other for dealing with CMOPs. Fig. 2.6 illustrates the working procedure of the CCMO algorithm [89]. MOEA/D-DAE [118] develops a detect-and-escape strategy to avoid being trapped in local optima and stuck in an unfeasible area.

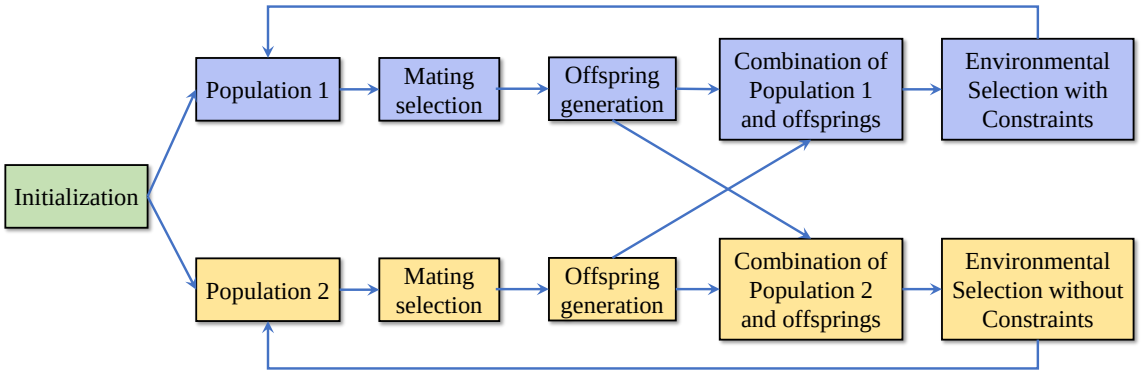


Figure 2.6: Procedure of the CCMO algorithm

Large-scale Multi-objective Optimization

Traditional MOEAs will encounter difficulties in dealing with large-scale MOPs [119]. For solving large-scale MOPs, some tailored MOEAs have been developed based on two main ideas.

The first idea for solving large-scale MOPs is decision variable decomposition, which divides the decision variables into different groups and optimizes each group of decision variables independently. For example, MOEA/DVA [67] adopts control property analysis to decompose the original MOP into a set of sub-MOPs based on position variables and mixed variables. It also uses variable linkage analysis to decompose high-dimensional distance variables into several low-dimensional subcomponents. Each subcomponent in sub-MOPs can be optimized separately. LMEA [114] divides the decision variables into convergence-related and diversity-related variables and optimizes them by different strategies. Fig. 2.7 shows the structure of the LMEA algorithm.

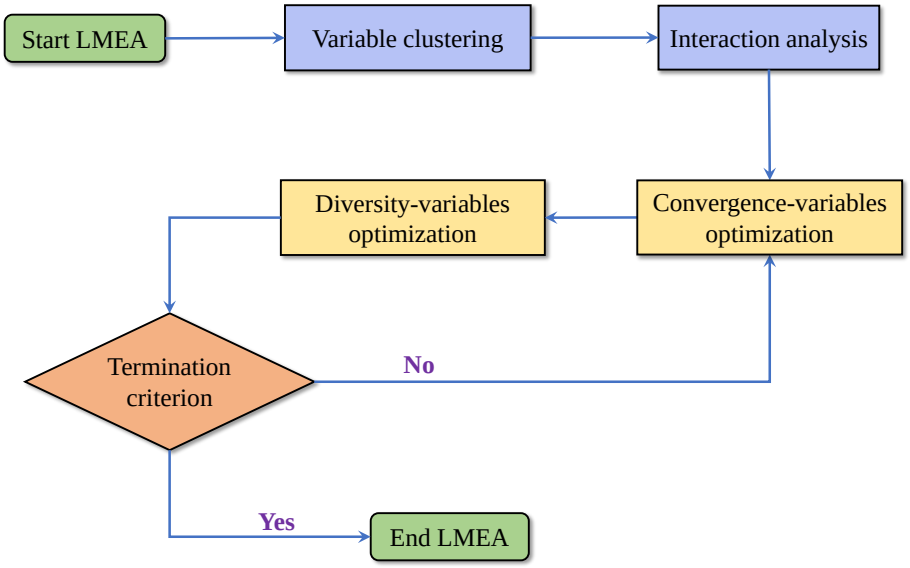


Figure 2.7: Outline of the LMEA algorithm

The second idea for solving large-scale MOPs is problem transformation, which aims to transform the original large-scale MOP into a small-scale problem. Weighted Optimization Framework (WOF) [120] is a generic framework for problem transformation. In WOF, a set of weights is applied to divide the decision variables into different groups. Each weight is related to multiple decision variables. The optimization of all the decision variables can be represented by the optimization of these weights. LCSA [121] uses a linear combination of population members to tackle large-scale MOPs. LSMOF [40] applies two reference points on a solution to search for better solutions in the

large-scale search space. Although these MOEAs are tailored for large-scale MOPs, they cannot be applied to MOPs with binary decision variables and they are shown to be of low efficiency with a large number of function evaluations.

Considering some large-scale MOPs whose Pareto optimal solutions are sparse, MOEA/PSL proposed in [88] uses two unsupervised neural networks (a restricted Boltzmann machine and a denoising autoencoder) to learn a sparse distribution and compact representation of the decision variables. SparseEA designed in [90] integrates real and binary encodings to represent a solution. A new population initialization strategy and genetic operators are suggested to generate solutions with sparsity, which can solve large-scale sparse benchmarks.

2.6.2 Computation Offloading Optimization Schemes

We apply multi-objective evolutionary algorithms to solve the offloading optimization problems. Recently, a range of optimization-based algorithms have been designed to deal with computation offloading problems for mobile cloud computing (MCC) and mobile edge computing (MEC).

Sheikh et al. [80] modeled the effect of parallel execution on multi-site computation offloading in MCC and used an existing genetic algorithm (GA) to solve this multi-site offloading problem. In [54] Kuang et al. established a system model in the MEC environment with multiple users and structured tasks. Then an offloading decision problem was formalized as a cost-minimization problem and a modified genetic algorithm was adopted to solve the offloading problem.

Guo et al. [38] studied the energy-efficient computation offloading scheme in the MEC system with small cell networks. In addition, they formulated the computation offloading problem as a mixed integer non-linear programming problem. Taking advantage of genetic algorithm (GA) and particle swarm optimization (PSO), an optimization method named hierarchical GA and PSO-based computation algorithm was developed to tackle this offloading problem.

An application placement technique proposed in [37] aims to minimize the execution time and energy consumption of IoT applications in edge and fog computing environments. Goudarzi et al. [37] designed a lightweight pre-scheduling algorithm to maximize the number of parallel tasks for execution. Furthermore, an optimized version of the Memetic Algorithm (MA) was put forward to make batch application placement decisions for concurrent IoT applications. The Memetic Algorithm (MA) is a combined algorithm of evolutionary algorithm such as GA with one local search method.

In [104] Xu et al. built a multi-objective optimization model of task offloading and proposed the non-dominated sorting genetic algorithm III (NSGA-III) to solve the offloading model. Peng et al. [73] established an end-edge-cloud collaborative computation offloading model in a heteroge-

neous edge-server environment, with the aim to minimize time and energy consumption. Further, an improved strength Pareto evolutionary algorithm 2 (SPEA2) [124] was adopted to address this model.

2.7 Summary

This chapter presented the background and related work of this thesis. At first, the basic principles of multi- and many-objective optimization, constrained and large-scale multi-objective optimization were given. A brief overview of the MOEA was described. Then the widely used benchmark suites for multi-objective optimization were introduced. The formal definition of evaluation metrics GD, IGD and HV were presented. The basic concepts and properties of computation offloading in MCC and MEC were explained. After that, some related work about evolutionary multi-objective optimization and computation offloading optimization schemes were introduced.

Part II

Evolutionary Multi-objective Optimization

Chapter 3

A Multi-objective Artificial Bee Colony Algorithm

This chapter presents a multi-objective artificial bee colony algorithm based on decomposition (called MOEA/D-ABC) for dealing with normalized and scaled multi-objective optimization problems (MOPs).

Multi-objective evolutionary algorithms (MOEAs) have been developed for solving MOPs [14, 23]. MOEA based on decomposition (MOEA/D) [113] is a novel MOEA framework, which decomposes a MOP into a series of scalar optimization problems. Recently, the MOEA/D framework has achieved great success and received much attention [92]. We try to improve the performance of the MOEA/D from three aspects, i.e., convergence, diversity, and scalability.

In this chapter, first we want to develop other nature inspired meta-heuristics so as to adopt an artificial bee colony (ABC) algorithm as the reproduction operator to improve the convergence of MOEA/D. Second, we substitute the original Tchebycheff approach with a modified Tchebycheff approach for improving diversity. Then, in terms of differently scaled problems, an adaptive normalization mechanism is incorporated into the proposed algorithm.

3.1 Classical Decomposition Approaches

MOEA/D is an efficient algorithm framework approaching the Pareto front. The decomposition methods are able to convert the MOPs into a number of single optimization problems. In addition, the decomposition methods often determine the evolving direction of the MOEAs. The weighted sum approach, the Tchebycheff approach and the penalty-based boundary intersection (PBI) ap-

proach are three widely used decomposition methods in the framework.

In the weighted sum approach, a scalar optimization problem can be described as follows:

$$\min_{x\in\Omega} \; g^{ws}\left(x\,|\lambda\right) = \min_{x\in\Omega} \; \sum_{i=1}^{m} \lambda_i \times f_i\left(x\right) \tag{3.1}$$

where $\lambda = \left(\lambda_1, \lambda_2, \cdots, \lambda_m\right)^T$ is a weight vector and $\sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0, i = 1, 2, \cdots, m$. It has been proved that the weighted sum approach does not work well with non-convex Pareto fronts [113].

In the Tchebycheff approach, a scalar optimization problem can be stated as follows:

$$\min_{x\in\Omega} \; g^{te}\left(x\,|\lambda, z^*\right) = \min_{x\in\Omega} \; \max_{1\leq i\leq m} \left\{\lambda_i \left|f_i\left(x\right) - {z_i}^*\right|\right\} \tag{3.2}$$

where $\lambda = \left(\lambda_1, \lambda_2, \cdots, \lambda_m\right)^T$ is a weight vector and $\sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0, i = 1, 2, \cdots, m$. $z^* = \left(z_1^*, z_2^*, \cdots, z_m^*\right)^T$ is the reference point. Because it is often time-consuming to compute the exact z_i^*, it is estimated by the minimum objective value f_i (i.e., $z_i^* = \min\left\{f_i\left(x\right) | x \in \Omega\right\}, i = 1, 2, \cdots, m$).

A scalar optimization problem of the penalty-based boundary intersection (PBI) approach is defined as follows:

$$\min_{x\in\Omega} \; g^{pbi}\left(x\,|\lambda, z^*\right) = \min_{x\in\Omega} \; \left(d_1 + \theta d_2\right) \tag{3.3}$$

where

$$\begin{cases} d_1 = \dfrac{\left\|\left(z^* - F(x)\right)^T \lambda\right\|}{\|\lambda\|} \\ d_2 = \left\|F\left(x\right) - \left(z^* + d_1 \dfrac{\lambda}{\|\lambda\|}\right)\right\| \end{cases} \tag{3.4}$$

Here θ is a user-predefined penalty parameter. As shown in Fig. 3.1, the line L is passing through the reference point z^* with the direction vector λ. The point y is the projection of $F\left(x\right)$ on the line L. For each solution, a PBI measure value of the form $d_1 + \theta d_2$ is used. The distance between z^* and y is d_1, which denotes the distance along the weight vector and controls convergence. The distance between $F\left(x\right)$ and y is d_2, which denotes the perpendicular distance and controls diversity.

3.2 The Artificial Bee Colony Algorithm

The artificial bee colony (ABC) algorithm is a population-based algorithm, which is motivated by the intelligent foraging behavior of a honey bee swarm [49]. In the ABC algorithm, each position

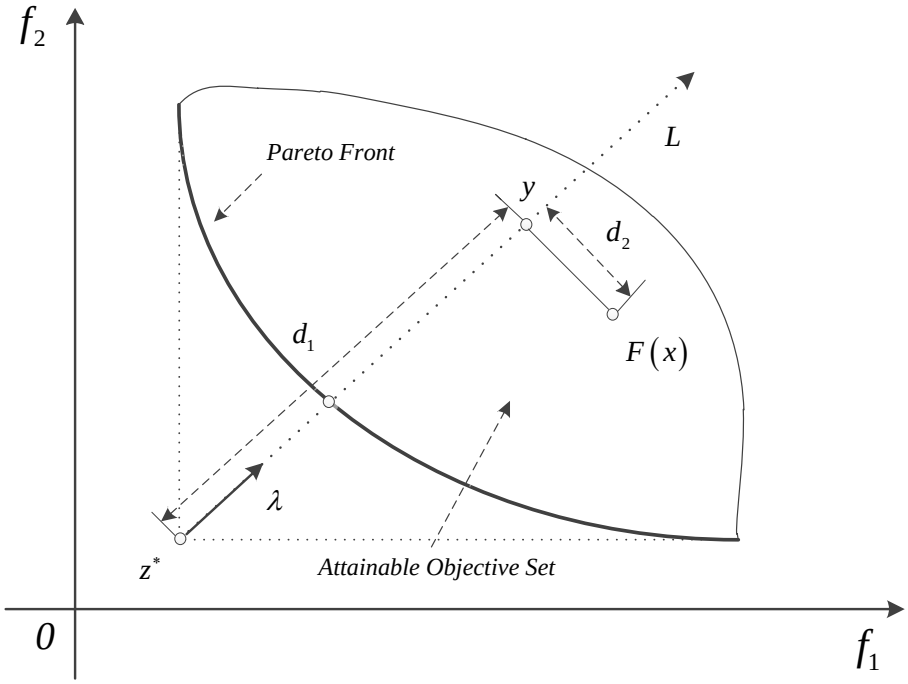


Figure 3.1: Illustration of penalty-based boundary intersection (PBI) approach

of a food source represents a potential solution of the optimization problem and the nectar amount of a food source corresponds to the fitness. The honey colony swarm contains three types of bees: employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to select good food sources, and scout bees searching for food randomly.

In the ABC algorithm, the number of employees and onlookers is equal to the number of food sources. The ABC algorithm first generates a randomly distributed initial population of N solutions in the swarm, where N is the swarm size. Then, the employed bees search the new solutions within the neighborhood in their memory. Let $X_i = \{x_{i,1}, x_{i,2}, \cdots, x_{i,n}\}$ represent the i-th solution in the swarm, where n is the dimension. Each employed bee X_i generates a new position V_i using the following formula:

$$V_{i,k} = X_{i,k} + \Phi_{i,k} \times (X_{i,k} - X_{j,k}) \tag{3.5}$$

where X_j is a randomly selected solution ($i \neq j$), k is a random dimension index from the set $\{1, 2, \cdots, n\}$, and $\Phi_{i,k}$ is a random number within the range $[-1, 1]$. After generating a new candidate solution V_i, a greedy selection between V_i and X_i is used. Comparing the fitness value between

V_i and X_i, the better one is adopted to update the population. Once the searching phase of the employed bees is completed, the employed bees share the food source information with the onlooker bees through waggle dances. An onlooker bee chooses a food source with a probability based on a roulette wheel selection mechanism. The probability P_i for the maximization problem is defined as follows:

$$P_i = \frac{fit_i}{\sum_j^N fit_j} \tag{3.6}$$

where fit_i is the fitness value of the i-th solution. The better solution often has a higher probability to be chosen to reproduce the new solution using Equation 3.5. If a position X_i cannot be improved through a predefined number of cycles, then it is replaced by the new solution X_i^{new} discovered by the scout bee using the following equation:

$$X_{i,k}^{new} = lb_i + rand(0,1) \times (ub_i - lb_i) \tag{3.7}$$

where $rand(0,1)$ is a random number in $[0,1]$. The upper and lower boundaries of the i-th dimension are lb_i and ub_i, respectively.

3.3 The Proposed MOEA/D-ABC

In this section we will present the details of the proposed MOEA/D-ABC. We combine the advantages of an artificial bee colony (ABC) algorithm and a multi-objective evolutionary algorithm based on decomposition (MOEA/D) [113], so we name the proposed algorithm MOEA/D-ABC. Compared with the original MOEA/D, an artificial bee colony (ABC) algorithm is adopted as a new reproduction operator to improve the convergence of MOEA/D. Second, a modified Tchebycheff approach instead of the original Tchebycheff approach is used to improve diversity.

3.3.1 General Framework

The general framework of the proposed MOEA/D-ABC is given in Algorithm 2. First, a set of uniformly distributed weight vectors $\Lambda = \left\{\lambda^1, \lambda^2, \cdots, \lambda^N\right\}$ is generated by the Das and Dennis's systematic approach [17] (Step 1 in Algorithm 2), $\lambda^1, \lambda^2, \cdots, \lambda^N$ are all m-dimensional weight vectors and m is the number of objectives. Then, a population of N solutions $P = \{x_1, x_2, \cdots, x_N\}$ is initialized randomly (Step 2 in Algorithm 2), after that the reference point $z^* = (z_1^*, z_2^*, \cdots, z_m^*)^T$ is initialized (Step 3 in Algorithm 2). According to the generated weight vectors, the neighborhood range T of subproblem i as $B(i) = \{i_1, \cdots, i_T\}$ can be obtained by computing the Euclidean

distance between all the weight vectors and finding the T closest weight vectors (Step 5 in Algorithm 2). Steps 7-17 are iterated until the termination criterion is met. At each iteration, for the solution x_i, the mating solutions x_k and x_l are chosen from the neighborhood $B(i)$. In MOEA/D-ABC, we use the ABC operator and polynomial mutation operator to reproduce the offspring y, which will be introduced in detail in Section 3.3.3. Then the new offspring is used to update the reference point and neighboring solutions. In addition, we use the modified Tchebycheff approach to determine the search direction for updating the neighboring solutions. Steps 9-11 in Algorithm 2 refers to Step 4 in Algorithm 1. Steps 12-13 in Algorithm 2 refers to Steps 5-6 in Algorithm 1. Steps 14-15 in Algorithm 2 refers to Step6 7-8 in Algorithm 1.

Algorithm 2: Framework of MOEA/D-ABC

```
1  Generate a set of weight vectors Λ ← {λ^1, λ^2, ··· , λ^N};
2  Initialize the population P ← {x_1, x_2, ··· , x_N};
3  Initialize the reference point z* ← (z_1*, z_2*, ··· , z_m*)^T;
4  for i = 1 : N do
5  |  B(i) ← {i_1, i_2, ··· , i_T}, where λ^{i_1}, λ^{i_2}, ··· , λ^{i_T} are the T closest weight vectors to λ^i ;
6  end
7  while the termination criterion is not satisfied do
8  |  for i = 1 : N do
9  |  |  E ← B(i) ;
10 |  |  Select an index k ∈ E based on roulette wheel selection ;
11 |  |  Randomly select an index l ∈ E and l ≠ k ;
12 |  |  ȳ ← ABCOperator(x_k, x_l) ;
13 |  |  y ← PolynomialMutationOperator(ȳ);
14 |  |  UpdateIdealPoint(y, z*);
15 |  |  UpdateNeighborhood(y, z*, Λ, B(i));
16 |  end
17 end
```

3.3.2 Modified Tchebycheff Approach

In MOEA/D-ABC, the modified Tchebycheff approach is defined as follows:

$$\min_{x\in\Omega} g^{mte}(x\,|\lambda, z^*) = \min_{x\in\Omega} \max_{1\le i\le m} \left\{ \frac{1}{\lambda_i} |f_i(x) - {z_i}^*| \right\} \tag{3.8}$$

where $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^T$ is a weight vector and $\sum_{i=1}^{m} \lambda_i = 1, \lambda_i \ge 0, i = 1, 2, \cdots, m$. $z^* = (z_1^*, z_2^*, \cdots, z_m^*)^T$ is the reference point. It is worth noting that the modified Tchebycheff approach

has two advantages [111] over the original one in MOEA/D [113]. First, the modified form can produce more uniformly distributed solutions with a set of uniformly spread weight vectors. Second, each weight vector corresponds to a unique solution on the Pareto front. The proof can be found in Theorem 3.3.1.

Theorem 3.3.1. Assume a straight line passing through reference point z^* with the direction vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ has a intersection with the Pareto front (PF), then the intersection point is the optimal solution to $\Gamma(x)$ (i.e., $\Gamma(x) = \max\limits_{1 \le i \le m} \left\{ \frac{1}{\lambda_i} |f_i(x) - {z_i}^*| \right\}$).

Proof. Let $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$ be the intersection point with the PF, then we can have the following equality

$$\frac{f_1(x) - z_1^*}{\lambda_1} = \frac{f_2(x) - z_2^*}{\lambda_2} = \cdots = \frac{f_m(x) - z_m^*}{\lambda_m} = C \tag{3.9}$$

where C is a constant. Suppose $F(x)$ is not the optimal solution to $\Gamma(x)$, then $\exists F(y)$ that satisfies $\Gamma(y) < \Gamma(x)$. According to Equation 3.9, $\Gamma(x) = C$. Then $\forall k \in \{1, 2, \cdots, m\}$, we have

$$\frac{f_k(y) - z_k^*}{\lambda_k} \le \Gamma(y) < C = \frac{f_k(x) - z_k^*}{\lambda_k} \tag{3.10}$$

Hence, $f_k(y) < f_k(x)$. This is in contradiction with the condition that $F(x)$ is the intersection point on the PF and the supposition is invalid. □

3.3.3 The ABC Operator

Inspired by the artificial bee colony (ABC) algorithm, we adopt the ABC operator to reproduce the offspring. For each solution x_i, one mating solution x_k is chosen based on the roulette wheel selection mechanism and another x_l $(l \neq k)$ is randomly selected from the neighborhood $B(i)$. To get the mating solution x_k, assuming there is a solution x_i and its associated weight vector λ^i. First, the fitness value of the solution x_i can be calculated using the following equation:

$$\Gamma(x_i) = \max_{1 \le j \le m} \left\{ \frac{1}{\lambda_j^i} |f_j(x_i) - {z_j}^*| \right\} \tag{3.11}$$

In this way we can obtain T fitness values $\Gamma(B(i))$ of the neighboring solutions with the same

weight vector λ^i. Then the fitness value of the solution x_i can be converted in the following way:

$$\Gamma^* (x_i) = \exp\left(\frac{-\Gamma (x_i)}{\sum \Gamma (B (i))/T}\right) \tag{3.12}$$

According to the converted T fitness values, the mating solution x_k can be determined using the roulette wheel selection mechanism. For each solution x_i, the new solution $\bar{y}$ is computed as follows:

$$\bar{y} = x_k + \Phi_i \times (x_k - x_l) \tag{3.13}$$

where Φ_i is a n-dimensional random vector within the range $[-1, 1]$. After using the ABC operator to obtain the new solution $\bar{y}$, we apply a polynomial mutation on $\bar{y}$ with probability p_m to produce a new offspring y. The polynomial mutation is defined as follows:

$$y_k = \begin{cases} \bar{y}_k + \sigma_k \times (b_k - a_k) & \text{with probability } p_m, \\ \bar{y}_k & \text{with probability } 1 - p_m. \end{cases} \tag{3.14}$$

with

$$\sigma_k = \begin{cases} (2 \times rand)^{\frac{1}{\eta+1}} - 1 & \text{if } rand < 0.5, \\ 1 - (2 - 2 \times rand)^{\frac{1}{\eta+1}} & \text{otherwise.} \end{cases}$$

where $rand$ is a uniformly distributed random number within $[0, 1]$. The lower and upper bounds of the k-th dimension are a_k and b_k. The distribution index η and the mutation rate p_m are two control parameters.

3.3.4 Adaptive Normalization

For disparately scaled objectives the original MOEA/D sometimes cannot provide satisfactory results. The normalization operators are by default incorporated into the MOEA/D framework. In recent research there are three typical normalization approaches proposed in MOEA/D [113], NSGA-III [22], and I-DBEA [2]. The normalization procedures in NSGA-III and I-DBEA are similar to some extent, as both aim at finding the extreme points to constitute a hyperplane. However, these two algorithms are more computationally expensive for solving the linear system of equations and sometimes result in abnormal normalization results [109]. Therefore, in this chapter we select a simple and efficient way to normalize the objectives.

For a solution x_i the objective value $f_j (x_i)$ $(j = 1, 2, \cdots, m)$ can be replaced with the normal-

ized objective value $\bar{f}_j(x_i)$ as follows:

$$\bar{f}_j(x_i) = \frac{f_j(x_i) - z_j^*}{z_j^{\max} - z_j^*} \tag{3.15}$$

where $z_j^{\max}$ is the maximum value of objective f_j in the current population.

3.3.5 Computational Complexity

For MOEA/D-ABC, the major computational costs are the iteration process in Algorithm 2. Steps 9-13 mainly need $O(mT)$ operations to calculate the modified Tchebycheff values for choosing the mating solutions based on the roulette wheel selection mechanism. Step 14 performs $O(m)$ comparisons to update the reference point. Step 15 requires $O(mT)$ computations to update the neighborhood. Thus, the overall computational complexity of MOEA/D-ABC is $O(mNT)$ in one generation. Considering the adaptive normalization operator incorporated into the MOEA/D-ABC for solving the scaled optimization problems, the computational complexity of MOEA/D-ABC will be $O\left(mN^2\right)$ in one generation since T is smaller than N.

3.4 Experimental Studies

In this section we compare the performance of the proposed algorithm with other state-of-the-art MOEAs for solving different MOPs.

3.4.1 Experiment Settings

The proposed MOEA/D-ABC is implemented in the PlatEMO framework [85]. For better comparison the other algorithms are also chosen from the PlatEMO. Two well-known ZDT [122] and DTLZ [25] test suites are used as test instances.

In order to evaluate the performance of the proposed algorithm, we have chosen the inverse generational distance (IGD) [93] as a performance metric which can reflect both convergence and diversity. Since the exact Pareto front of the test problems is known we can easily locate some uniformly targeted points in the optimal surface. As for the IGD metric a smaller value means that the obtained solutions have better quality. For each test instance 30 independent runs are performed and mean and standard deviation of the IGD values are recorded. For all algorithms we use the solutions from the final generation to compute the performance metrics.

In the experiment the performance of MOEA/D-ABC is compared with NSGA-II [24], MOEA/D

[113] and MOEA/D-DE [58]. The original MOEA/D study proposes two procedures MOEA/D-TCH using the Tchebycheff and MOEA/D-PBI using the penalty-based boundary intersection (PBI) approach. Table 3.1 presents some parameters for crossover and mutation operators used in the proposed algorithm (MOEA/D-ABC) and compared algorithms NSGA-II [24], MOEA/D-TCH [113] and MOEA/D-DE [58].

Table 3.1: Parameters for crossover and mutation

Parameters	MOEA/D-ABC	NSGA-II	MOEA/D-TCH	MOEA/D-DE
SBX probability (p_c)	-	1	1	-
Polynomial mutation probability (p_m)	$1/n$	$1/n$	$1/n$	$1/n$
Distribution index for crossover (η_c)	-	20	20	-
Distribution index for mutation (η_m)	20	20	20	20
DE operator control parameter (CR)	-	-	-	1
DE operator control parameter (F)	-	-	-	0.5

The neighborhood size T is set to 20 and the penalty parameter θ is set to 5 for MOEA/D-PBI. In MOEA/D-DE the probability δ of choosing the parent solution form the whole population is set to 0.9 and the maximum number of replaced solutions n_r is set to 2. As analyzed above MOEA/D-ABC has the obvious advantage of having less parameters.

3.4.2 Normalized Test Problems

Initially we use the ZDT problems and the DTLZ problems (DTLZ1, DTLZ2, DTLZ3, DTLZ4) to test the performance of the respectively used algorithms. The number of variables D is set according to the original papers. Since the test problems have similar range of values for each objective they are called "normalized test problems". For all 2-objective ($m = 2$) ZDT test problems the population size N in NSGA-II and other variants of MOEA/D is set to be 100 and the number of function evaluations (FEs) is set to 30000. The number of function evaluations (FEs) means the number of calculating objective functions. For all 3-objective ($m = 3$) DTLZ test problems N is set to 200 and FEs is set to 100000.

Figs. 3.2 and 3.3 show the obtained fronts with the median value of IGD performance metric of all algorithms for ZDT4 and DTLZ1. As shown from Fig. 3.2, we can observe that the proposed MOEA/D-ABC can determine the Pareto optimal solutions with better convergence and diversity. Compared with the other three algorithms MOEA/D-DE has the worst convergence for the ZDT4 problem with its many local optima. According to the analysis of convergence we find that the ABC operator improves the convergence of MOEA/D in comparison with the simulated binary crossover (SBX) operator [113]. In Fig. 3.3, NSGA-II can get the random non-dominated solutions

in the Pareto front. Both MOEA/D-TCH and MOEA/D-DE use the Tchebycheff approach as a decomposition method to obtain the similar Pareto front. MOEA/D-ABC performs much better than MOEA/D-TCH and MOEA/D-DE with regard to the diversity which illustrates that the modified Tchebycheff approach improves the diversity of MOEA/D compared with the original Tchebycheff approach. Table 3.2 shows that MOEA/D-ABC outperforms the other three algorithms with respect to the IGD performance metric. The best result in each row is highlighted.

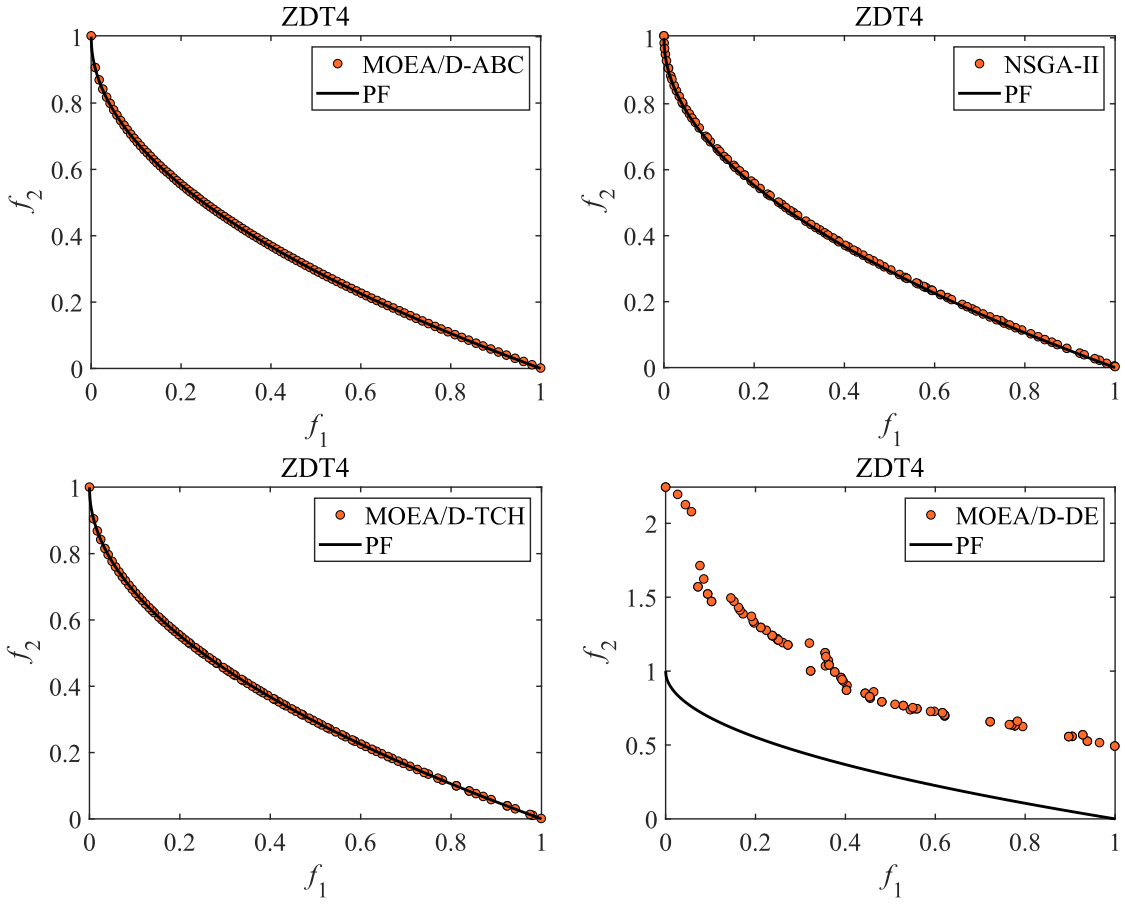


Figure 3.2: Final solutions obtained by MOEA/D-ABC, NSGA-II, MOEA/D-TCH, and MOEA/D-DE on ZDT4

3.4.3 Scaled Test Problems

To investigate the proposed algorithm's performance in the case of disparately scaled objectives we choose the SZDT1 and SZDT2 as two-objective scaled test instances and SDTLZ1 and SDTLZ2

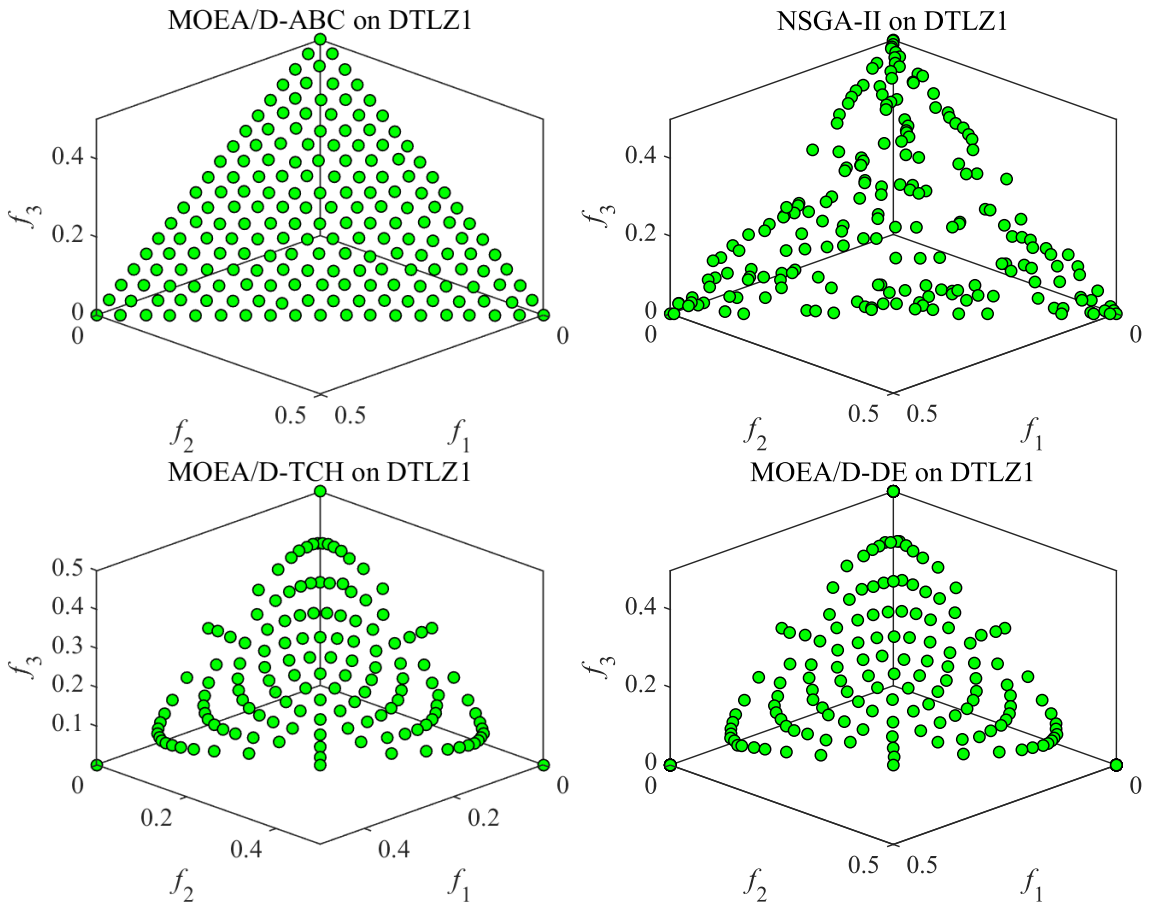


Figure 3.3: Final solutions obtained by MOEA/D-ABC, NSGA-II, MOEA/D-TCH, and MOEA/D-DE on DTLZ1

as three-objective scaled test instances. The SZDT and SDTLZ test suites are described in Section 2.3.

To handle the differently scaled test problems, we incorporate the adaptive normalization operator presented in Section 3.3.4 into the proposed MOEA/D-ABC. The original MOEA/D-TCH with and without normalization procedure is also used to compare the performance. For clarity, we denote the MOEA/D-ABC using the normalization procedure as MOEA/D-ABC-N, MOEA/D-TCH with normalization procedure as MOEA/D-TCH-N, respectively. Fig. 3.4 shows the distribution of obtained solutions for MOEA/D-ABC-N, MOEA/D-TCH-N and MOEA/D-TCH on SZDT2 and SDTLZ1. It is clear that the normalization operator can greatly improve the performance for handling the scaled problems. Both MOEA/D-ABC-N and MOEA/D-TCH-N can obtain better distributed solutions

Table 3.2: IGD values for MOEA/D-ABC, NSGA-II, MOEA/D-TCH, and MOEA/D-DE on ZDT and DTLZ test instances

Problem	N	m	D	FEs	MOEA/D-ABC	NSGA-II	MOEA/D-TCH	MOEA/D-DE
ZDT1	100	2	30	30000	4.0371e-3 (6.09e-5)	4.6043e-3 (1.84e-4)	5.8106e-3 (5.88e-3)	1.1626e-2 (5.42e-3)
ZDT2	100	2	30	30000	3.8379e-3 (2.29e-5)	4.7864e-3 (1.99e-4)	5.3845e-3 (4.66e-3)	9.4609e-3 (3.62e-3)
ZDT3	100	2	30	30000	1.0928e-2 (3.85e-2)	4.1278e-2 (5.03e-2)	1.9680e-2 (2.06e-2)	2.5511e-2 (1.52e-2)
ZDT4	100	2	10	30000	4.5511e-3 (9.93e-4)	5.4563e-3 (9.52e-4)	7.3588e-3 (4.00e-3)	1.8529e-1 (1.62e-1)
ZDT6	100	2	10	30000	3.1078e-3 (1.07e-5)	3.7673e-3 (1.14e-4)	3.1968e-3 (4.79e-5)	3.1125e-3 (1.63e-5)
DTLZ1	200	3	7	100000	1.4208e-2 (6.95e-4)	1.9097e-2 (9.01e-4)	1.9937e-2 (2.02e-5)	1.9716e-2 (5.19e-5)
DTLZ2	200	3	12	100000	3.7745e-2 (3.05e-4)	4.8807e-2 (1.49e-3)	4.9259e-2 (7.96e-5)	4.8923e-2 (2.25e-4)
DTLZ3	200	3	12	100000	4.3475e-2 (2.96e-3)	4.8337e-2 (1.22e-3)	4.8881e-2 (2.52e-4)	1.3899e-1 (4.83e-1)
DTLZ4	200	3	12	100000	4.1621e-2 (1.35e-3)	4.8543e-2 (1.33e-3)	2.7569e-1 (2.73e-1)	7.3774e-2 (6.23e-2)

than MOEA/D-TCH with regard to SZDT2. MOEA/D-TCH is not able to handle SDTLZ1 without normalization. It is interesting to observe that MOEA/D-ABC-N is superior to MOEA/D-TCH-N with respect to diversity for solving SDTLZ1. The IGD performance metric values of concerning algorithms are shown in Table 3.3 which also verifies the efficiency and reliability of MOEA/D-ABC with normalization for solving disparately scaled objective problems.

Table 3.3: IGD values for MOEA/D-ABC-N, MOEA/D-TCH-N, and MOEA/D-TCH on SZDT1-2 and SDTLZ1-2 test instances

Problem	N	m	D	FEs	MOEA/D-ABC-N	MOEA/D-TCH-N	MOEA/D-TCH
SZDT1	200	2	30	100000	1.1069e-2 (3.68e-5)	1.1043e-2 (7.72e-6)	5.0087e-2(6.85e-5)
SZDT2	200	2	30	100000	1.1358e-2 (1.14e-5)	6.8675e-1 (1.42e+0)	4.0293e-2(8.80e-6)
SDTLZ1	200	3	7	100000	1.2620e-1 (2.06e-2)	5.2850e-1 (5.72e-3)	9.1805e+0(8.39e-3)
SDTLZ2	200	3	12	100000	3.1699e-1 (2.07e-2)	1.0600e+0 (2.27e-3)	1.5530e+1(9.49e-3)

3.4.4 MOEA/D-ABC VS MOEA/D-PBI

The proposed MOEA/D-ABC adopts the modified Tchebycheff approach while MOEA/D-PBI uses the penalty-based boundary intersection (PBI) approach. In the original MOEA/D study [113], MOEA/D-PBI can obtain much better distribution of solutions than NSGA-II and MOEA/D-TCH on DTLZ1 and DTLZ2 instances when setting the penalty parameter to 5. According to the experiments on normalized test problems, MOEA/D-ABC can also get good results on three-objective instances. To further compare the performance of MOEA/D-ABC and MOEA/D-PBI, we choose DTLZ5 and

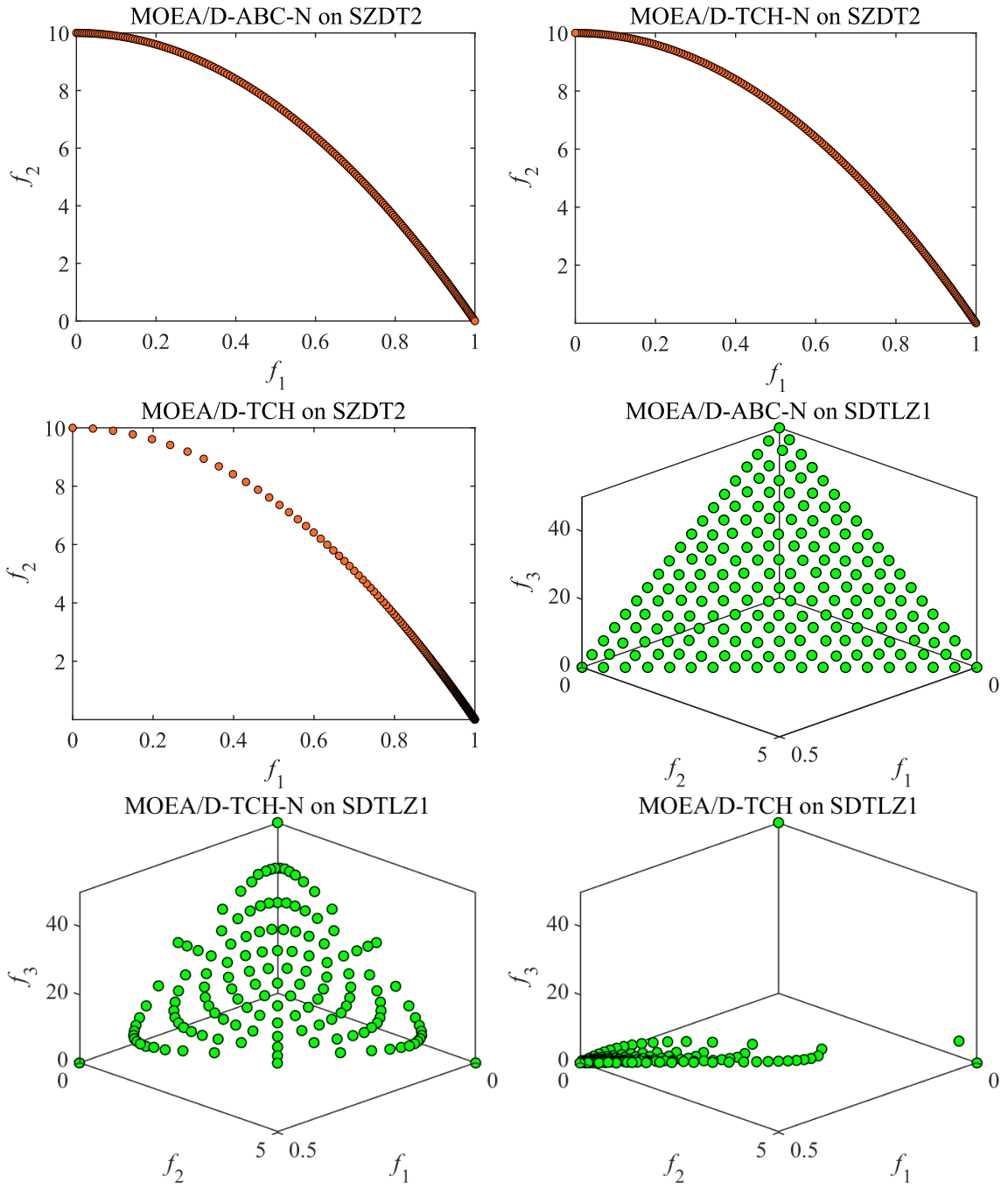


Figure 3.4: Final solutions obtained by MOEA/D-ABC-N, MOEA/D-TCH-N, and MOEA/D-TCH on SZDT2 and SDTLZ1

DTLZ6 as the test instances.

Fig. 3.5 shows the obtained Pareto fronts with MOEA/D-ABC and MOEA/D-PBI on these two three-objective test instances. It is clear that MOEA/D-PBI is unable to find the convergent front with the penalty factor 5. However, MOEA/D-ABC can determine the front approaching the true Pareto front. Table 3.4 shows the IGD metric of the obtained solutions with MOEA/D-ABC and MOEA/D-PBI for DTLZ5 and DTLZ6 instances. Based on the above result analysis we see that the use of a penalty parameter cannot always obtain good results. MOEA/D-PBI requires an appropriate setting of the penalty parameter for different problems. MOEA/D-ABC is a more stable and efficient algorithm to solve different optimization problems.

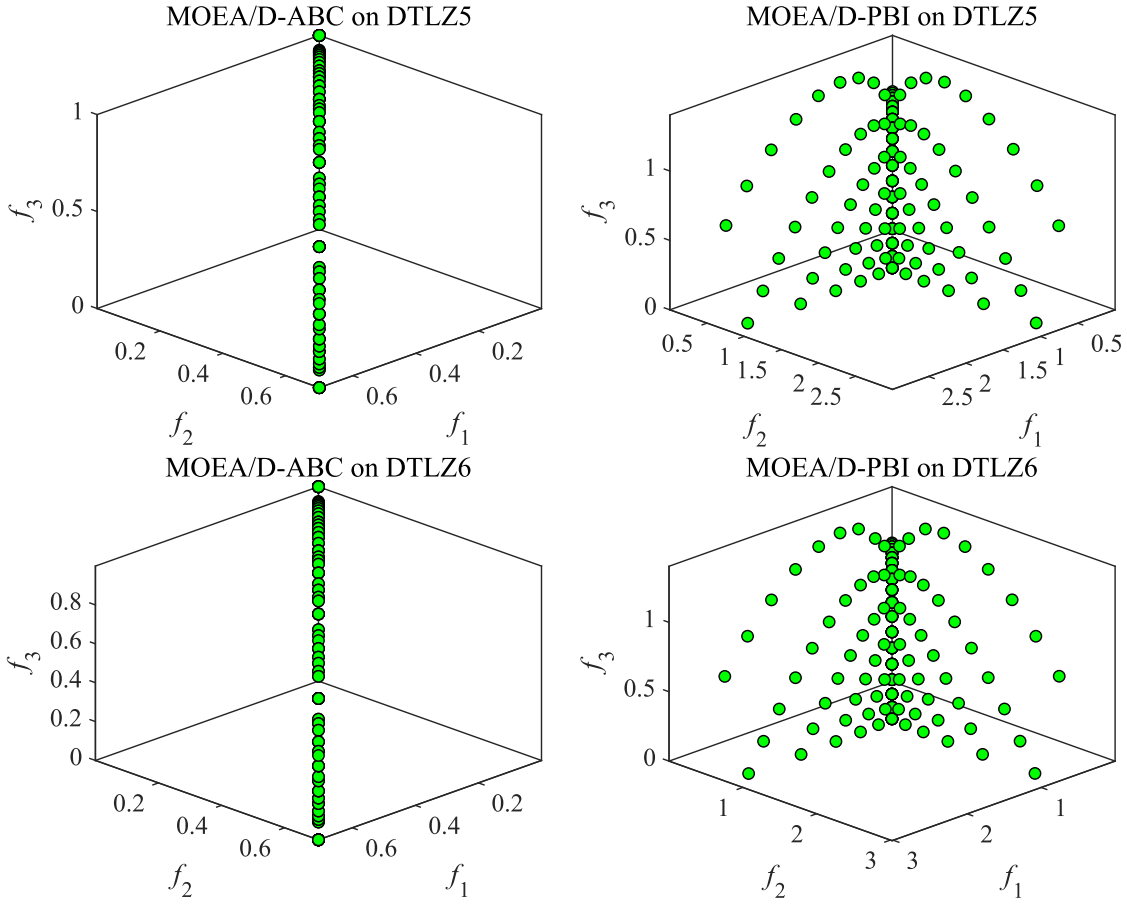


Figure 3.5: Final solutions obtained by MOEA/D-ABC and MOEA/D-PBI on DTLZ5 and DTLZ6

Table 3.4: IGD values for MOEA/D-ABC and MOEA/D-PBI on DTLZ5 and DTLZ6 test instances

Problem	N	m	D	FEs	MOEA/D-ABC	MOEA/D-PBI
DTLZ5	200	3	12	100000	1.1250e-2 (2.06e-5)	2.2605e-2 (1.95e-5)
DTLZ6	200	3	12	100000	1.1319e-2 (7.43e-6)	2.2632e-2 (4.78e-6)

3.5 Summary

In this chapter we have developed a multi-objective artificial bee colony algorithm based on decomposition (MOEA/D-ABC) for solving MOPs. The proposed MOEA/D-ABC approach adopts a novel ABC operator as new reproduction operator and a modified Tchebycheff approach as new decomposition method, respectively. The above two operators are used to improve the convergence and diversity of the algorithm. Furthermore, the adaptive normalization operator is incorporated into the proposed MOEA/D-ABC for handling differently scaled problems.

In the experiment two well-known test suites and some modified scaled test instances have been applied to test the performance of proposed MOEA/D-ABC and compare them with other state-of-the-art MOEAs. The test problems involve fronts that have convex, concave, disjointed, non-uniformly distributed, differently scaled, and many local fronts where an optimization algorithm can get stuck in. The proposed MOEA/D-ABC can obtain a well-converging and well-diversified set of solutions repeatedly for all problems, which shows its obvious advantage over other compared MOEAs. Moreover, there is another advantage of MOEA/D-ABC which is that it does not require any additional parameters with respect to the reproduction operator compared with other MOEA/Ds.

Chapter 4

A Many-objective Decomposition-based Algorithm

This chapter illustrates a decomposition-based evolutionary algorithm with adaptive weight vectors (called DBEA-AWV) for solving the normalized and scaled many-objective optimization problems (MaOPs).

Multi-objective optimization problems (MOPs) involve more than one conflicting objective to be optimized. Especially, MOPs with more than three objectives are known as many-objective optimization problems (MaOPs). With the growth of the number of objectives, more and more solutions are non-dominated by each other. Multi-objective evolutionary algorithms (MOEAs) face more challenges for solving MaOPs since it is difficult to select non-dominated solutions.

Although different MOEAs using decomposition approaches in the MOEA/D framework have been verified on different normalized MOPs and MaOPs, the literature [22, 57] has demonstrated the unstable performance of MOEA/D variants when dealing with scaled problems. In other words, even adopting different normalization approaches into MOEA/D framework, MOEA/D versions can't always get good results with regard to scaled problems. To address the issue, we propose a decomposition-based evolutionary algorithm with adaptive weight vectors (DBEA-AWV) for both the normalized and scaled MOPs and MaOPs. The main contributions of this chapter are summarized as follows:

- An adaptive weight vectors adjusting method is proposed for problems with disparately scaled objectives instead of using normalization approaches.
- Based on the adaptive weight vectors, we analyze the characteristics of the existing six popular decomposition approaches and find the best one. Further, one novel replacement strategy is

adopted to keep the balance between convergence and diversity for solving MOPs and MaOPs.

4.1 Compared Decomposition Approaches

In the original MOEA/D, three decomposition approaches are reported: the weighted sum, Tchebycheff and penalty-based boundary intersection (PBI). The weighted sum approach has some shortcomings to deal with concave Pareto fronts [113].

PBI approach has gained particular research interest because of the ability to control convergence and diversity. To improve the performance of the original PBI with a constant penalty value, Yang et al. [105] proposed two new penalty schemes, i.e., adaptive penalty scheme (APS) and subproblem-based penalty scheme (SPS). APS is defined as follows:

$$\theta = \theta_{\min} + (\theta_{\max} - \theta_{\min}) \frac{t}{t_{\max}} \tag{4.1}$$

where t is the iteration number, $t_{\max}$ is maximum number of iterations. $\theta_{\max}$ and $\theta_{\min}$ are the upper and lower bounds of θ, respectively. $\theta_{\min} = 1$ and $\theta_{\max} = 10$ are recommended in the original paper.

The subproblem-based penalty scheme (SPS) is described as follows:

$$\theta_j = e^{\alpha\beta_j} \tag{4.2}$$

$$\beta_j = \max_{1\leq i\leq m} \lambda_i^j - \min_{1\leq i\leq m} \lambda_i^j \tag{4.3}$$

where θ_j represents the penalty value for a weight vector λ^j. β_j is the difference between the maximum and minimum value of λ^j. α is a control parameter, and $\alpha = 4$ is suggested.

Moreover, an adaptive PBI selection is developed in [39], an angle-based dynamic penalty factor adaptation strategy is determined as follows:

$$\theta_k = K \cdot m \cdot \left(\alpha_k^{ind} + \alpha_k^{neighbor}\right) \tag{4.4}$$

where α_k^{ind} is the angle between the current solution x_k and λ^k, $\alpha_k^{neighbor}$ is the angle between the weight vector λ^k and the closest neighboring one. m is the number of objectives. K is a pre-defined scaling parameter, and $K = 0.06$ is suggested.

4.2 The Proposed DBEA-AWV

This section presents the details of the proposed DBEA-AWV.

4.2.1 General Framework

The general framework of the proposed DBEA-AWV is described as Algorithm 3. First, Das and Dennis's systematic approach [17] is used to generate a set of uniform weight vectors. The population $P \leftarrow \{x_1, x_2, \cdots, x_N\}$ is randomly generated, then the reference point is initialized. The neighborhood set of each weight vector can be derived based on the Euclidean distance. The widely used simulated binary crossover (SBX) and polynomial mutation [113] are applied to produce the offspring. The other two main components, i.e., weight vector adaptation and replacement strategy of updating population will be introduced in detail in the following sections. The flowchart of the proposed DBEA-AWV is shown in Fig. 4.1.

Algorithm 3: Framework of DBEA-AWV

Input: A set of uniform weight vectors $\Lambda_0 \leftarrow \left\{\lambda_0^1, \lambda_0^2, \cdots, \lambda_0^N\right\}$, the maximum number of generations $t_{\max}$

Output: The final population P

```
Initialize the population P ← {x_1, x_2, ··· , x_N};
Initialize the reference point z* ← (z*_1, z*_2, ··· , z*_m)^T;
Set Λ = Λ_0;
for i = 1 : N do
    B(i) ← {i_1, i_2, ··· , i_T}, where λ^{i_1}, λ^{i_2}, ··· , λ^{i_T} are T closest weight vectors to λ^i;
end
while t < t_max do
    for i = 1 : N do
        y = Offspring_Creation(P_t, λ^i, B(i));
        z* = Update_Ideal_Point(y, z*);
        P_{t+1} = Update_Population(y, z*, Λ_t, P_t);
    end
    Λ_{t+1} = Weight_Vector_Adaption(t, P_{t+1}, Λ_t, Λ_0) ;
    t = t + 1;
end
```

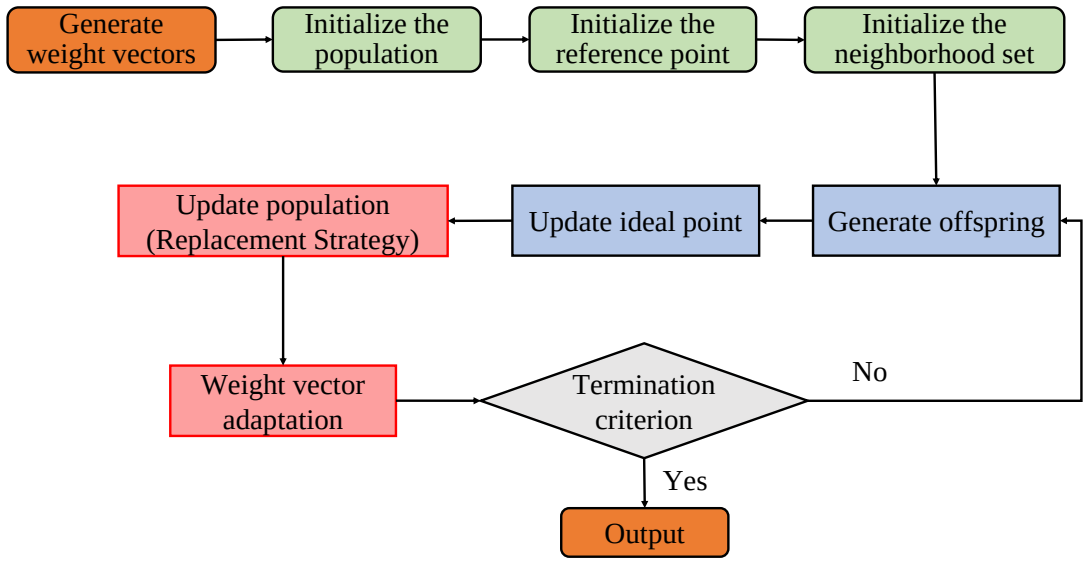


Figure 4.1: The flowchart of DBEA-AWV

4.2.2 Adaptive Weight Vectors

Given a set of uniform weight vectors in a hyperplane, the MOEA/D variants can often produce uniformly distributed solutions with regard to normalized problems. For scaled problems, it is a challenge to use fixed weight vectors or some normalization approaches [22,57]. We adopt a weight vector adaptation method to solve both the normalized and scaled problems. The adaptive weight vectors are adjusted by the ranges of objective values as follows:

$$\lambda^i_{t+1,k} = \frac{\lambda^i_{0,k} \times \left(f^{\max}_{t+1,k} - f^{\min}_{t+1,k}\right)}{\sum_{j=1}^{m} \times \left(\lambda^i_{0,j} \times \left(f^{\max}_{t+1,j} - f^{\min}_{t+1,j}\right)\right)} \tag{4.5}$$

where $i = 1, 2, \cdots, N$ and $k = 1, 2, \cdots, m$, $\lambda^i_{t+1,k}$ denotes k-th value of i-th adaptive weight vector for the next generation $t+1$. $\lambda^i_{0,k}$ denotes the k-th value of i-th weight vector in Λ_0. $f^{\max}_{t+1,k}$ and $f^{\min}_{t+1,k}$ represent the maximum and minimum values of the population in the generation $t+1$. The denominator in Equation 4.5 makes sure that $\sum_{k=1}^{m} \lambda^i_{t+1,k} = 1$ is suitable for different decomposition approaches in MOEA/D framework, which is different from the strategy in RVEA [11]. Based on the weight vector adaption strategy, the proposed DBEA-AWV is able to deal with

disparately scaled problems.

The literature [36] suggested that weight vector adaptation should be periodically executed to ensure convergence in the search process. The parameter f_r is used to control the frequency, the smaller f_r is, the higher frequency of weight vector adaptation will be employed. The weight vector adaptation method is described as Algorithm 4. Fig. 4.2 presents the adaptive weight vectors for Pareto fronts with different scales.

Algorithm 4: Weight vector adaptation

Input: Weight vector set Λ_0 and Λ_t, generation index t, population P_{t+1}
Output: Weight vector set Λ_{t+1} and neighborhood set B

if $\left(\frac{t}{t_{\max}} \mod f_r\right) == 0$ **then**
  Calculate the maximum and minimum objective values of $f_{t+1,k}^{\max}$ and $f_{t+1,k}^{\min}$ of P_{t+1}, respectively;
  for $i \leftarrow 1$ *to* N **do**
    for $k \leftarrow 1$ *to* m **do**
      $\lambda_{t+1,k}^{i} = \frac{\lambda_{0,k}^{i} \times \left(f_{t+1,k}^{\max} - f_{t+1,k}^{\min}\right)}{\sum_{j=1}^{m} \times \left(\lambda_{0,j}^{i} \times \left(f_{t+1,j}^{\max} - f_{t+1,j}^{\min}\right)\right)}$;
    end
  end
  for $i \leftarrow 1$ *to* N **do**
    Calculate the neighborhood set $B\left(i\right) \leftarrow \{i_1, i_2, \cdots, i_T\}$;
  end
else
  $\Lambda_{t+1} = \Lambda_t$;
end

It has been proven that the modified Tchebycheff approach can produce more uniformly distributed solutions against the original one [75]. Furthermore, we compare six different decomposition approaches (introduced in Section 4.1) based on the adaptive weight vectors to solve the SDTLZ1 problem having objectives with different scales [22], and the results are shown in Fig. 4.3.

From Fig. 4.3, we can see that except for the modified Tchebycheff approach, the proposed algorithm with other decomposition methods cannot find the true Pareto fronts of SDTLZ1. However, it is noted that the PBI approach based on adaptive weight vectors might be suitable for some small scaled problems, i.e., WFG test problems [44], as illustrated in Fig. 4.4. Based on the analysis, we can learn that the Tchebycheff approach suffers the diversity loss employing the adaptive weight vectors. PBI methods seem to be suitable for solving some small scaled problems. The literature [39, 57] could also demonstrate it to some extent. On the other hand, the penalty parameter

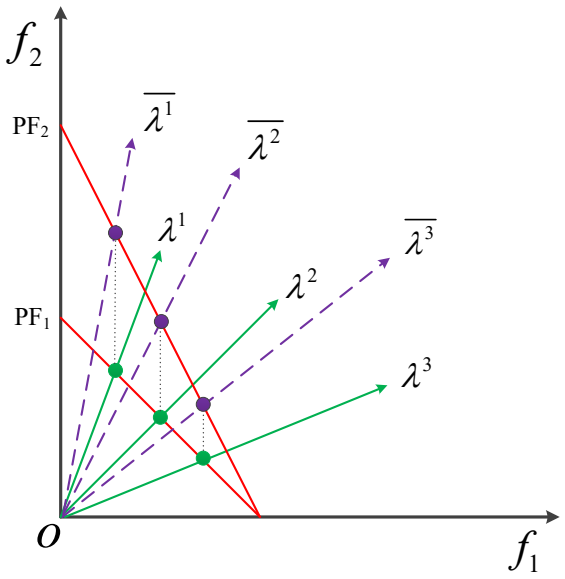


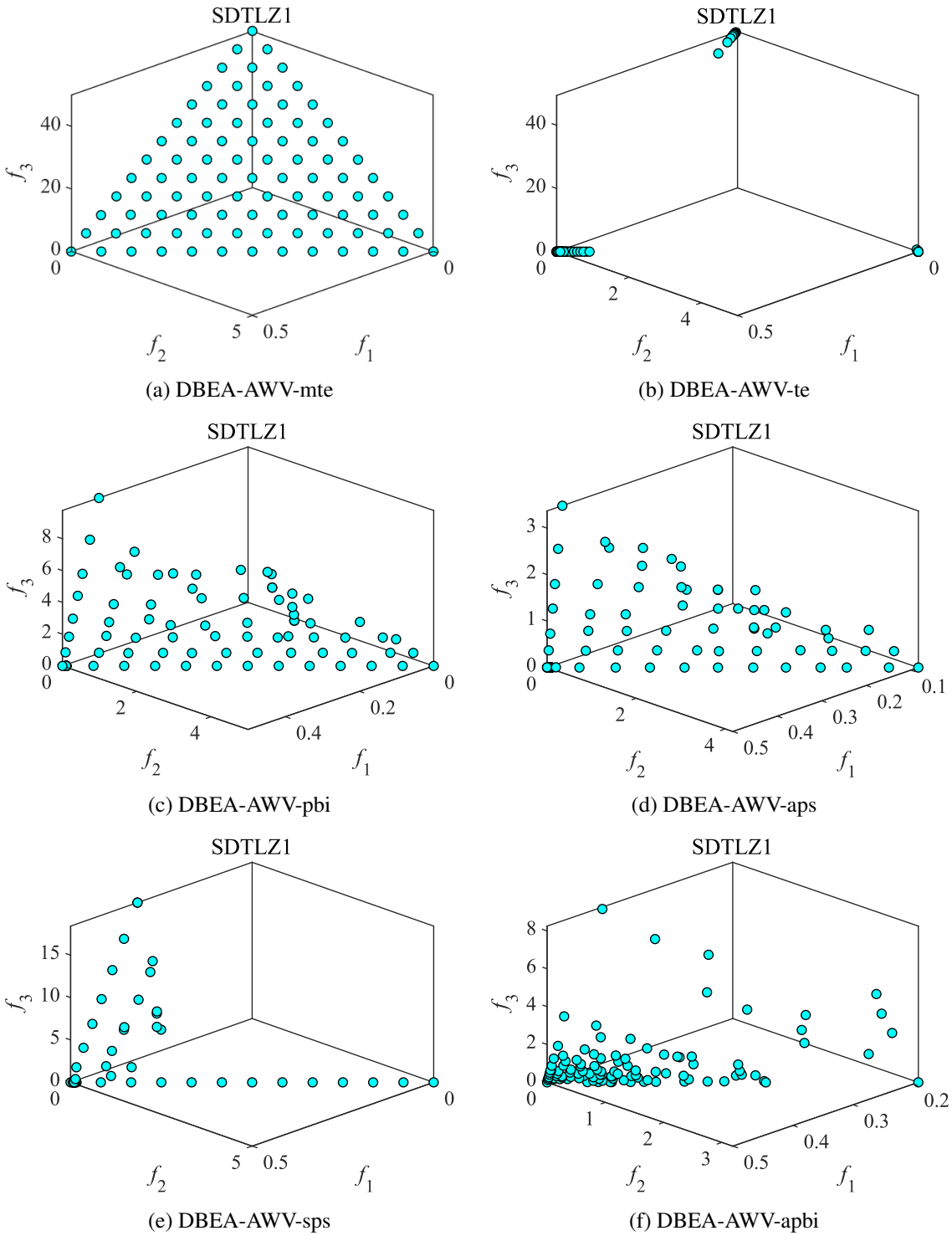
Figure 4.2: Adaptive weight vectors for Pareto fronts with different scales

is always difficult to set for different scaled problems. Therefore we select the modified Tchebycheff approach combined with adaptive weight vectors as our algorithm, denoted as DBEA-AWV to replace DBEA-AWV-mte for convenience.

4.2.3 Replacement Strategy

Compared with MOPs, in MaOPs it should be paid more attention to balancing the diversity and convergence based on a set of uniform weight vectors. Instead of updating the whole neighboring solutions, we choose one novel replacement strategy for MOPs and MaOPs.

With respect to MOPs and MaOPs, we calculate the acute angles between the offspring and all the solutions in the current population when an offspring is produced. A cluster of K front solutions is marked based on the sorted increasing acute angles. Then the offspring is compared with these K solutions one by one using the modified Tchebycheff value, and the process is terminated until one of K solutions is replaced with the offspring. In this way, the balance factor K is vital for controlling the convergence and diversity and the influence of K will be discussed in Section 4.3.4. Fig. 4.5 presents the replacement strategy with a cluster of K solutions.



(a) DBEA-AWV-mte (b) DBEA-AWV-te

(c) DBEA-AWV-pbi (d) DBEA-AWV-aps

(e) DBEA-AWV-sps (f) DBEA-AWV-apbi

Figure 4.3: The obtained Pareto fronts of six decomposition approaches on SDTLZ1

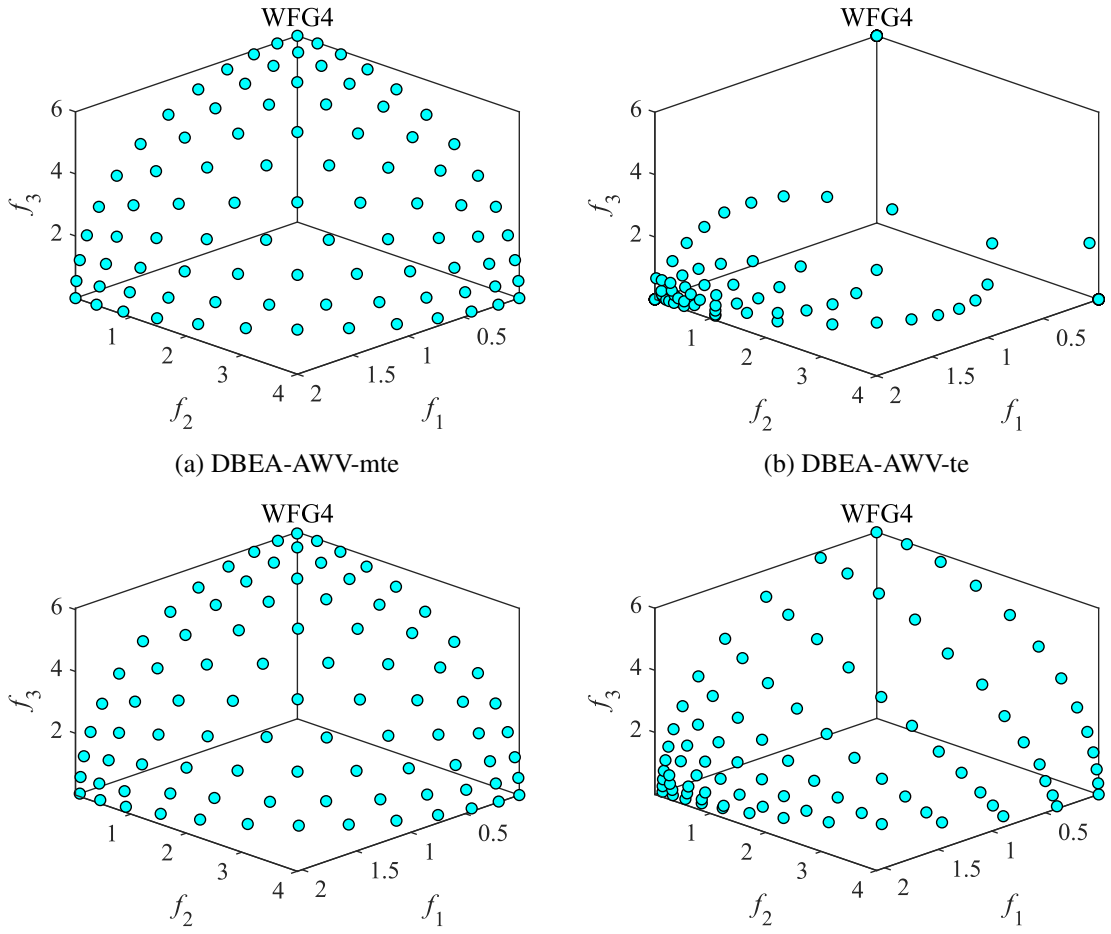


Figure 4.4: The obtained Pareto fronts of three decomposition approaches on WFG4

4.2.4 Computational Complexity

In the proposed DBEA-AWV, the major computational costs are the iteration process in Algorithm 3. Step 9 randomly chooses two solutions from the neighborhood set for genetic operators. Step 10 requires $O(m)$ comparisons to update the reference point. Step 11 performs $O(mN)$ operations to update the population for MOPs and MaOPs at the worst case. For the weight vector adaptation strategy in Step 13, it needs $O(mN)$ and $O\left(N^2\right)$ operations to update the weight vectors and the neighborhood set. Since it has N passes from Step 8 to Step 12, the overall computational complexity becomes $O\left(mN^2\right)$ for one generation of DBEA-AWV.

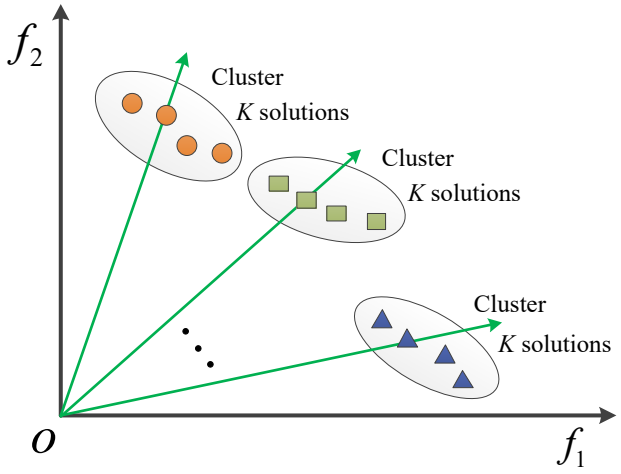


Figure 4.5: The replacement strategy with a cluster of K solutions

4.3 Experimental Studies

In this section empirical experiments are conducted on different MOPs and MaOPs to compare DBEA-AWV with other state-of-the-art algorithms. We then analyze the influence of parameters in DBEA-AWV.

4.3.1 Experimental Design

The experimental design includes four parts.

1) Test problems. The test problems are chosen from the widely used benchmark test suites ZDT [122] (including scaled ZDT, i.e. SZDT), DTLZ [25] (including scaled DTLZ, i.e. SDTLZ).

2) Performance metrics. The inverted generational distance (IGD) [125] and hypervolume (HV) [125] are adopted to evaluate the performance of the tested algorithms. For scaled problems, the objective values are normalized by the ideal and nadir points of the true PF. The smaller IGD and larger HV mean better.

3) Comparing algorithms. To assess the performance of the proposed DBEA-AWV, overall ten different state-of-the-art algorithms are chosen to evaluate. Five MOEAs are considered for MOPs. NSGA-II and IBEA are the representative Pareto dominance and indicator-based MOEAs, respectively. Other three MOEA/D (using Tchebycheff) [113], MOEA/D-AWA [75] and MOEA/D-STM [61] are the popular MOEA/D variants. Five many-objective optimization algorithms are selected for MaOPs. NSGA-III and RVEA are two widely used algorithms with good perfor-

mance. Other three MOEA/DD [60], MOEA/D-DU [111] and MOEA/D-PaS [95] are the typical decomposition-based algorithms for many-objective optimization.

4) Parameter settings. The population size of decomposition-based algorithms is controlled by a parameter H ($N = C_{H+m-1}^{m-1}$). To obtain the uniform weight vectors, a two-layered weight vectors method [22] is used. Table 4.1 lists the population size used for different number of objectives of proposed DBEA-AWV. For comparison, other tested algorithms adopt the same population size.

Table 4.1: The population size

No. of objectives (m)	Parameter(H_1, H_2)	Population size (N)
2	99, 0	100
3	13, 0	105
5	5, 0	210
8	3, 2	156
10	3, 2	275

The neighborhood size of decomposition-based algorithms is set to $0.1N$. The crossover probability and distribution index of SBX are set to $p_c = 1$ and $\eta_c = 20$, respectively. For polynomial mutation, the mutation probability and distribution index are set to $p_m = 1/n$ and $\eta_m = 20$, where n is the number of decision variables. For other parameters for specific algorithms, we use the same settings as the original references.

In DBEA-AWV, the frequency control parameter is set to $f_r = 0.2$, and the factor in replacement strategy is set to $K = 5$. More details of parameter analysis will be discussed in Section 4.3.4.

The termination of each run is the maximal number of generations, which is set to 1000 for all test problems. Each algorithm is run 30 times independently on each test instance, and the average and standard deviation of metric values are recorded. The Wilcoxon rank sum test at a 5% significance level is used to compare the experimental results, where the symbol '$+$', '$-$' and '$\approx$' denotes that the result of another algorithm is significantly better, significantly worse and similar to that obtained by DBEA-AWV, respectively.

4.3.2 Comparative Results on MOPs

Table 4.2 presents the IGD metric values obtained by NSGA-II, IBEA, MOEA/D, MOEA/D-AWA, MOEA/D-STM, and DBEA-AWV on a variety of MOPs, including ZDT1-ZDT6, SZDT1-SZDT6, DTLZ1-DTLZ4 and SDTLZ1-SDTLZ4. The best result in each row is highlighted. The proposed DBEA-AWV could achieve the best performance on 15 of 18 instances of all the test problems, while compared algorithms could get one or two best results at most. According to

the Wilcoxon rank sum test, the proposed DBEA-AWV has great advantages. It is still noted that MOEA/D-AWA shows some features suitable for the non-uniform distributed solutions especially for disconnected Pareto fronts. Fig. 4.6 shows the IGD values obtained by different tested algorithms. The compared algorithms can get good results for ZDT4 and SZDT4. The proposed DBEA-AWV can solve the 3-objective SDTLZ problem well compared with other algorithms.

Table 4.2: The IGD values obtained by tested algorithms

Problem	m	NSGA-II	IBEA	MOEA/D	MOEA/D-AWA	MOEA/D-STM	DBEA-AWV
ZDT1	2	4.6126e-3 (1.69e-4) −	4.5333e-3 (1.44e-4) −	3.8878e-3 (6.84e-8) ≈	3.9579e-3 (3.64e-5) −	3.9352e-3 (2.16e-5) −	3.8876e-3 (6.06e-7)
ZDT2	2	4.7196e-3 (1.31e-4) −	9.5288e-3 (1.11e-3) −	3.8070e-3 (8.79e-9) +	3.8303e-3 (1.70e-5) −	3.8367e-3 (1.18e-5) −	3.8070e-3 (1.23e-8)
ZDT3	2	1.1804e-2 (1.33e-2) −	4.9370e-2 (3.98e-2) −	1.1079e-2 (6.36e-6) −	5.0208e-3 (1.34e-4) +	1.1012e-2 (2.61e-5) −	7.0573e-3 (1.55e-2)
ZDT4	2	4.4721e-3 (1.48e-4) −	1.9465e-2 (2.60e-3) −	3.8975e-3 (1.27e-5) ≈	3.9465e-3 (5.25e-5) −	3.9749e-3 (1.21e-4) −	3.8907e-3 (1.37e-5)
ZDT6	2	3.8764e-3 (1.00e-4) −	5.3567e-3 (1.43e-4) −	3.2597e-3 (3.78e-7) −	3.2617e-3 (1.44e-5) −	3.2594e-3 (8.86e-8) −	3.1575e-3 (2.27e-7)
SZDT1	2	4.5832e-3 (1.49e-4) −	4.4852e-3 (1.81e-4) −	1.2390e-2 (2.16e-6) −	4.3416e-3 (1.21e-4) −	1.2406e-2 (2.13e-5) −	3.8875e-3 (7.56e-7)
SZDT2	2	4.7551e-3 (2.19e-4) −	9.1512e-3 (9.84e-4) −	1.5905e-2 (2.39e-7) −	6.7764e-3 (9.80e-4) −	1.6000e-2 (5.40e-5) −	3.8070e-3 (1.27e-8)
SZDT3	2	8.5945e-3 (9.99e-3) +	5.0946e-2 (3.93e-2) −	5.4132e-2 (7.66e-5) −	7.1083e-3 (1.38e-3) ≈	5.3839e-2 (1.83e-4) −	1.0178e-2 (9.86e-3)
SZDT4	2	4.6028e-3 (1.51e-4) −	1.8516e-2 (2.32e-3) −	1.2400e-2 (2.38e-5) −	4.3634e-3 (1.21e-4) −	1.2422e-2 (2.22e-5) −	3.9264e-3 (3.12e-5)
SZDT6	2	4.0015e-3 (2.01e-4) −	5.3703e-3 (9.87e-5) −	1.5110e-2 (1.92e-6) −	3.6376e-3 (8.29e-5) −	1.5109e-2 (2.13e-6) −	3.1672e-3 (2.22e-5)
$+/-/\approx$		1/9/0	0/10/0	1/7/2	1/8/1	0/10/0	
DTLZ1	3	5.3162e-2 (2.73e-3) −	3.1082e-1 (5.65e-2) −	5.6991e-2 (2.11e-5) −	4.0135e-2 (5.80e-4) −	3.8079e-2 (5.46e-5) −	3.7967e-2 (2.47e-5)
DTLZ2	3	6.7512e-2 (2.40e-3) −	7.9687e-2 (2.17e-3) −	6.9623e-2 (5.18e-5) −	5.0666e-2 (2.13e-4) −	5.1043e-2 (2.03e-4) −	5.0318e-2 (1.09e-5)
DTLZ3	3	6.7180e-2 (2.59e-3) −	4.7455e-1 (7.88e-3) −	6.9437e-2 (2.02e-4) −	5.1034e-2 (3.45e-4) −	5.3434e-2 (1.04e-3) −	5.0627e-2 (1.67e-4)
DTLZ4	3	6.6333e-2 (2.01e-3) −	7.7829e-2 (2.01e-3) −	3.4147e-1 (3.43e-1) −	1.1616e-1 (1.69e-1) −	7.6677e-2 (6.17e-2) −	5.0327e-2 (2.01e-5)
SDTLZ1	3	5.2909e-2 (2.25e-3) −	3.0006e-1 (3.67e-2) −	2.9216e-1 (2.80e-4) −	1.4347e-1 (5.21e-2) −	3.3976e-1 (8.95e-2) −	3.7971e-2 (2.02e-5)
SDTLZ2	3	6.7847e-2 (2.88e-3) −	7.8525e-2 (2.31e-3) −	3.4039e-1 (1.06e-4) −	1.2649e-1 (1.20e-2) −	2.7225e-1 (2.09e-4) −	5.0368e-2 (9.58e-5)
SDTLZ3	3	6.8050e-2 (3.08e-3) −	4.7493e-1 (9.90e-3) −	3.4169e-1 (1.12e-3) −	1.3736e-1 (1.21e-2) −	1.3067e+0 (2.36e+0) −	5.1217e-2 (7.75e-4)
SDTLZ4	3	9.5373e-2 (1.61e-1) −	1.0637e-1 (1.59e-1) −	6.0224e-1 (2.58e-1) −	2.7603e-1 (2.45e-1) −	2.9495e-1 (3.71e-2) −	5.0329e-2 (1.06e-5)
$+/-/\approx$		0/8/0	0/8/0	0/8/0	0/8/0	0/8/0	

Fig. 4.7 shows the obtained Pareto fronts with medium value of IGD metric of all six algorithms for 3-objective SDTLZ3. We can find that only DBEA-AWV can get the true Pareto front of the scaled SDTLZ3 with better convergence and diversity.

4.3.3 Comparative Results on MaOPs

Table 4.3 presents the HV metric values obtained by NSGA-III, RVEA, MOEA/DD, MOEA/D-DU, MOEA/D-PaS, and DBEA-AWV on different MaOPs, including many-objective DTLZ1-4 and SDTLZ1-4. The proposed DBEA-AWV could achieve the best performance on 15 of 24 instances, while the number of best results obtained by NSGA-III, RVEA, MOEA/DD, MOEA/D-DU and MOEA/D-PaS are 1, 0, 2, 6, and 0, respectively. Although NSGA-III can get only one best result, it has the stable performance for most of the MaOPs. MOEA/D-DU and MOEA/DD can acquire some best results for normalized many-objective DTLZ problems, while they seem to encounter difficulties when facing the scaled test instances. The proposed DBEA-AWV performs much better

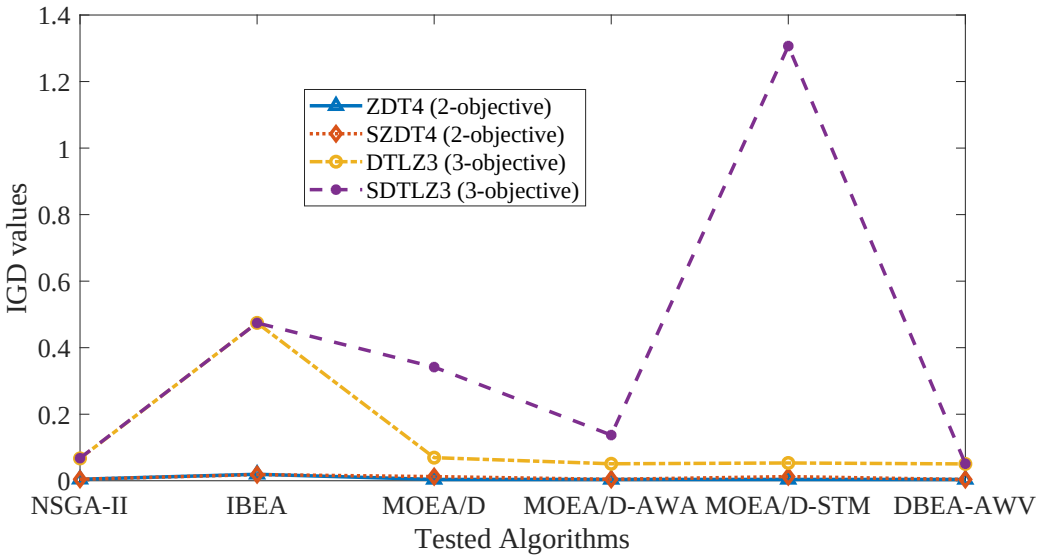


Figure 4.6: The IGD values obtained by different algorithms

than other compared algorithms on MaOPs with disparately scaled objectives. As shown in Table 4.3, DBEA-AWV is always reliable and effective to achieve good performance for both normalized and scaled MaOPs. Fig. 4.8 shows the HV values obtained by different tested algorithms. NSGA-III, RVEA, MOEA/DD, and MOEA/D-DU can get good results for 10-objective DTLZ1 but RVEA, MOEA/DD, and MOEA/D-DU show poor performance for scaled problems. The proposed DBEA-AWV is competitive for the normalized and scaled many-objective optimization problems.

Fig. 4.9 and Fig. 4.10 show the obtained Pareto fronts with medium value of HV metric of all six algorithms for 10-objective DTLZ1 and SDTLZ3, respectively. It can be observed that NSGA-III, RVEA, MOEA/DD, MOEA/D-DU and DBEA-AWV have obtained good approximations to true Pareto front of 10-objective DTLZ1. For 10-objective SDTLZ3, only NSGA-III, MOEA/D-DU and DBEA-AWV get the right fronts. Furthermore, DBEA-AWV can acquire a better performance with regard to convergence and diversity compared with NSGA-III, MOEA/D-DU. In general the proposed DBEA-AWV has a promising versatility of solving normalized and scaled MaOPs.

Compared with the classical decomposition-based algorithm MOEA/D [113], the obviously increased complexity of the proposed DBEA-AWV is the adaptative weight vectors operation. The weight vector adaptation is periodically executed and will not increase a lot of added overhead. However, the traditional decomposition-based algorithms can not get good results of scaled many-objective optimization problems, whereas the proposed DBEA-AWV can deal well with normalized and scaled multi- and many-objective optimization problems.

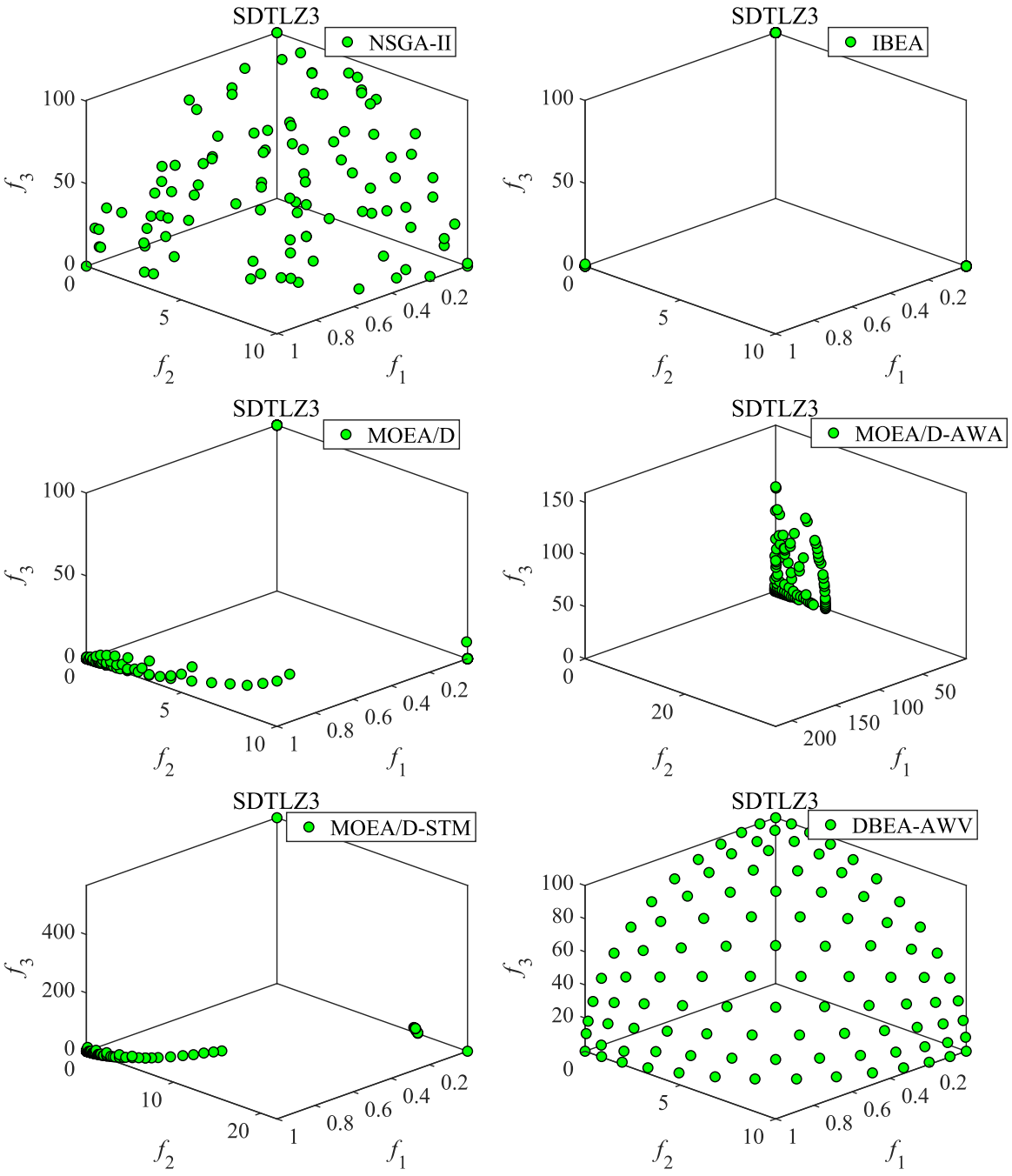


Figure 4.7: The obtained Pareto fronts of tested algorithms on 3-objective SDTLZ3

Table 4.3: The HV values obtained by tested algorithms

Problem	m	NSGA-III	RVEA	MOEA/DD	MOEA/D-DU	MOEA/D-PaS	DBEA-AWV
DTLZ1	5	9.7985e-1 (1.62e-4) ≈	9.7982e-1 (1.18e-4) ≈	9.7989e-1 (1.53e-4) ≈	9.7990e-1 (1.60e-4) ≈	5.5473e-1 (4.53e-1) −	9.7985e-1 (1.56e-4)
	8	9.9714e-1 (1.12e-3) −	9.9761e-1 (4.64e-5) +	9.9756e-1 (6.69e-5) +	9.9765e-1 (6.60e-5) ≈	6.1110e-2 (1.67e-1) −	9.9732e-1 (1.42e-3)
	10	9.9680e-1 (1.32e-2) −	9.9969e-1 (1.87e-5) −	9.9967e-1 (1.46e-5) −	9.9969e-1 (1.80e-5) ≈	2.7064e-5 (1.48e-4) −	9.9970e-1 (1.58e-5)
DTLZ2	5	8.1254e-1 (4.40e-4) ≈	8.1261e-1 (4.47e-4) ≈	8.1263e-1 (4.13e-4) ≈	8.1263e-1 (3.38e-4) ≈	7.6761e-1 (2.02e-2) −	8.1250e-1 (3.68e-4)
	8	9.1111e-1 (3.34e-2) −	9.2411e-1 (2.45e-4) −	9.2411e-1 (2.73e-4) −	9.2436e-1 (8.79e-4) −	4.6198e-1 (2.37e-1) −	9.2475e-1 (3.91e-4)
	10	9.5953e-1 (2.12e-2) −	9.6976e-1 (1.58e-4) −	9.6979e-1 (1.82e-4) −	9.7019e-1 (3.45e-4) −	1.5943e-1 (1.90e-1) −	9.7043e-1 (2.01e-4)
DTLZ3	5	8.1105e-1 (1.11e-3) ≈	8.1131e-1 (1.04e-3) ≈	8.1190e-1 (7.01e-4) +	8.1191e-1 (6.71e-4) +	4.6863e-1 (3.02e-1) −	8.1141e-1 (8.05e-4)
	8	8.7304e-1 (1.73e-1) −	9.2233e-1 (1.39e-3) ≈	9.2182e-1 (1.62e-3) ≈	9.2242e-1 (1.60e-3) ≈	4.7565e-2 (5.00e-2) −	9.1732e-1 (1.54e-2)
	10	9.2389e-1 (1.56e-1) ≈	9.6936e-1 (4.14e-4) +	9.6950e-1 (2.36e-4) +	9.6880e-1 (5.82e-4) ≈	4.8485e-2 (4.61e-2) −	9.6021e-1 (1.22e-2)
DTLZ4	5	8.0895e-1 (1.98e-2) ≈	8.1264e-1 (2.81e-4) ≈	8.1270e-1 (4.78e-4) ≈	8.1260e-1 (3.98e-4) ≈	7.5010e-1 (3.86e-2) −	8.1276e-1 (4.83e-4)
	8	9.1312e-1 (3.64e-2) −	9.2400e-1 (2.16e-4) −	9.2406e-1 (1.87e-4) −	9.2665e-1 (5.08e-4) +	8.9996e-1 (6.68e-3) −	9.2617e-1 (3.53e-4)
	10	9.6840e-1 (6.21e-3) −	9.6975e-1 (1.81e-4) −	9.6979e-1 (1.79e-4) −	9.7123e-1 (1.92e-4) +	9.5729e-1 (7.90e-3) −	9.7106e-1 (2.36e-4)
SDTLZ1	5	9.7833e-1 (2.96e-3) −	7.0877e-1 (1.48e-1) −	4.4899e-1 (6.13e-3) −	9.3765e-1 (4.68e-4) −	6.1446e-1 (4.65e-1) −	9.7978e-1 (3.18e-4)
	8	9.9638e-1 (3.12e-3) ≈	8.5694e-1 (6.72e-2) −	6.6415e-1 (1.21e-2) −	9.9470e-1 (4.49e-4) −	3.0818e-2 (8.90e-2) −	9.9729e-1 (5.77e-4)
	10	9.9467e-1 (1.39e-2) ≈	9.7052e-1 (9.45e-3) −	8.2216e-1 (8.84e-3) −	9.9927e-1 (4.96e-5) −	5.8189e-2 (1.60e-1) −	9.9961e-1 (9.88e-5)
SDTLZ2	5	8.1255e-1 (4.18e-4) +	7.7032e-1 (4.27e-3) −	2.4895e-1 (1.89e-2) −	6.7704e-1 (1.05e-3) −	7.5242e-1 (5.37e-2) −	8.1221e-1 (5.38e-4)
	8	9.2012e-1 (1.10e-2) −	7.7615e-1 (3.49e-2) −	3.2121e-1 (1.59e-2) −	8.9447e-1 (2.91e-3) −	4.8659e-1 (2.49e-1) −	9.2255e-1 (7.29e-4)
	10	9.6749e-1 (6.36e-3) ≈	9.1550e-1 (9.00e-3) −	5.0061e-1 (2.32e-2) −	9.5674e-1 (1.24e-3) −	1.2844e-1 (1.41e-1) −	9.6933e-1 (4.21e-4)
SDTLZ3	5	8.0842e-1 (6.29e-3) −	2.9667e-1 (1.66e-1) −	2.6951e-1 (1.55e-2) −	6.7218e-1 (5.08e-3) −	5.1518e-1 (3.01e-1) −	8.0941e-1 (9.55e-4)
	8	8.4412e-1 (2.08e-1) ≈	4.7586e-1 (1.24e-1) −	3.2048e-1 (1.84e-2) −	8.7872e-1 (7.63e-3) −	8.2632e-2 (1.13e-1) −	9.0631e-1 (1.74e-2)
	10	9.1410e-1 (1.21e-1) ≈	7.8909e-1 (7.86e-2) −	4.9249e-1 (1.57e-2) −	9.5030e-1 (2.14e-3) −	6.9697e-2 (3.91e-2) −	9.6090e-1 (5.65e-3)
SDTLZ4	5	8.1213e-1 (4.80e-4) ≈	7.8040e-1 (2.00e-3) −	2.8442e-1 (1.94e-2) −	6.7723e-1 (4.42e-4) −	7.5781e-1 (3.21e-2) −	8.1232e-1 (4.05e-4)
	8	9.1587e-1 (2.38e-2) −	8.4798e-1 (5.48e-3) −	3.6363e-1 (3.29e-2) −	8.9998e-1 (2.86e-4) −	8.9667e-1 (1.96e-2) −	9.2619e-1 (3.18e-4)
	10	9.6701e-1 (9.66e-3) −	9.4574e-1 (2.87e-3) −	5.4249e-1 (1.74e-2) −	9.6072e-1 (2.02e-4) −	9.5751e-1 (6.86e-3) −	9.7110e-1 (1.49e-4)
$+/-/\approx$		1/12/11	2/17/5	3/17/4	3/14/7	0/24/0	

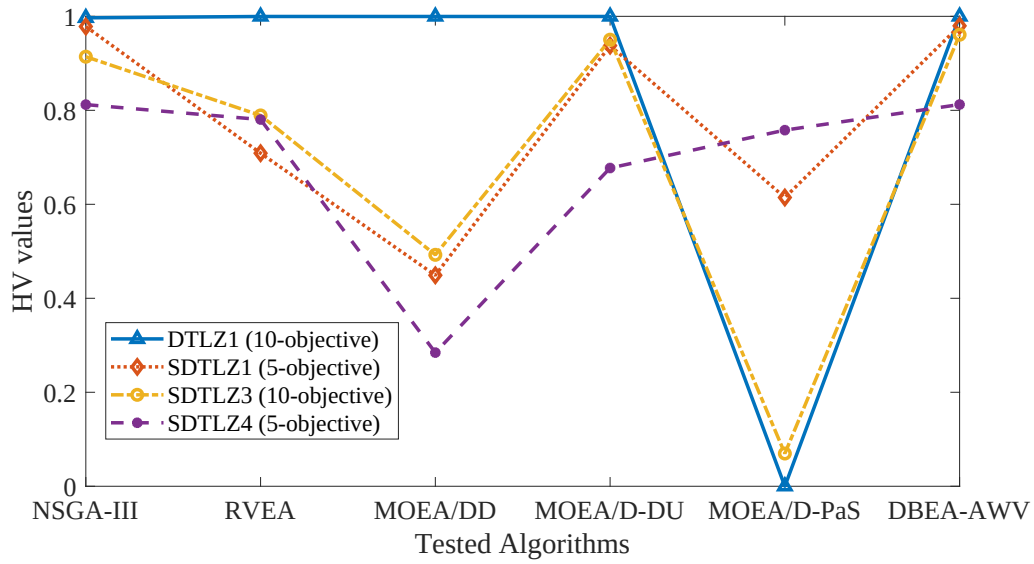


Figure 4.8: The HV values obtained by different algorithms

4.3.4 Parameter Sensitivity Analysis

There are two major parameters to be specified in the proposed DBEA-AWV, i.e., K controlling the balance between convergence and diversity and f_r controlling frequency of employing the weight vector adaptation. To investigate the sensitivity of these two parameters, different settings of K and f_r are used in DBEA-AWV on DTLZ2 and SDTLZ3, which represent normalized and scaled problems, respectively.

We first analyze the sensitivity of parameter K, where K varies from 1 to 20 and f_r is fixed to 0.2. The average HV values obtained by DBEA-AWV with different K over 30 independent runs are shown in Fig. 4.11. Two observations from the results can be noted. The first observation is that DTLZ2 having a simple search landscape is not sensitive to the parameter K. Second, SDTLZ3 is a representative disparately scaled problem with multimodality, where the performance starts to be better with increasing values of K in the beginning. However, the optimization results cannot always be better with increasing of K when K becomes larger than 15. It implies that a suitable setting of K can achieve a better balance between convergence and diversity especially for many-objective optimization. Based on the analysis, K is recommended to be selected between $[5, 15]$.

The sensitivity analysis of parameter f_r is carried out, where f_r varies from 0.01 to 0.5 and K is fixed to 5. The average HV values obtained by DBEA-AWV with different f_r over 30 independent runs are shown in Fig. 4.12. It can be observed that the normalized problem DTLZ2 having the

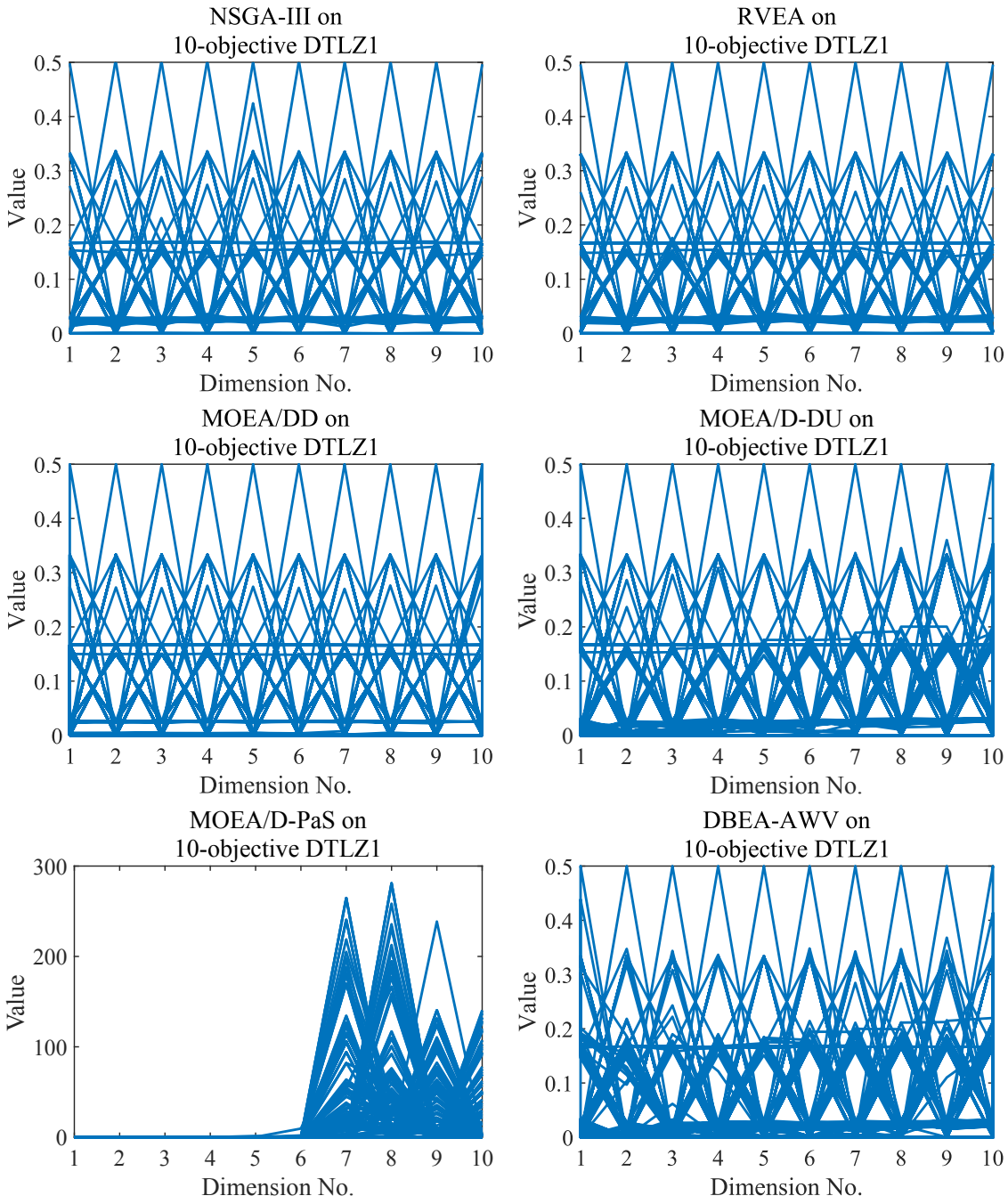


Figure 4.9: The obtained Pareto fronts of tested algorithms on 10-objective DTLZ1

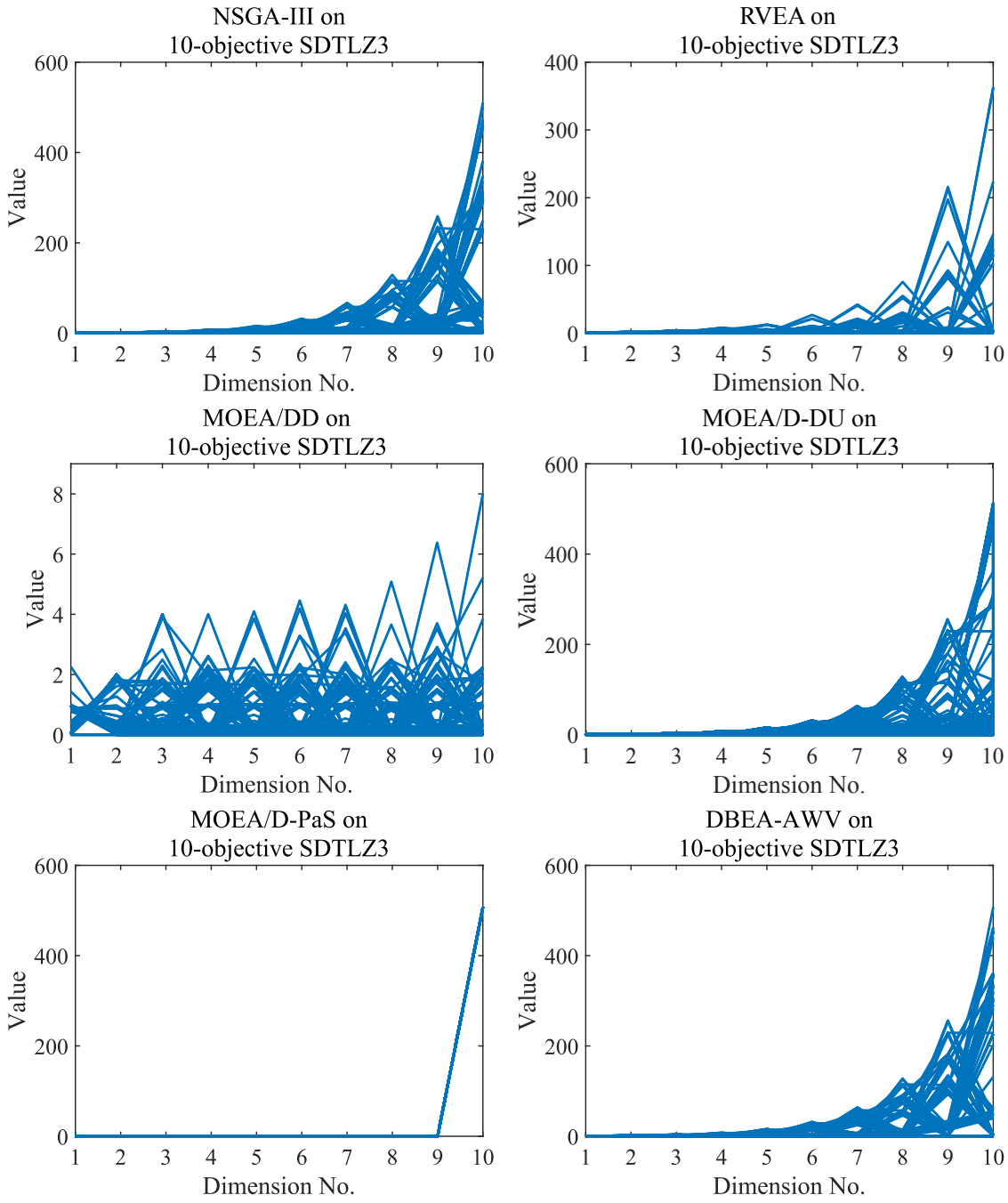


Figure 4.10: The obtained Pareto fronts of tested algorithms on 10-objective SDTLZ3

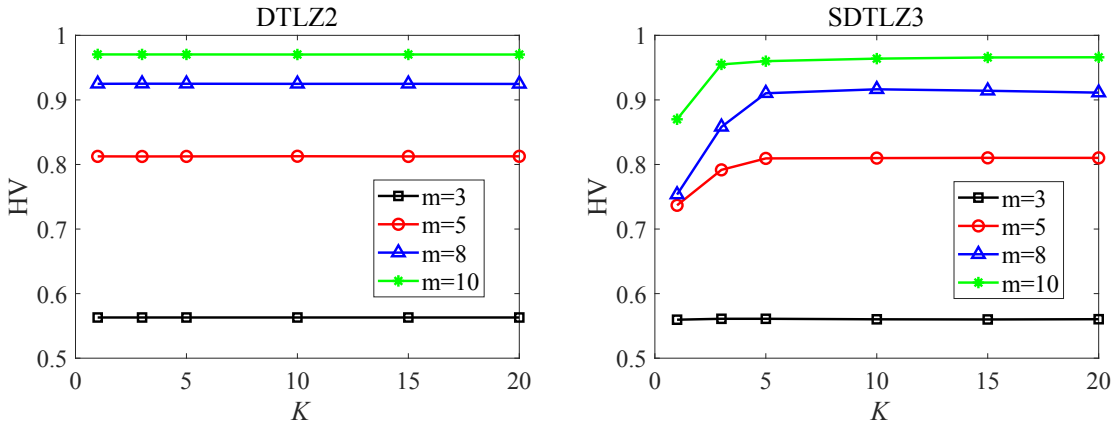


Figure 4.11: The average HV values obtained by DBEA-AWV with different K

same range of objectives is insensitive to the parameter f_r. For SDTLZ3 with high dimensional objectives, a too small f_r, employing weight vector adaptation frequently, will lead to deterioration of performance of DBEA-AWV. On the other hand, a too large f_r is also not beneficial for the performance. Therefore, f_r is recommended to be selected between $[0.1, 0.3]$.

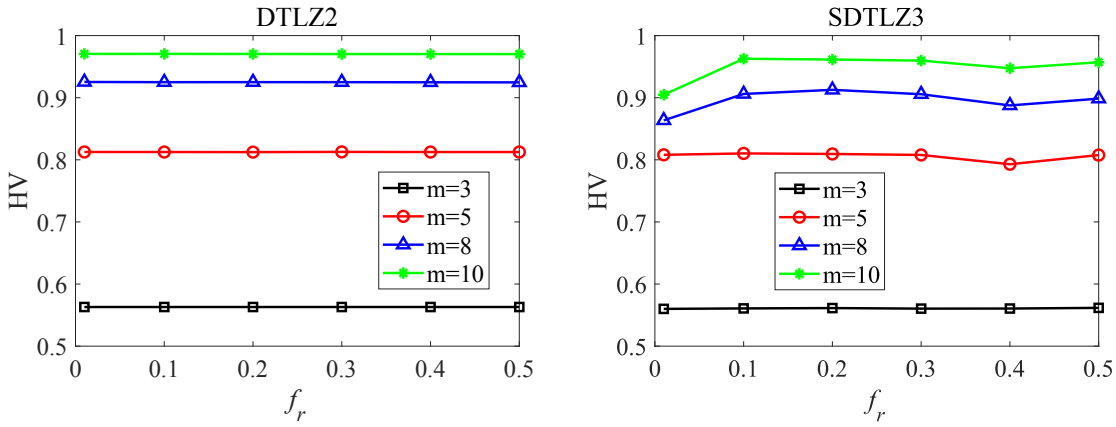


Figure 4.12: The average HV values obtained by DBEA-AWV with different f_r

4.4 Summary

In this chapter, we have proposed the DBEA-AWV algorithm for solving normalized and scaled MOPs and MaOPs. In order to improve the performance of DBEA-AWV on disparately scaled

problems, a strategy for adapting weight vectors is adopted to tune weight vectors according to the range of each objective with respect to candidate solutions. Based on the adaptive weight vectors, we compare several existing popular decomposition approaches and find that the modified Tchebycheff approach shows high efficiency in dealing with both normalized and scaled problems. What's more, a novel replacement strategy is developed to achieve a better balance between convergence and diversity especially for MaOPs.

To investigate the performance of DBEA-AWV, we compare DBEA-AWV with ten state-of-the-art MOEAs. The experimental results demonstrate that DBEA-AWV is competitive and efficient compared with other algorithms.

Chapter 5

An Adaptive Algorithm for Irregular Pareto Fronts

This chapter shows an adaptive algorithm (called AMAWV) for solving multi- and many-objective optimization problems (MOPs and MaOPs) with irregular Pareto fronts.

Most multi-objective evolutionary algorithms (MOEAs) may have the advantages of solving MOPs with regular Pareto fronts, but they often encounter difficulties in dealing with problems with complex irregular Pareto fronts. For instance, MOEA/D [113] is a general decomposition-based MOEA framework for decomposing a MOP into a number of single-objective (or multi-objective) optimization subproblems. The performance of solutions generated by decomposition-based evolutionary algorithms depends heavily on the Pareto front shapes [45].

In this chapter, we present a novel archive maintenance method for adapting weight vectors in the decomposition-based multi-objective evolutionary algorithms (called AMAWV) for solving MOPs and MaOPs with regular and irregular Pareto fronts. The main contributions of this chapter are summarized as follows:

- A novel archive maintenance strategy is proposed for deleting the dominance resistant solutions as well as retaining good diversity.
- An efficient weight vector adaptation method is presented for solving different MOPs and MaOPs with various Pareto front shapes.
- The proposed algorithm is competitive compared with the other five state-of-the-art algorithms on test problems with a variety of Pareto front shapes.

5.1 Irregular Pareto Fronts

In MOEA/D, each weight vector corresponds to one subproblem, ideally associated with one solution. A set of weight vectors will determine the search direction and the diversity of solutions is controlled by these weight vectors as well as Pareto front shapes. If the shape of a set of weight vectors shares a similar shape with Pareto front, the obtained solutions can be uniformly distributed on the Pareto front.

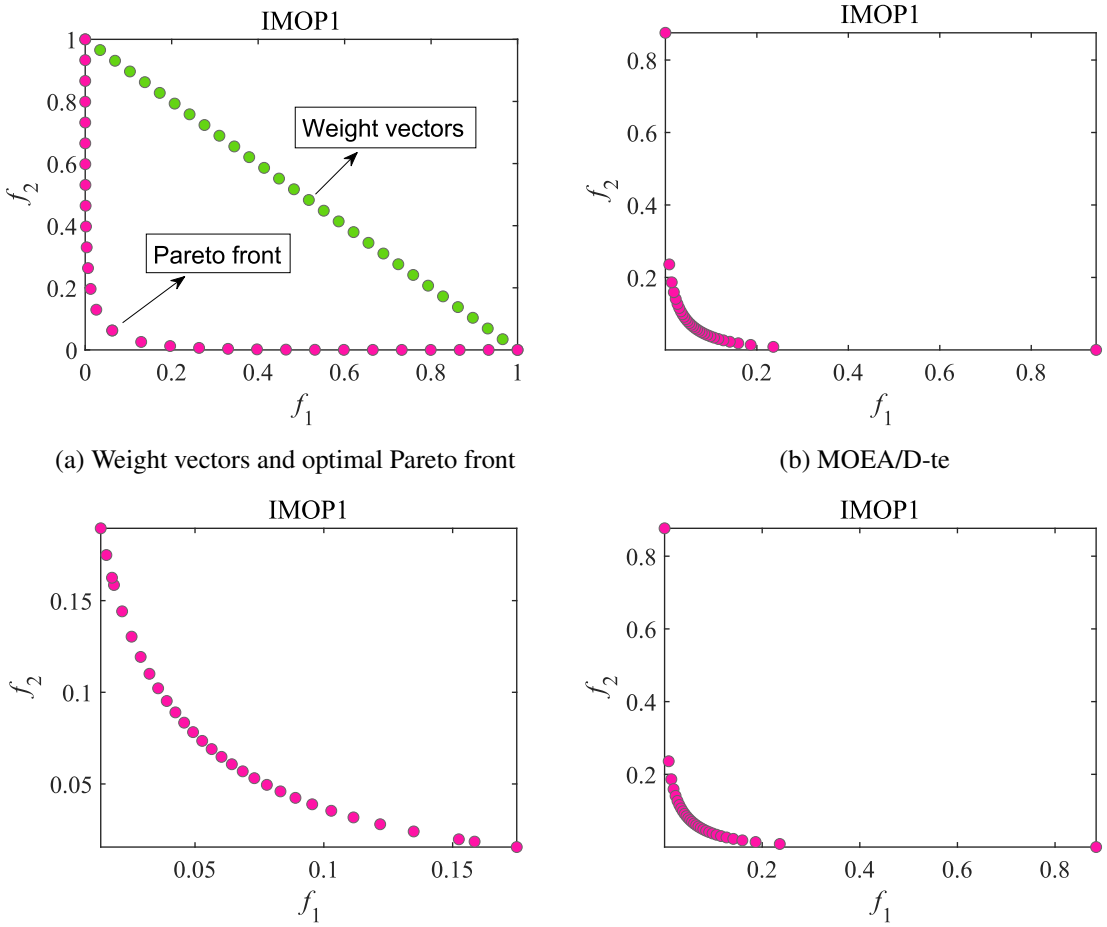
In general, the weight vectors are systematically generated and distributed uniformly in a unit simplex. In that way, if the MOPs have regular Pareto front (i.e., simplex-like), e.g., a triangle plane or a sphere, the decomposition-based algorithms can get a set of solutions suited to the Pareto front. However, when MOPs have irregular Pareto fronts (e.g., convex, concave, inverted, disconnected, degenerated or scaled) [63], a set of uniformly distributed weight vectors may not result in a set of uniformly distributed Pareto optimal solutions.

When using the decomposition-based evolutionary algorithms to deal with MOPs with irregular Pareto fronts, some weight vectors may have no intersection with the Pareto front and lead to several weight vectors associated with one solution. In this case, the number of obtained Pareto optimal solutions may be less than the weight vectors. Fig. 5.1 and Fig. 5.2 present two examples that the sets of Pareto optimal solutions are obtained by decomposition-based evolutionary algorithms on irregular Pareto fronts (convex IMOP1 and inverted IDTLZ1) [46, 86]. MOEA/D-te, MOEA/D-pbi and MOEA/D-mte represent that MOEA/D algorithm adopts the Tchebycheff approach, the penalty-based boundary intersection and modified Tchebycheff approach, respectively.

As shown in the first two subfigures (Figs. 5.1a and 5.2a), the shapes of weight vectors are different from the shapes of Pareto fronts. The Pareto front of IMOP1 is very convex, while the obtained Pareto optimal solutions of different decomposition approaches in MOEA/D are non-uniformly distributed and few solutions are found on the boundary. The Pareto front of IDTLZ1 is inverted, whereas the obtained Pareto optimal solutions of MOEA/D focus on the boundary and sparse solutions are generated in the center. The above examples elaborate on the difficulties of finding a set of weight vectors to be suitable for any MOP. It is necessary to adjust the weight vectors to be adaptive for MOPs with various Pareto fronts.

5.2 The Proposed AMAWV

This section presents the details of the proposed AMAWV.



(a) Weight vectors and optimal Pareto front

(b) MOEA/D-te

(c) MOEA/D-pbi

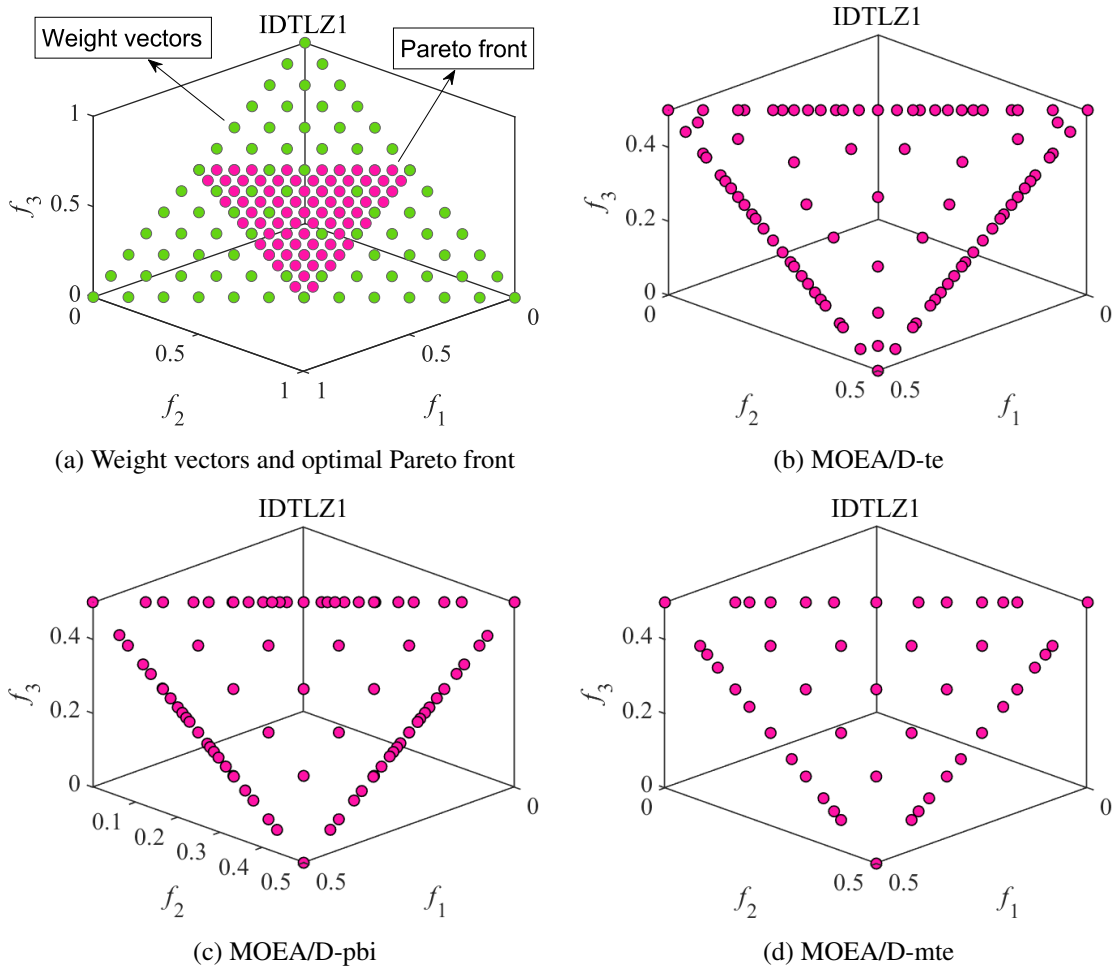
(d) MOEA/D-mte

Figure 5.1: The performance of different decomposition approaches in MOEA/D on IMOP1

5.2.1 General Framework

The general framework of the proposed AMAWV is described in Algorithm 5. First, the population $P \leftarrow \{x^1, x^2, \cdots, x^N\}$ are randomly generated in the whole decision space, then the reference point $z^* = (z_1^*, z_2^*, \cdots, z_m^*)^T$ is initialized. After that we can initialize the archive A by adding the non-dominated solutions from the population P. A set of uniformly random weight vectors $\Lambda = \{\lambda^1, \lambda^2, \cdots, \lambda^N\}$ are generated as follows [18].

First, 5000 weight vectors are uniformly randomly generated for forming the set Λ_1. Λ is initialized as the set containing all the weight vectors $(1\ 0\ ...\ 0\ 0), (0\ 1\ ...\ 0\ 0), ..., (0\ 0\ ...\ 0\ 1)$. Second, the weight vector in Λ_1 with the largest distance to Λ is found, added to Λ, and removed from Λ_1.



(a) Weight vectors and optimal Pareto front

(b) MOEA/D-te

(c) MOEA/D-pbi

(d) MOEA/D-mte

Figure 5.2: The performance of different decomposition approaches in MOEA/D on IDTLZ1

This process is repeated until the size of Λ is N.

The AMAWV adopts the uniformly random weight vectors method instead of Das and Dennis's systematic approach [17]. The advantage of this approach is that the population size is flexible, which is independent on the number of objectives. In the light of the generated weight vectors, the neighborhood set of subproblem i as $B(i) = \{i_1, \cdots, i_T\}$ can be obtained by computing the Euclidean distance, where T is the neighborhood size. Steps 8-28 in Algorithm 5 are iterated until the termination criterion is met. At each iteration, the mating pool is allowed to be selected from the whole population with a low probability $1 - \delta$. The widely used simulated binary crossover (SBX) and polynomial mutation are randomly selected mating solutions from E to generate offspring y.

Then offspring y is used to update the reference point, the population and the archive. When the size of the archive exceeds the predefined limit size (N_A), a novel strategy is adopted to maintain the archive (Steps 20-22 in Algorithm 5), which will be introduced in Section 5.2.2. When the frequency of updating the weight vectors is satisfied with the designed requirements, the weight vector adaptation method is conducted (Steps 23-26 in Algorithm 5), which will be presented in Section 5.2.3.

5.2.2 Archive Maintenance

In AMAWV, we use the Pareto dominance to select the non-dominated solutions to be added to the archive, when the size of the archive exceeds the pre-set capacity (N_A). A novel archive maintenance strategy is applied to remove some dominance resistance solutions and some other solutions with poor distribution. The archive maintenance strategy is presented in Algorithm 6.

Aiming at dealing with the scaled problems with different objectives ranges, the normalization approach in NSGA-III [22] is adopted to normalize the solutions in the archive (Step 2 in Algorithm 6). The main idea of this approach is to find some extreme solutions to construct the hyperplane. The extreme solutions are determined by minimizing the achievement scalarizing function (ASF):

$$ASF\left(x,w\right)=\max_{i=1}^{m}\frac{f_i\left(x\right)-z_i^*}{w_i},\quad for\ x\in S_t \tag{5.1}$$

where S_t represents the current population. w is the axis direction, $w_i=10^{-6}$ when it is zero. After m extreme solutions have been adopted, they are used to construct a hyperplane and the intercept a_j of the j-th objective axis on the hyperplane can be computed. Then the solutions are normalized as follows:

$$\bar{f}_j\left(x\right)=\frac{f_j\left(x\right)-z_j^*}{a_j} \tag{5.2}$$

where $f_j\left(x\right)$ is the j-th objective value of solution x, $\bar{f}_j\left(x\right)$ is the normalized objective value.

According to the non-dominated solutions in the archive during the evolution process, we can estimate the shape of PF (Step 3 in Algorithm 6). The PF shape can guide the search direction [102]. First, the m solutions in the archive closest to the m-dimensional vector $V=(1,1,\ldots,1)$ are identified based on the angle between the non-dominated solutions and vector V. Then the ratio

$$r=\frac{\overline{d}}{d^{\perp}} \tag{5.3}$$

is used to estimate the PF shape, where $\overline{d}$ is the average Euclidean distance from m closest solutions

Algorithm 5: Framework of AMAWV

Input: A set of weight vectors $\Lambda \leftarrow \left\{\lambda^1, \lambda^2, \cdots, \lambda^N\right\}$, the maximum number of generations $t_{\max}$

Output: The final population P

Initialize the population $P \leftarrow \left\{x^1, x^2, \cdots, x^N\right\}$;
Initialize the reference point $z^* \leftarrow \left(z_1^*, z_2^*, \cdots, z_m^*\right)^T$;
Initialize the archive A;
for $i = 1 : N$ **do**
    $B\left(i\right) \leftarrow \{i_1, i_2, \cdots, i_T\}$, where $\lambda^{i_1}, \lambda^{i_2}, \cdots, \lambda^{i_T}$ are T closest weight vectors to λ^i;
end
$t \leftarrow 1$;
while $t < t_{\max}$ **do**
    for $i = 1 : N$ **do**
        if $uniform\left(0, 1\right) < \delta$ **then**
            $E \leftarrow B\left(i\right)$;
        else
            $E \leftarrow \{1, 2, \ldots, N\}$;
        end
        $y = \text{offspring\_creation}\left(E\right)$;
        $z^* = \text{Update\_Ideal\_Point}\left(y, z^*\right)$;
        $P = \text{Update\_Population}\left(y, z^*, \Lambda, P\right)$;
        $A = \text{Update\_Archive}\left(y, A\right)$;
    end
    if $|A| > N_A$ **then**
        Maintain the archive A;
    end
    if $t > t_{\max} \times 10\% \wedge t == t_{\max} \times 5\% \wedge t < t_{\max} \times 90\%$ **then**
        $\Lambda = \text{Weight\_Vector\_Adaption}\left(t, P, A, \Lambda\right)$;
        Update the neighborhood set of each weight vector of Λ;
    end
    $t = t + 1$;
end

Algorithm 6: Archive Maintenance

Input: The archive A ($|A| > N_A$)
Output: The new archive $newA$

1 $newA \leftarrow \phi$;
2 $A \leftarrow \text{normalization}\,(A)$;
3 $r \leftarrow estimateShape(A)$;
4 Set the reference point z^* to z^{nad} if $r < 1.1$, or itself otherwise;
5 $\{S+, S-\} \leftarrow classificationByHypercube\,(A)$;
6 **if** $|S+| > N_A$ **then**
7 /*——— $newA \leftarrow \text{selection}\,(S+, S-)$ ———*/
8 Add m extreme solutions into $newA$ and remove them from $S+$;
9 **repeat**
10 Add into $newA$ the solution in $S+$ that has the maximum angle to $newA$;
11 **until** $|newA| = N_A$
12 **else**
13 $newA = S+$;
14 **end**

to the coordinate origin, and $d^{\perp}$ is the Euclidean distance from coordinate origin to the hyperplane $\sum_{i=1}^{m} f_i = 1$. Since $d^{\perp} = \frac{|-1|}{\sqrt{m}} = \frac{|1|}{\sqrt{m}}$, we obtain

$$r = \overline{d} \times \sqrt{m} \tag{5.4}$$

Therefore, the shape of the PF can be estimated as convex (if $r < 0.9$), linear (if $r \in [0.9, 1.1]$), and concave (if $r > 1.1$).

According to the estimated PF shape, we set the reference point z^* to z^{nad} if $r < 1.1$, or itself otherwise (Step 4 in Algorithm 6). The coordinates of the nadir point correspond to the maximum value of each objective on the true Pareto front. The nadir point z^{nad} is usually estimated to be $z^{nad} = z^{\max} = (z_1^{\max}, z_2^{\max}, \cdots, z_m^{\max})$. But here we set the $z^{nad} = I = (1, 1, \cdots, 1)$ after normalizing the objectives. Then the solutions in the archive can be divided into two repositories ($S+$ and $S-$) inside and outside the hypercube (Step 5 in Algorithm 6), which is bounded by z^* and z^{nad}.

The new archive ($newA$) will select the non-dominated solutions from the $S+$ and $S-$ (Steps 6-14 in Algorithm 6). When the size of $S+$ is beyond N_A, the m extreme solutions are added into $newA$ and removed from $S+$. Then we use the one by one adding solution procedure to choose the solutions from $S+$. At each stage, the solution in $S+$ that has the maximum angle to $newA$ will

be placed into the $newA$ [101]. The above operation is repeated until the size of the new archive is equal to N_A. When the size of $S+$ is smaller than N_A, all the solutions in $S+$ will be added into $newA$. In the literature [102] the Pareto-adaptive reference points are used to calculate fitness values and the union population selects both the dominated and non-dominated solutions based on the fitness values. Also, the solutions outside the hypercube can be added to the population. In this chapter, only the non-dominated solutions inside the hypercube are added to the archive to adjust the weight vectors for various PF shapes, which can avoid the dominance resistant solutions to improve the convergence.

The reference point set ($z^* \leftarrow z^{nad}$ if $r < 1.1$) plays an important role in the archive maintenance strategy, we will explain the process using the 2-dimensional example as shown in Fig. 5.3. Through the adaptive normalization procedure and classification by the hypercube, we can delete some dominance resistant solutions such as the solution C (or D), if C has an extremely poor value in the second objective (f_2) but has optimal value in the first objective (f_1). In this way, some extremely poor solutions in the archive can be deleted and will not guide the solutions to search in the wrong direction. This improves the convergence of the algorithm. On the other hand, in the one by one adding solution procedure, we use the maximum angle as the criterion to select a solution with good distribution. If the PF shape is very convex, there are many solutions along the coordinate (such as A and B), the angle between the vector $\overrightarrow{z^*A}$ and $\overrightarrow{z^*B}$ is close to zero. At this time, it is easy to ignore the solutions along the coordinate and finally obtain the non-dominated solution set with poor distribution. But the angle between the vector $\overrightarrow{z^{nad}A}$ and $\overrightarrow{z^{nad}B}$ is clear in this situation, we can differentiate these solutions by substituting the reference point z^* with z^{nad}. Therefore, we use different reference points (z^* or z^{nad}) based on the different PF shapes in order to obtain a good spread of non-dominated solutions in the archive.

5.2.3 Weight Vector Adaptation

The non-dominated solutions in the archive can reflect the PF shape to some extent, hence it is necessary to use the information of the archive to guide the search direction of the decomposition-based algorithms for solving problems with irregular Pareto fronts. The weight vector addition and deletion method is an effective way to adapt the weight vectors for various PF shapes. We select an efficient strategy to add and delete some weight vectors.

Inspired by the literature [63], we first compare the population with the archive to find the undeveloped solutions. If a solution in the archive is located in a niche which has no solution in the population, the solution is considered as an undeveloped solution. The radius of the niche is set to the median of the distances from all the solutions to their closest solution in the archive. After

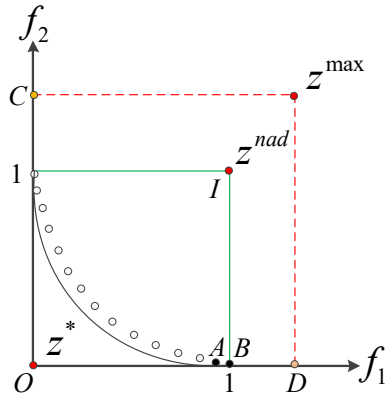


Figure 5.3: Illustration of the reference points

finding all the undeveloped solutions we then compute the corresponding weight vectors of these solutions. Formally, let $z^* \leftarrow (z_1^*, z_2^*, \cdots, z_m^*)^T$ be the reference point and $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^T$ be the optimal weight vector to a solution q in the modified Tchebycheff approach. Then it holds that

$$\frac{f_1(q) - z_1^*}{\lambda_1} = \frac{f_2(q) - z_2^*}{\lambda_2} = \cdots = \frac{f_m(q) - z_m^*}{\lambda_m} \tag{5.5}$$

Since $\lambda_1 + \lambda_2 + \cdots + \lambda_m = 1$, we get

$$\lambda = \left(\frac{f_1(q) - z_1^*}{\sum_{i=1}^m f_i(q) - z_i^*}, \cdots, \frac{f_m(q) - z_m^*}{\sum_{i=1}^m f_i(q) - z_i^*} \right) \tag{5.6}$$

After obtaining the undeveloped solutions and their corresponding weight vectors, we need to determine whether the found undeveloped solutions are promising or not. For each of these undeveloped weight vectors, we find its neighboring weight vectors as well as their associated solutions in the population. Let q be an undeveloped solution in the archive and λ^q be its corresponding weight vector. Let λ^p be one of the neighboring weight vectors of λ^q and p be its associated solution in the population. The solution q outperforms p if

$$g(q, \lambda^q) < g(p, \lambda^q) \tag{5.7}$$

or

$$g(q, \lambda^q) = g(p, \lambda^q) \;\; and \;\; \sum_{i=1}^m f_i(q) < \sum_{i=1}^m f_i(p) \tag{5.8}$$

where $g()$ denotes the modified Tchebycheff function, $f_i()$ denotes the i-th objective function, and

m is the number of objectives. If the undeveloped solution q outperforms all of its neighboring solutions in the population on the basis of λ^q, q will be considered the promising solution and added to the population.

After placing the undeveloped and promising solutions with their corresponding weight vectors into the population, the number of solutions and weight vectors in the new population may be beyond the predefined size N, hence the weight vector deletion operation needs to be applied. The one by one removing solution procedure is designed to delete the redundant population solutions and corresponding weight vectors. That means at each time, the solution (along with its weight vector) in the population which has minimum distance to another solution is chosen to be deleted. If there are several solutions with the same minimum distance the second smallest distance will be considered and so forth [124]. The deletion process will repeat until the number of weight vectors is equal to N.

The frequency of updating the weight vectors plays an important part of the performance of decomposition-based algorithms. The frequent change of weight vectors may deteriorate the convergence of these algorithms. In AMAWV the weight vectors do not change during the first 10% and last 10% generations for retaining the same search direction to improve convergence. The weight vector adaptation operation is conducted every 5% of the total generations in the middle evolution process.

The weight vector deletion process and the frequency of updating the weight vectors in the proposed algorithm are different from the literature [63]. The weight vector deletion process is more efficient and the frequency of updating the weight vectors can improve the whole convergence. In addition, the novel archive maintenance strategy for avoiding the dominance resistant solutions is used to maintain the archive, which can guide the weight vectors addition and deletion process to balance the convergence and diversity.

5.2.4 Computational Complexity

In the proposed AMAWV, the major computational costs are the iteration process in Algorithm 5. For one generation of AMAWV, Step 15 needs $O(N)$ operations to produce the offspring. Step 16 needs $O(mN)$ comparisons to update the reference point. Step 17 performs $O\left(mN^2\right)$ operations to update the population at the worst case. Step 18 needs $O\left(mNN_A\right)$ comparisons to update the archive. Step 21 requires $O\left(mN_A^2\right)$ operations to maintain the archive.

On average, the weight vector adaptation needs $O\left(N_A^2 \log N_A\right)$ operations and it needs $O\left(N^2\right)$ comparisons to update the neighborhood set. Hence, the overall computational complexity at one generation of AMAWV is $\max\left\{O\left(N_A^2 \log N_A\right), O\left(mN_A^2\right)\right\}$.

5.3 Experimental Studies

In this section empirical experiments are conducted on MOPs and MaOPs with different PF shapes to compare AMAWV with other five state-of-the-art algorithms.

5.3.1 Experimental Design

Five state-of-the-art algorithms, MOEA/D [113], MOEA/D-AWA [75], NSGA-III [22], RVEA [11] and VaEA [101] are chosen to evaluate the performance of AMAWV. In MOEA/D, the modified Tchebycheff approach is also applied.

The test problems are chosen from widely used benchmark test suites. They are categorized into eight groups according to different PF shape properties [63]: simplex-like (DTLZ1, DTLZ2, CDTLZ2 and MaF3), inverted (IDTLZ1 and IDTLZ2), highly nonlinear (SCH1 and FON), disconnected (ZDT3 and DTLZ7), degenerated (DTLZ5 and DTLZ6), scaled (SDTLZ1 and SDTLZ2), mixed (SCH2 and MaF4) and high-dimensional (DTLZ2-10 and IDTLZ1-10). The test problems in the first seven groups are 2- or 3-objective and the last group are 10-objective.

The inverted generational distance (IGD) [125] and hypervolume (HV) [125] are adopted to evaluate the performance of the compared algorithms. The smaller IGD and larger HV mean better.

The neighborhood size T is set to $0.1N$, the probability δ of selecting from the mating pool is set to 0.9, the capacity N_A of the archive is $2N$. The crossover probability and distribution index of SBX are set to $p_c = 1$ and $\eta_c = 20$, respectively. For polynomial mutation, the mutation probability and distribution index are set to $p_m = 1/n$ and $\eta_m = 20$, where n is the number of decision variables.

The population size in decomposition-based algorithms cannot be arbitrary. For a fair comparison, we set the population size N to 100, 105 and 275 for the 2-, 3- and 10-objective problems, respectively. The maximal number of generations is set to 1000 for all the problems. Each algorithm is executed 30 times independently on each test instance, and the average and standard deviation of the metric values are recorded. The Wilcoxon rank sum test at a 5% significance level is used to compare the experimental results, where the symbol '$+$', '$-$' and '$\approx$' denotes that the result of another algorithm is significantly better, significantly worse and similar to that obtained by AMAWV, respectively.

5.3.2 Experimental Results

Table 5.1 presents the IGD metric values obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on the test problems with different PF shapes. The best result in each

row is highlighted. The proposed algorithm has achieved the best performance on 13 of 18 test instances, while the number of best results obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA are 1, 1, 1, 2 and 0, respectively. It can be seen that the proposed AMAWV has a clear advantage over other compared algorithms with regard to the IGD metric. Figs. 5.4 and 5.5 show the final non-dominated solution set with the median IGD value obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on 3-objective MaF3 and 3-objective DTLZ7. For 3-objective MaF3, the PF shape is convex, MOEA/D, NSGA-III, RVEA and VaEA focus on the center part of PF, MOEA/D-AWA could tune the diversity of the solutions a little, AMAWV can achieve the best balance between convergence and diversity. For 3-objective DTLZ7 the PF shape is disconnected, while VaEA and AMAWV can obtain the solution set with good distribution compared with the other four algorithms.

Table 5.1: The IGD values obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on test problems

Problem	MOEA/D	MOEA/D-AWA	NSGA-III	RVEA	VaEA	AMAWV
DTLZ1	1.8983e-2 (9.28e-6) +	1.9741e-2 (4.76e-4) ≈	1.8989e-2 (2.69e-5) +	1.8981e-2 (9.20e-6) +	2.7894e-2 (8.25e-3) −	1.9675e-2 (1.69e-4)
DTLZ2	5.0315e-2 (6.41e-6) +	5.0674e-2 (1.91e-4) +	5.0301e-2 (1.23e-6) +	5.0301e-2 (1.35e-6) +	5.3517e-2 (6.94e-4) −	5.1491e-2 (4.15e-4)
CDTLZ2	4.3371e-2 (4.40e-6) −	3.3688e-2 (1.20e-3) −	4.3347e-2 (1.68e-4) −	3.8134e-2 (4.14e-4) −	5.8261e-2 (4.21e-3) −	3.1819e-2 (4.98e-4)
MaF3	4.3944e-2 (5.50e-4) −	3.6480e-2 (1.24e-3) −	4.4005e-2 (6.12e-4) −	3.9123e-2 (9.02e-4) −	1.0286e-1 (6.82e-2) −	3.2005e-2 (6.31e-4)
IDTLZ1	3.2323e-2 (7.86e-6) −	2.0001e-2 (1.47e-4) −	2.7614e-2 (5.36e-4) −	4.5021e-2 (1.16e-2) −	2.9573e-2 (1.21e-2) −	1.9765e-2 (1.41e-4)
IDTLZ2	9.7749e-2 (1.85e-5) −	5.2547e-2 (5.48e-4) −	6.8724e-2 (2.75e-3) −	7.6365e-2 (8.07e-4) −	6.9191e-2 (2.00e-3) −	5.2189e-2 (4.33e-4)
SCH1	4.7648e-2 (9.46e-5) −	1.7135e-2 (7.35e-5) ≈	4.7622e-2 (1.34e-4) −	4.4682e-2 (2.16e-4) −	5.5476e-2 (6.59e-3) −	1.7146e-2 (8.56e-5)
FON	3.5827e-3 (1.04e-5) +	3.6737e-3 (2.93e-5) +	3.5951e-3 (7.79e-6) +	4.6246e-3 (3.89e-4) −	4.6102e-3 (9.91e-5) −	3.9019e-3 (7.87e-5)
ZDT3	1.0967e-2 (4.34e-5) −	4.8912e-3 (6.16e-5) −	1.0013e-2 (9.94e-3) −	8.4826e-3 (9.45e-4) −	1.3966e-2 (1.29e-2) −	4.7013e-3 (1.12e-4)
DTLZ7	2.4607e-1 (1.85e-1) −	1.3131e-1 (9.16e-2) −	7.1013e-2 (3.20e-3) −	1.0369e-1 (2.33e-3) −	5.9058e-2 (1.32e-3) −	5.3663e-2 (7.62e-4)
DTLZ5	1.8610e-2 (2.32e-6) −	4.9479e-3 (8.68e-5) −	1.2185e-2 (1.64e-3) −	5.9830e-2 (2.74e-3) −	4.6950e-3 (1.33e-4) −	4.0480e-3 (6.67e-5)
DTLZ6	1.8612e-2 (1.83e-6) −	4.8461e-3 (1.26e-4) −	1.6910e-2 (2.32e-3) −	6.3984e-2 (1.12e-2) −	4.3482e-3 (1.50e-4) −	4.0338e-3 (4.62e-5)
SDTLZ1	2.7841e+0 (5.43e-3) −	2.1104e+0 (1.14e+0) −	9.6234e-1 (2.85e-2) −	8.9768e-1 (4.91e-2) −	8.1286e-1 (3.72e-1) −	6.0928e-1 (1.68e-2)
SDTLZ2	5.2277e+0 (5.55e-4) −	1.8208e+0 (3.04e-1) −	1.4903e+0 (3.61e-4) ≈	1.4973e+0 (8.44e-3) ≈	1.5179e+0 (8.60e-2) ≈	1.5013e+0 (6.69e-2)
SCH2	1.0517e-1 (8.89e-5) −	2.1807e-2 (5.20e-4) −	3.2901e-2 (3.76e-3) −	4.4899e-2 (2.58e-4) −	3.2417e-2 (3.11e-3) −	2.0903e-2 (1.98e-4)
MaF4	5.8395e-1 (1.55e-3) −	2.5536e-1 (5.95e-3) −	3.2311e-1 (1.63e-2) −	3.8401e-1 (1.06e-1) −	4.5213e-1 (2.27e-1) −	2.3830e-1 (3.08e-3)
DTLZ2-10	4.5187e-1 (2.73e-2) −	4.2634e-1 (1.46e-2) −	4.4224e-1 (4.33e-2) −	4.2101e-1 (3.62e-4) −	4.1345e-1 (2.02e-3) −	4.0212e-1 (7.49e-3)
IDTLZ1-10	2.3840e-1 (6.38e-3) −	1.7218e-1 (2.55e-2) −	1.4086e-1 (3.51e-3) −	2.6238e-1 (4.61e-2) −	1.1330e-1 (1.42e-2) −	1.1279e-1 (1.86e-3)
$+/-/\approx$	3/15/0	2/14/2	3/14/1	2/15/1	0/17/1	

Table 5.2 presents the HV metric values obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on the test problems with different PF shapes. The proposed algorithm has achieved the best performance on 12 of 18 test instances, while the number of best results obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA are 1, 1, 2, 2 and 0, respectively. It can be observed that proposed AMAWV outperforms the other five compared algorithms a lot with respect to HV metric. Figs. 5.6 and 5.7 show the final non-dominated solution set with the median HV value obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on

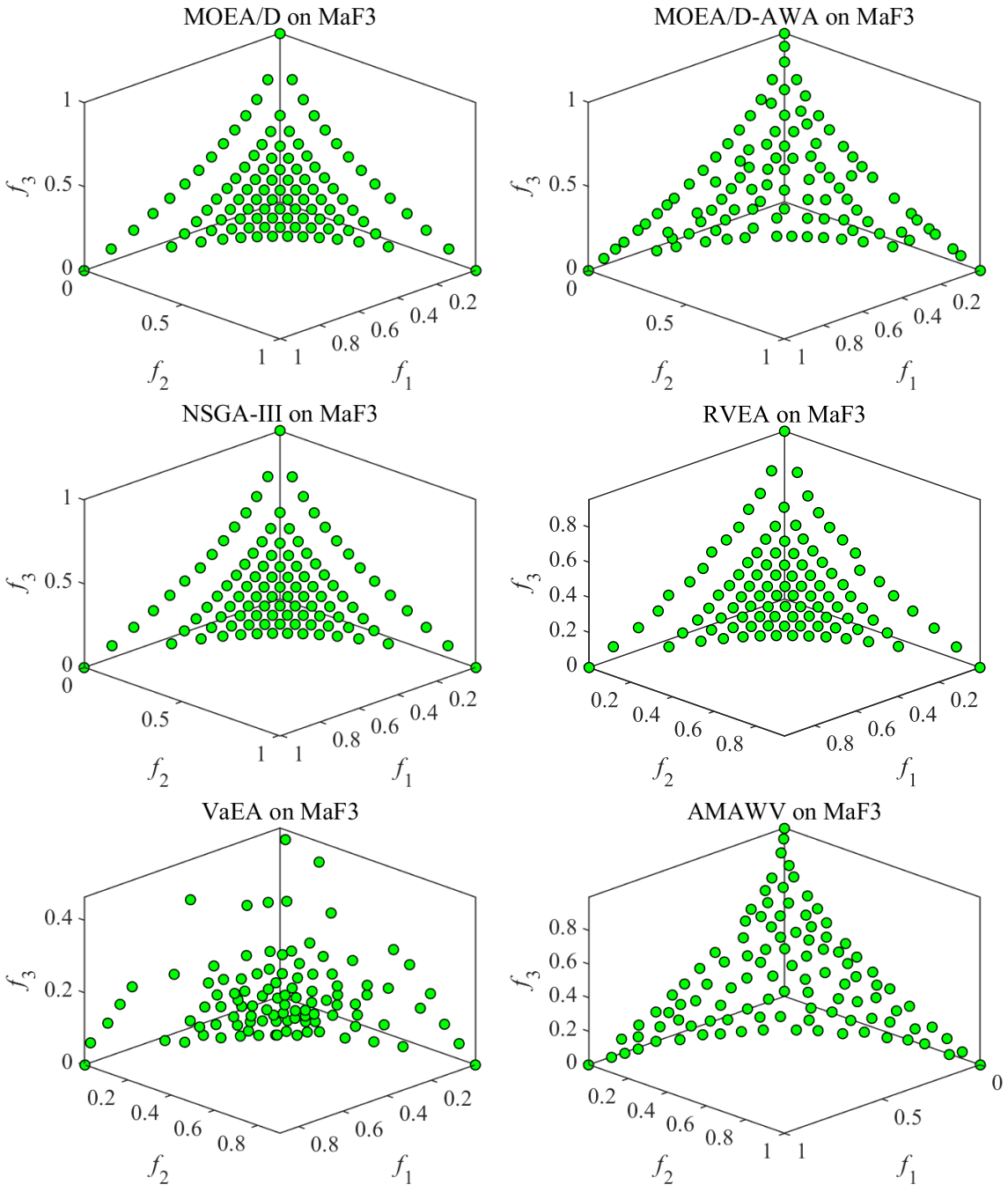


Figure 5.4: The non-dominated solution set with the median IGD value obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on 3-objective MaF3

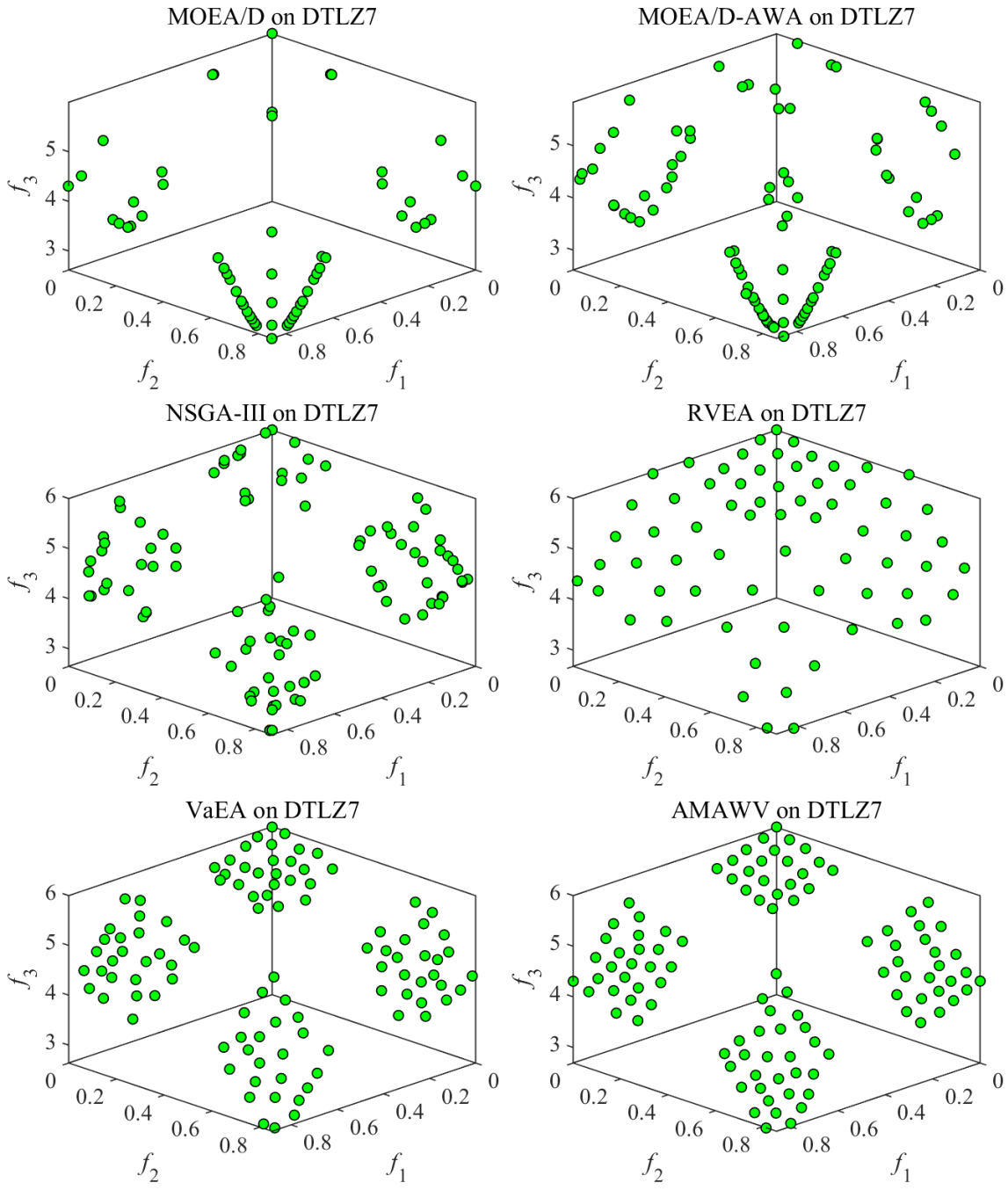


Figure 5.5: The non-dominated solution set with the median IGD value obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on 3-objective DTLZ7

3-objective MaF4 and 10-objective IDTLZ1. For 3-objective MaF4, it has an inverted badly scaled PF shape, MOEA/D could only find a part of PF, the solution set of RVEA seems to be sparse, AMAWV can obtain a good spread of the solutions compared with MOEA/D-AWA, NSGA-III and VaEA. For 10-objective IDTLZ1, it has a high-dimensional and inverted PF shape, RVEA cannot find the true PF, MOEA/D and MOEA/D-AWA could only find a small part of solutions, NSGA-III fails to cover the whole PF, VaEA has some solutions which fail to converge to the true PF. Only AMAWV can obtain a solution set to cover the whole PF with good convergence and diversity.

Table 5.2: The HV values obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on test problems

Problem	MOEA/D	MOEA/D-AWA	NSGA-III	RVEA	VaEA	AMAWV
DTLZ1	8.4428e-1 (1.30e-4) +	8.3738e-1 (7.26e-3) −	8.4420e-1 (3.31e-4) +	8.4431e-1 (1.33e-4) +	8.2025e-1 (1.92e-2) −	8.4286e-1 (4.05e-4)
DTLZ2	5.6302e-1 (4.11e-6) +	5.6491e-1 (3.92e-4) +	5.6302e-1 (6.40e-6) +	5.6301e-1 (1.99e-5) +	5.5825e-1 (1.37e-3) −	5.6162e-1 (9.39e-4)
CDTLZ2	9.5967e-1 (3.52e-6) −	9.6306e-1 (3.28e-4) −	9.5969e-1 (7.09e-5) −	9.6054e-1 (5.25e-4) −	9.5481e-1 (1.55e-3) −	9.6363e-1 (1.98e-4)
MaF3	9.5889e-1 (6.72e-4) −	9.6187e-1 (7.05e-4) −	9.5863e-1 (6.55e-4) −	9.6045e-1 (1.01e-3) −	9.1572e-1 (5.99e-2) −	9.6287e-1 (5.19e-4)
IDTLZ1	2.0345e-1 (1.01e-4) −	2.2342e-1 (3.22e-4) ≈	2.1113e-1 (8.86e-4) −	1.7968e-1 (1.62e-2) −	2.0726e-1 (1.74e-2) −	2.2353e-1 (3.10e-4)
IDTLZ2	5.0850e-1 (1.58e-5) −	5.3777e-1 (6.64e-4) −	5.2007e-1 (2.86e-3) −	5.1442e-1 (9.90e-4) −	5.3035e-1 (1.24e-3) −	5.3953e-1 (6.11e-4)
SCH1	8.5798e-1 (4.28e-6) −	8.5913e-1 (4.27e-5) ≈	8.5798e-1 (5.79e-6) −	8.5810e-1 (9.65e-6) −	8.5688e-1 (5.55e-4) −	8.5914e-1 (4.07e-5)
FON	4.3160e-1 (2.93e-5) +	4.3151e-1 (3.87e-5) ≈	4.3151e-1 (2.76e-5) ≈	4.2916e-1 (7.22e-4) −	4.2963e-1 (2.01e-4) −	4.3151e-1 (4.84e-5)
ZDT3	5.8139e-1 (1.29e-5) −	5.8320e-1 (2.61e-5) −	5.8217e-1 (3.33e-2) −	5.7843e-1 (1.11e-3) −	5.8321e-1 (4.34e-2) ≈	5.8330e-1 (5.12e-5)
DTLZ7	2.5390e-1 (1.82e-2) −	2.6177e-1 (1.12e-2) −	2.7184e-1 (1.63e-3) −	2.6020e-1 (2.47e-3) −	2.7779e-1 (7.25e-4) −	2.8080e-1 (2.75e-4)
DTLZ5	1.9268e-1 (2.21e-6) −	1.9963e-1 (9.10e-5) −	1.9466e-1 (1.05e-3) −	1.6334e-1 (2.50e-3) −	1.9967e-1 (1.24e-4) −	2.0016e-1 (1.87e-4)
DTLZ6	1.9268e-1 (1.49e-6) −	1.9966e-1 (8.29e-5) −	1.9231e-1 (1.66e-3) −	1.5706e-1 (7.70e-3) −	1.9993e-1 (8.08e-5) −	2.0025e-1 (3.85e-5)
SDTLZ1	6.8595e-1 (3.79e-4) −	7.5145e-1 (4.16e-2) −	8.4384e-1 (2.06e-3) +	8.4121e-1 (1.46e-2) −	8.2476e-1 (1.78e-2) −	8.4274e-1 (3.69e-4)
SDTLZ2	4.3510e-1 (1.29e-5) −	5.2712e-1 (1.06e-2) −	5.6302e-1 (6.68e-6) +	5.6256e-1 (2.49e-4) ≈	5.5817e-1 (1.43e-3) −	5.6239e-1 (5.72e-4)
SCH2	6.4666e-1 (5.20e-5) −	6.5528e-1 (4.61e-5) ≈	6.5458e-1 (2.12e-4) −	6.5214e-1 (9.42e-5) −	6.5499e-1 (3.46e-4) −	6.5529e-1 (1.15e-4)
MaF4	4.7320e-1 (1.08e-3) −	5.3039e-1 (1.87e-3) −	5.1973e-1 (4.55e-3) −	5.0494e-1 (2.65e-2) −	4.9773e-1 (5.77e-2) −	5.3759e-1 (8.18e-4)
DTLZ2-10	9.5643e-1 (1.30e-2) +	9.6452e-1 (6.90e-3) +	9.6119e-1 (1.73e-2) +	9.6979e-1 (2.08e-4) +	9.4757e-1 (3.02e-3) +	9.3416e-1 (1.67e-2)
IDTLZ1-10	6.4382-2 (2.88e-4) −	1.6567-1 (2.55e-2) −	9.5142e-2 (6.59e-3) −	3.6636e-2 (4.37e-3) −	3.3557e-1 (1.38e-2) ≈	3.3716e-1 (7.14e-3)
$+/-/\approx$	4/14/0	2/12/4	5/12/1	3/14/1	1/15/2	

5.4 Summary

In this chapter, we proposed a novel archive maintenance for adapting weight vectors to make the decomposition-based multi-objective evolutionary algorithms solve MOPs and MaOPs with different PF shapes. The novel archive maintenance strategy can delete some dominance resistant solutions to improve the convergence and the one by one adding solution procedure can retain good diversity. The weight vector adaptation method helps the decomposition-based algorithms to be suitable for different problems with various PF properties. The experimental results have demonstrated the superiority and versatility of the proposed algorithm.

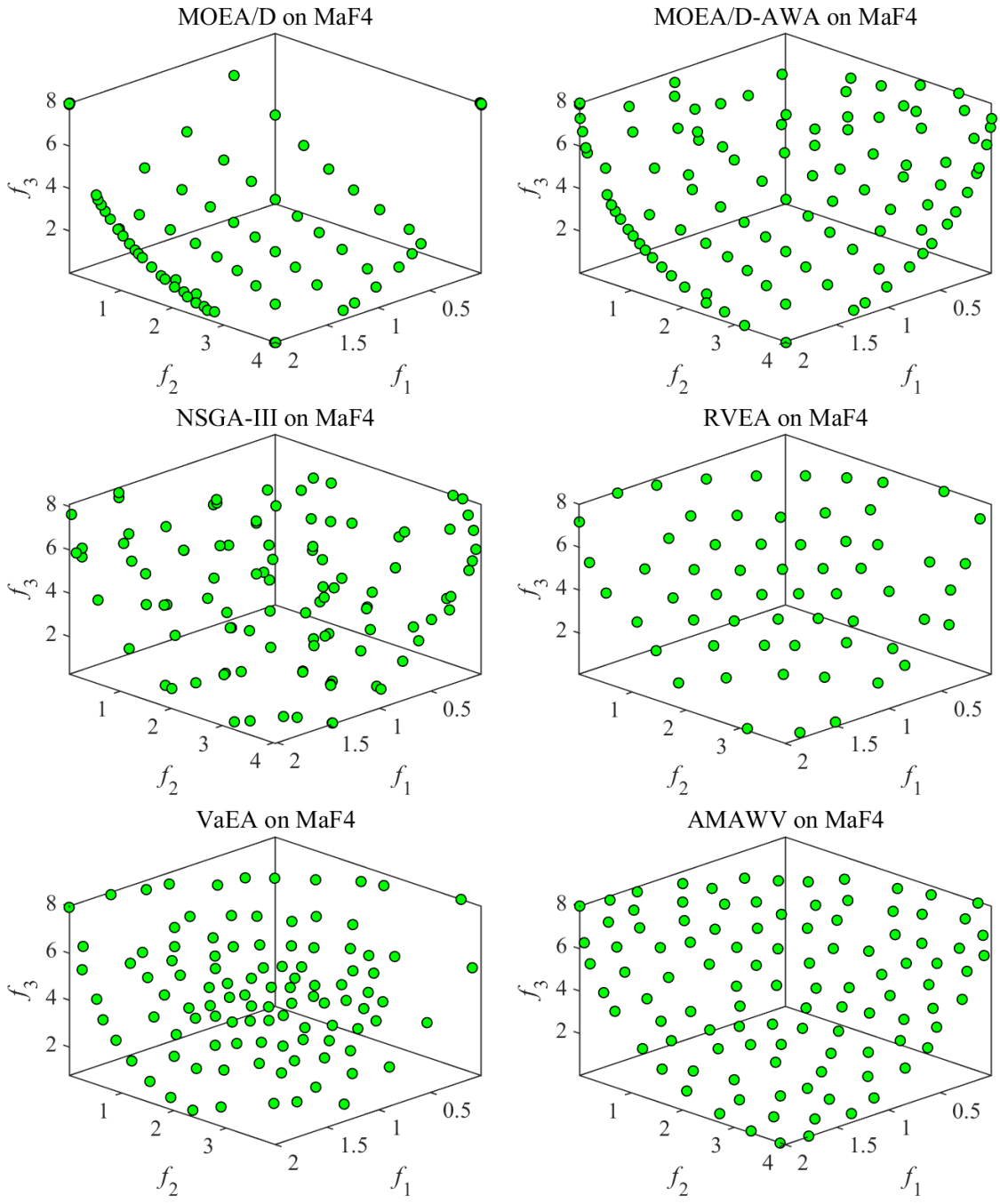


Figure 5.6: The non-dominated solution set with the median HV value obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on 3-objective MaF4

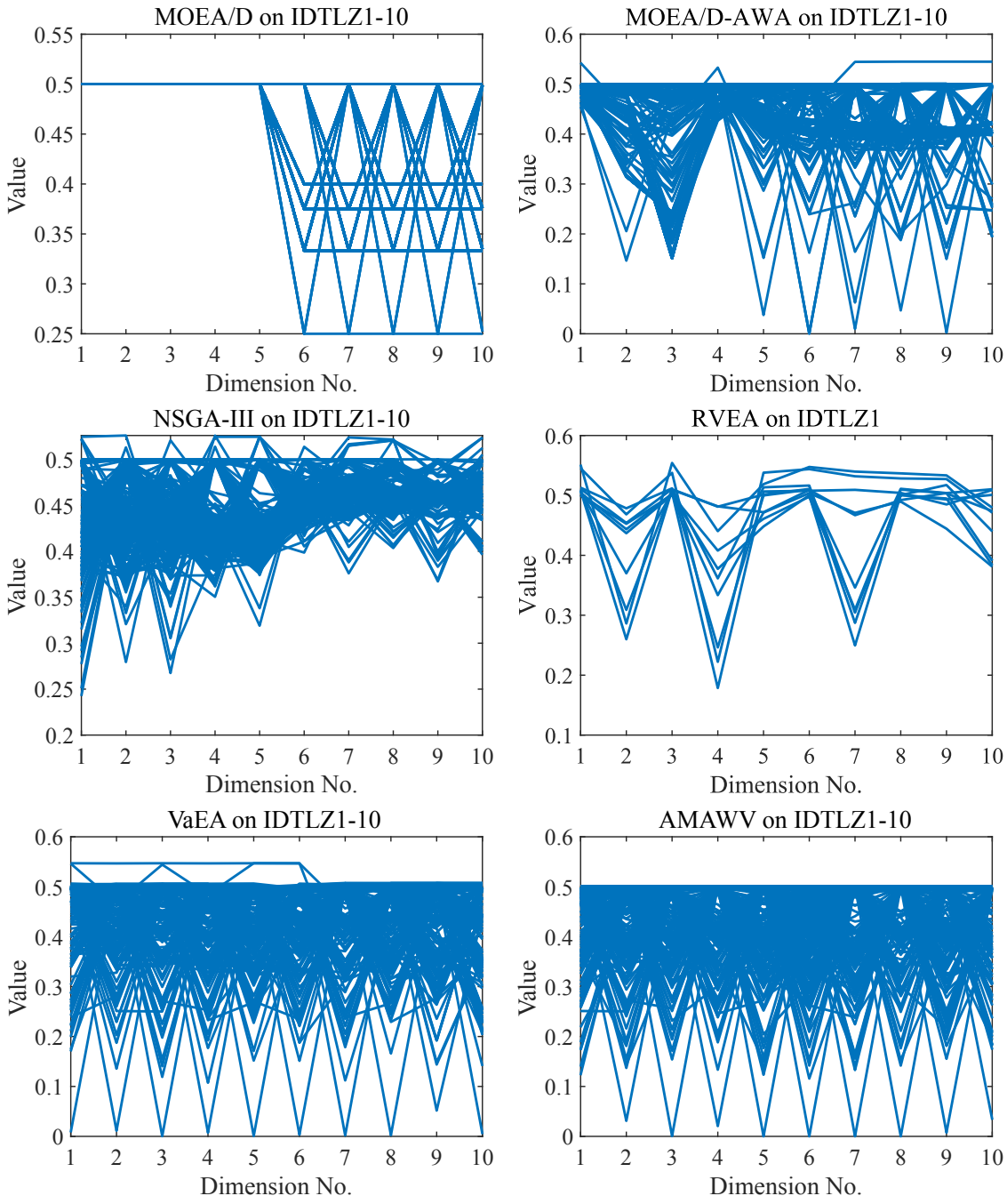


Figure 5.7: The non-dominated solution set with the median HV value obtained by MOEA/D, MOEA/D-AWA, NSGA-III, RVEA, VaEA and AMAWV on 10-objective IDTLZ1

Chapter 6

Three Constrained Algorithms with Better Versatility

This chapter presents three constrained multi-objective optimization algorithms (called PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE) to solve a variety of constrained multi-objective optimization problems (CMOPs). We combine the push and pull search (PPS) framework with the algorithms NSGA-II [24], SPEA2 [124], and SPEA2-SDE [62], so we name the proposed algorithms PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE.

Many real-world MOPs consist of different conflicting objectives with a set of inequality and/or equality constraints, referred to as CMOPs. CMOPs are challenging because of the difficulty in striking a good balance of optimizing objectives and satisfying constraints simultaneously. Some constrained MOEAs (CMOEAs) have achieved good performance on certain CMOPs, while they show poor versatility on complicated CMOPs, e.g., CMOPs with large infeasible regions, CMOPs with adjustable difficulties and CMOPs with both decision and objective constraints. To address this issue, in this chapter we propose three CMOEAs for solving different kinds of CMOPs.

Inspired by the push and pull search (PPS) framework, three CMOEAs are developed by combining the advantages of population-based search algorithms with flexible constraint handling mechanisms. In addition, three popular and challenging constrained benchmark suites are selected to test the performance of the proposed algorithms by comparing them to the other seven state-of-the-art CMOEAs. The experimental results demonstrate the effectiveness and superiority of the proposed algorithms.

6.1 PPS Framework

The push and pull search (PPS) framework was proposed to solve CMOPs by Fan et al. [33]. The search process of PPS is divided into two different stages: push and pull search stages. In the first push stage, the working population is pushed to approach the unconstrained Pareto front without considering any constraints, which can help the solutions to get across infeasible regions. Afterwards, a constraint handling mechanism is used to pull the working population to approach the constrained Pareto front in the pull stage.

The condition when to convert from the push stage to pull stage is important, which can be suggested as [33, 35]:

$$r_k = \max\{rz_k, rn_k\} \leq \varepsilon \tag{6.1}$$

where ε (suggested $\varepsilon = 0.001$) is a threshold. r_k denotes the maximum rate of change between the ideal and nadir points during the last l generations. The coordinates of the nadir point correspond to the maximum value of each objective on the true Pareto front. rz_k and rn_k represent the rates of change of the ideal and nadir points during the last l generations, defined as follows:

$$rz_k = \max_{i=1,\cdots,m} \left\{ \frac{\left| z_i^k - z_i^{k-l} \right|}{\max\left\{ \left| z_i^{k-l} \right|, \Delta \right\}} \right\} \tag{6.2}$$

$$rn_k = \max_{i=1,\cdots,m} \left\{ \frac{\left| n_i^k - n_i^{k-l} \right|}{\max\left\{ \left| n_i^{k-l} \right|, \Delta \right\}} \right\} \tag{6.3}$$

where $z^k = \left(z_1^k, \cdots, z_m^k\right)$ and $n^k = \left(n_1^k, \cdots, n_m^k\right)$ are the ideal and nadir points in the k-th generation, respectively. $z^{k-l} = \left(z_1^{k-l}, \cdots, z_m^{k-l}\right)$ and $n^{k-l} = \left(n_1^{k-l}, \cdots, n_m^{k-l}\right)$ are the ideal and nadir points in the $k-l$-th generation. Δ (suggested $\Delta = 1e-6$) is a very small positive number, which is used to make sure that the denominators in Equations 6.2 and 6.3 are not equal to zero. rz_k and rn_k are two points in the interval [0,1].

r_k is initialized to 1 at the beginning of the search, and is updated at each iteration according to Equation 6.1. When r_k is less than or equal to ε, the push stage will be transformed into pull stage.

To summarize, PPS has two potential advantages over other constraint handling techniques [35]. During the first push stage, a multi-objective evolutionary algorithm is adopted to approximate the Pareto front without considering any constraints, which can help the working population to get across the large infeasible regions and avoid the distance between the unconstrained PF and true PF. After obtaining the unconstrained PF in the push stage, some valuable information can be collected

to guide the parameter setting for the constraint handling approaches in the pull stage, which can enhance the adaptability of the algorithm.

6.2 The Proposed PPS-NSGA-II/SPEA2/SPEA2-SDE

This section presents the details of three proposed algorithms PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE.

6.2.1 General Framework

The flowchart of PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE is shown in Fig. 6.1. The general frameworks of the three proposed algorithms are presented in Algorithm 7 and Algorithm 8, respectively. Please note that PPS-SPEA2 and PPS-SPEA2-SDE share the same framework but have different fitness calculating methods. In the framework of PPS-NSGA-II, the non-dominated front numbers and crowding distances of solutions are calculated by the fast non-dominated sorting approach [24] with and without considering constraints, respectively. The whole search process consists of two stages: push and pull search. When $PushStage = true$, the push stage is utilized, the parents are selected via binary tournament selection as the mating pool without considering constraints and then offspring solutions O are generated from the mating pool. When $PushStage = false$, the pull stage is applied, a constraint handling mechanism is embedded into NSGA-II to pull the working population to the constrained PF. The parameter r_k for switching from push to pull stage is updated iteratively.

In the framework of PPS-SPEA2 and PPS-SPEA2-SDE, different fitness calculation methods are adopted instead of calculating the non-dominated front numbers and crowding distances in PPS-NSGA-II. The fitness calculating methods can reflect both the performance of convergence and diversity of each solution in the population. Without loss of generality, the smaller fitness value means better performance. The whole search processes of PPS-SPEA2 and PPS-SPEA2-SDE also include push and pull search stages. In the push stage, we use SPEA2 and SPEA2-SDE without considering any constraints to search the unconstrained PF. In the pull stage, the constraint handing approaches are applied to search the constrained PF. More details about the main operations in PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE are presented in Sections 6.2.2, 6.2.3, and 6.2.4, respectively.

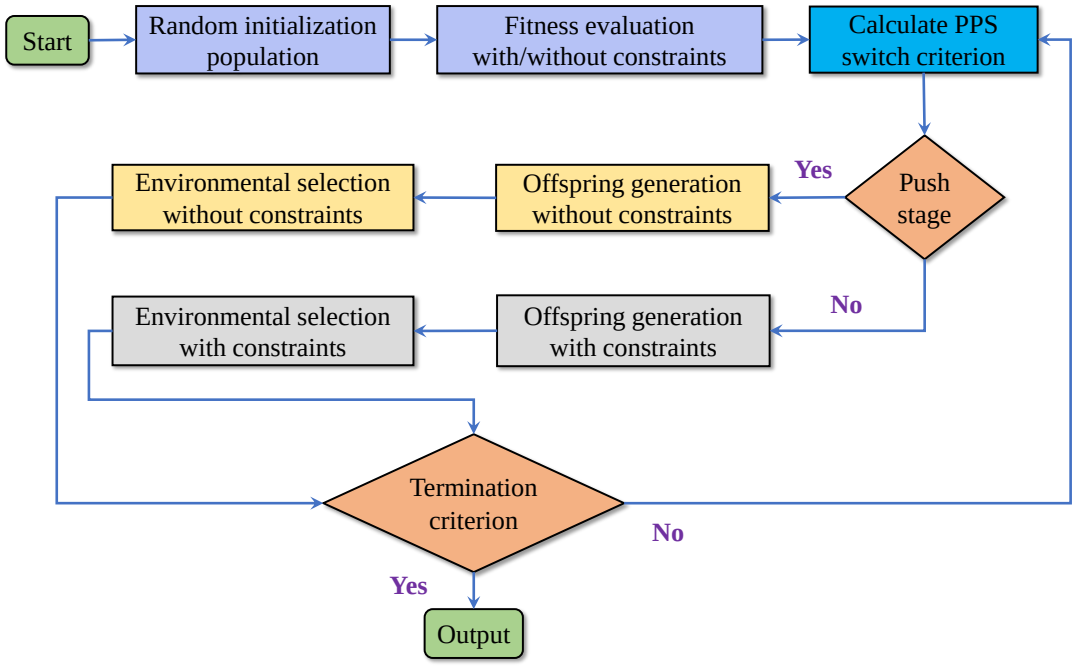


Figure 6.1: The flowchart of PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE

6.2.2 PPS-NSGA-II

In the original NSGA-II, Deb et al. [24] embedded feasibility into Pareto dominance and developed a constraint dominance principle (CDP) to deal with constraints. If a solution x^i is said to constrained-dominate a solution x^j, one of the following three conditions holds: 1) x^i is a feasible solution and x^j is an infeasible solution. 2) Solutions x^i and x^j are both feasible solutions, and solution x^i Pareto dominates solution x^j in terms of objectives. 3) Solutions x^i and x^j are both infeasible solutions, and solution x^i has a lower overall constraint violation than that of solution x^j.

PPS-NSGA-II is an instantiation of the PPS framework of a specific type of NSGA-II algorithm [24]. In the push search stage, we use an unconstrained NSGA-II to search for both feasible and infeasible solutions to minimize the objectives of solutions without considering any constraints, which aims to approach the unconstrained PF. The non-dominated front numbers and crowding distances of solutions are calculated by the fast non-dominated sorting approach. The crowding distance is defined as the average distance between its two closest points on each objective. Then $\widetilde{N}$ parents are selected as mating pool via binary tournament selection based on the non-dominated front numbers and crowding distances. The two parents are randomly selected from the mating pool

Algorithm 7: Framework of PPS-NSGA-II

Input: The population size $\widetilde{N}$
Output: The final population P

$P \leftarrow Initialization\left(\widetilde{N}\right)$;
$[F_1, F_2, \cdots] \leftarrow NDSorting\left(P.objs\right)$;
$CrowdDis \leftarrow CrowdingDistance\left(F_1, F_2, \cdots\right)$;
$[F_1{}', F_2{}', \cdots] \leftarrow NDSorting\left(P.objs, P.cons\right)$;
$CrowdDis' \leftarrow CrowdingDistance\left(F_1{}', F_2{}', \cdots\right)$;
Set $r_k = 1.0$, $PushStage = true$;
while $termination\ criterion\ not\ fulfilled$ **do**
    Calculate r_k according to Equation 6.1;
    if $r_k \leq \varepsilon$ *and* $PushStage = true$ **then**
        $PushStage = false$;
    end
    if $PushStage = true$ **then**
        $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $[F_1, F_2, \cdots]$ and $CrowdDis$ in P;
        $O \leftarrow OffspringGeneration\left(P, P'\right)$;
        $\left(P, [F_1, F_2, \cdots], CrowdDis\right) \leftarrow EnvironemntalSelection\left(P \cup O\right)$;
    else
        $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $[F_1{}', F_2{}', \cdots]$ and $CrowdDis'$ in P;
        $O \leftarrow OffspringGeneration\left(P, P'\right)$;
        $\left(P, [F_1{}', F_2{}', \cdots], CrowdDis'\right) \leftarrow EnvironemntalSelection'\left(P \cup O\right)$;
    end
end

to generate two offspring solutions, and a genetic operator [24] or differential evolution operator [58] can be applied as offspring generating operator. Thus the environmental selection operation is adopted to update the non-dominated front numbers and crowding distances as well as the new population.

The ideal and nadir points are updated at each iteration. And the maximum rate of change between the ideal and nadir points (r_k) during the last l generations is calculated. When r_k satisfies the condition of switching from the push to pull stages, the pull search stage is starting. In the pull search stage, the constraint dominance principle (CDP) is applied to calculate the non-dominated front numbers and crowding distances. Then the new mating pool and offspring solutions are generated based on the new non-dominated front numbers and crowding distances with respect to the

Algorithm 8: Frameworks of PPS-SPEA2 and PPS-SPEA2-SDE

Input: The population size $\widetilde{N}$
Output: The final population P

$P \leftarrow Initialization\left(\widetilde{N}\right)$;
$Fitness \leftarrow CalFitness\left(P.objs\right)$;
$Fitness' \leftarrow CalFitness\left(P.objs, P.cons\right)$;
Set $r_k = 1.0$, $PushStage = true$;
while $termination\ criterion\ not\ fulfilled$ **do**
    Calculate r_k according to Equation 6.1;
    if $r_k \leq \varepsilon$ *and* $PushStage = true$ **then**
        $PushStage = false$;
    end
    if $PushStage = true$ **then**
        $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $Fitness$ in P;
        $O \leftarrow OffspringGeneration\left(P, P'\right)$;
        $\left(P, Fitness\right) \leftarrow EnvironemntalSelection\left(P \cup O\right)$;
    else
        $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection according to $Fitness'$ in P;
        $O \leftarrow OffspringGeneration\left(P, P'\right)$;
        $\left(P, Fitness'\right) \leftarrow EnvironemntalSelection'\left(P \cup O\right)$;
    end
end

constraints. Finally, a set of feasible solutions will be updated and obtained in the environmental selection operation.

6.2.3 PPS-SPEA2

PPS-SPEA2 is an instantiation of the PPS framework of a specific type of SPEA2 algorithm [124]. In PPS-NSGA-II, the non-dominated front number represents the performance of convergence and the crowding distance reflects the performance of diversity. However, the fitness metric value is used to measure both convergence and diversity in PPS-SPEA2. The fitness evaluation strategy shares the same idea to the one in the original SPEA2. First of all, let the solution set R_x store all the solutions dominated by x and the solution set S_x store all the solutions dominating x, the raw fitness $R\left(x\right)$ of a solution x is calculated as:

$$R\left(x\right) = \sum_{y \in S_x} \left|R_y\right| \tag{6.4}$$

where $|R_y|$ denotes the number of solutions in the set. $R(x) = 0$ means solution x is a non-dominated solution. What's more, additional density information is needed to distinguish the quality of different non-dominated solutions. The k-th nearest neighbor method [81] is applied to measure the density information of solutions. Then $\left\lfloor \sqrt{2\widetilde{N}} \right\rfloor$-th nearest neighbor x' of solution x is detected, the density $D(x)$ corresponding to x is calculated as:

$$D(x) = \frac{1}{dist(x, x') + 2} \tag{6.5}$$

where $dist(x, x')$ denotes the Euclidean distance between solutions x and x'.

Hence, the fitness of the solution x can be expressed as follows:

$$fit(x) = R(x) + D(x) \tag{6.6}$$

where x is the non-dominated solution when $fit(x) < 1$. Obviously, smaller fitness means better quality of the solution.

PPS-SPEA2 also has two search stages: push and pull stages. In the push stage, no constraints will be considered into the fitness evaluation method, PPS-SPEA2 can search for unconstrained solutions. In the pull stage, the constraint dominance principle (CDP) is embedded into the fitness evaluation method, PPS-SPEA2 can pull the unconstrained solutions to the feasible regions. It is necessary to point out that the solution which has the minimum distance to another solution is chosen to be deleted in the environmental selection operation. If there are several solutions having the same minimum distance, we consider the second smallest distances and so forth.

6.2.4 PPS-SPEA2-SDE

PPS-SPEA2-SDE is an instantiation of the PPS framework of a specific type of SPEA2-SDE algorithm [62]. Compared PPS-SPEA2 with PPS-SPEA2-SDE, the fitness calculating method is different. In PPS-SPEA2-SDE, the shift-based density estimation (SDE) strategy is used to measure the density of the solutions. The shifted-based density estimation based distance between solution x and solution y ($y \in P \backslash \{x\}$) can be calculated as:

$$SDE(x, y) = \sqrt{\sum_{i=1}^{m} \left(\max\{0, f_i(y) - f_i(x)\}\right)^2} \tag{6.7}$$

Similar to PPS-SPEA2, the fitness of the solution x can be expressed as follows:

$$fit\left(x\right) = R\left(x\right) + \frac{1}{SDE\left(x, x'\right) + 2} \tag{6.8}$$

where $R\left(x\right)$ is the same to that of PPS-SPEA2. $SDE\left(x, x'\right)$ is the SDE crowding degree of the solution x with regard to its $\left\lfloor \sqrt{2\widetilde{N}} \right\rfloor$-th nearest neighbor x'. Afterwards, PPS-SPEA2-SDE shares the same search process with PPS-SPEA2. It is noted that the solution which has the minimum SDE based distance to another solution is chosen to be deleted in the environmental selection operation. If there are several solutions having the same minimum SDE based distance, we consider the second smallest distances and so forth.

6.2.5 Computational Complexity

For the proposed algorithms PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE, the major costs are the iteration process in Algorithm 7 and Algorithm 8. In PPS-NSGA-II, the worst-case time complexities of the maximum rate of change between the ideal and nadir points are $O(\widetilde{M}\widetilde{N})$, where $\widetilde{M}$ is the number of objectives and $\widetilde{N}$ is the population size. The mating selection operator needs $O(\widetilde{N})$ operations for the binary tournament selection. The offspring reproduction needs $O(\widetilde{N}D)$ operations to generate offspring solutions, where D is the number of decision variables. The non-dominated sorting operator and environmental selection operator need $O(\widetilde{M}\widetilde{N}^2)$ operations. Thus, the overall computational complexity of PPS-NSGA-II within one generation is $O(\widetilde{M}\widetilde{N}^2)$. The computational complexity of original NSGA-II is also $O(\widetilde{M}\widetilde{N}^2)$ [24]. We can see that the proposed PPS-NSGA-II has the same computational complexity as the original NSGA-II. However, the proposed PPS-NSGA-II has two search stages, the first stage without considering any constraints and the second stage considering the constraints. The original constrained NSGA-II only has the second search stage. Hence, the proposed PPS-NSGA-II needs more calculation for the first stage search.

In PPS-SPEA2, the time complexity of the fitness calculating procedure is $O(\widetilde{N}^2 \log \widetilde{N})$. The binary tournament selection needs $O(\widetilde{N})$ operations and the offspring generation needs $O(\widetilde{N}D)$ operations. The worst run-time complexity of the environmental selection operator is $O(\widetilde{N}^3)$, on average the complexity will be lower $O(\widetilde{N}^2 \log \widetilde{N})$ [124]. Thus, the worst overall computational complexity of PPS-SPEA2 within one generation is $O(\widetilde{N}^3)$. In PPS-SPEA2-SDE, the time complexity of the fitness calculating procedure is $O(\widetilde{M}\widetilde{N}^2)$. And the worst run-time complexity of the environmental selection operator is $O(\widetilde{N}^3)$. Since $\widetilde{N}$ is often larger than $\widetilde{M}$. Hence, the worst overall computational complexity of PPS-SPEA2-SDE within one generation is $O(\widetilde{N}^3)$.

6.3 Simulations on Benchmark Problems

To test the performance of the proposed algorithms PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE, we select seven algorithms for comparison, namely TiGE-2 [117], constrained NSGA-II [24], C-TAEA [59], PPS-MOEA/D [33], ToP [66], CCMO [89], and CMOEA-MS [91] to solve the LIR-CMOP test suite [32], DAS-CMOP test suite [34], and DOC test suite [66].

6.3.1 Parameter Settings

Benchmark Suites

A series of CMOPs are selected from three challenging benchmark suites LIR-CMOP, DAS-CMOP, and DOC test suites, which are used as test problems. LIR-CMOP test problems have large infeasible regions with different shapes. DAS-CMOP test problems' difficulties can be adjustable and the number of objectives can be scalable. DOC test problems consider both decision and objective constraints at the same time. These selected CMOPs have different characteristics which can pose different difficulties for the CMOEAs. The number of objectives in LIR-CMOP13, LIR-CMOP14, DASCMOP7, DAS-CMOP8, DAS-CMOP9, DOC8 and DOC9 is 3, the number of objectives in the other CMOPs is 2. And the number of decision variables in all CMOPs is set to 10 [91].

Compared Algorithms

Seven popular CMOEAs (TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, and CMOEA-MS) are selected as compared algorithms. TiGE-2 utilizes three different indicators to measure convergence, diversity, and feasibility. NSGA-II adopts constraint dominance principle (CDP) to deal with constraints. C-TAEA maintains the convergence-oriented archive and the diversity-oriented archive simultaneously for constrained multi-objective optimization. PPS-MOEA/D embedded the MOEA/D algorithm into the PPS framework, and an improved epsilon constraint handling is applied into MOEA/D. A simple and efficient two-phase framework is implemented in ToP algorithm. In the first phase, a CMOP is transformed into a single-objective optimization problem to find the potentially feasible regions. In the second phase, one CMOEA is applicable to get the final solutions. CCMO used a coevolutionary framework and was assisted by a simple helper problem for solving CMOPs. CMOEA-MS adjusted the fitness evaluation strategies to balance objective optimization and constraint satisfaction in constrained evolutionary multi-objective optimization.

For the above algorithms, TiGE-2, NSGA-II, C-TAEA, CCMO and CMOEA-MS used simulated

binary crossover [20] and polynomial mutation [21] to generate offspring solutions, while PPS-MOEA/D, ToP, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE adopt differential evolution [58] and polynomial mutation to generate offspring solutions. The simulation crossover probability and polynomial mutation probability are set to 1.0 and $1/D$, respectively. The distribution index of crossover and mutation is set to 20. The differential evolution control parameters CR and F are set to 1 and 0.5, respectively. All the other parameters in these algorithms are set as suggested in their original parameters. For a fair comparison of different CMOEAs, the population size is set to 100 for all the test problems with two objectives and 300 for the problems with three objectives. The total number of function evaluations (FEs) is set to 200,000 for all two-objective problems, and to 400,000 for three-objective CMOPs.

Performance Metrics

In order to evaluate the performance of compared CMOEAs, the inverted generational distance (IGD) [9] and hypervolume (HV) [96] metrics are adopted. IGD and HV can reflect the information of both convergence and diversity. The smaller IGD and larger HV mean better.

Each algorithm is executed 30 times independently on each test problem, and the average and standard deviation of performance metric values are recorded. The Wilcoxon rank sum test at a 5% significance level is used to compare the experimental results, where the symbol '$+$', '$-$' and '$\approx$' denotes that the result of another algorithm is significantly better, significantly worse and similar to that obtained by the proposed algorithm.

6.3.2 Simulation Results

Comparisons on LIR-CMOP Suite

The LIR-CMOP suite includes 14 challenging test problems, which have very large infeasible regions. These test problems have different unconstrained and constrained Pareto fronts, where the large infeasible regions may obstruct the solutions to approximate the true Pareto fronts. It can be observed from Table 6.1 that PPS-MOEA/D has achieved the best performance on five test instances, PPS-SPEA2-SDE exhibits the best performance on four test problems, PPS-SPEA2 gets one best results, while both PPS-CCMO and CMOEA-MS gain the best results on two CMOPs, respectively. According to the Wilcoxon rank sum test analysis, PPS-MOEA/D and PPS-NSGA-II have achieved better performance than PPS-SPEA2-SDE on LIR-CMOP suite. PPS-SPEA2 and PPS-SPEA2 have similar performance, which also gain better performance than CCMO and CMOEA-MS ad outperforms the other four algorithms (TiGE-2, NSGA-II, C-TAEA and ToP).

As shown in Figs. 6.2 and 6.3, TiGE-2 cannot find the true Pareto front of LIR-CMOP7. NSGA-II, C-TAEA and CMOEA-MS find some solutions that fail to jump over the large infeasible regions. PPS-MOEA/D, To/P and PPS-SPEA2 get some solutions that cannot converge to the true PF. CCMO, PPS-NSGA-II and PPS-SPEA2-SDE can gain good performance with regard to convergence and diversity. The PPS framework helps NSGA-II and SPEA2-SDE cross the large infeasible solutions and obtain populations distributed uniformly on the PF.

Table 6.1: The IGD values obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on the LIR-CMOP benchmark suite. The best result in each row is highlighted.

Problem	TiGE-2	NSGA-II	C-TAEA	PPS-MOEA/D	ToP
LIR-CMOP1	9.3392e-2 (3.23e-2) −	2.1920e-1 (8.39e-2) −	2.9600e-1 (1.54e-1) −	1.0552e-2 (5.51e-3) +	1.3882e-1 (1.06e-1) −
LIR-CMOP2	1.0701e-1 (3.21e-2) −	1.9173e-1 (6.41e-2) −	7.7720e-2 (2.32e-2) −	6.3287e-3 (7.28e-4) +	1.3271e-1 (1.15e-1) −
LIR-CMOP3	9.2810e-2 (3.19e-2) −	2.4396e-1 (5.70e-2) −	3.2290e-1 (1.81e-1) −	1.1475e-2 (1.42e-2) ≈	3.5708e-1 (7.76e-2) −
LIR-CMOP4	1.0274e-1 (2.14e-2) −	2.5887e-1 (6.96e-2) −	1.7434e-1 (8.94e-2) −	4.6253e-3 (2.68e-3) ≈	3.0743e-1 (5.65e-2) −
LIR-CMOP5	3.2555e-1 (7.96e-2) −	5.8024e-1 (5.14e-1) −	9.3841e-2 (2.21e-2) −	6.5300e-3 (6.11e-4) −	1.9760e-1 (4.34e-1) −
LIR-CMOP6	4.2489e-1 (1.72e-1) −	5.3830e-1 (5.17e-1) −	1.0403e-1 (8.33e-2) −	7.9585e-3 (1.21e-3) −	3.7106e-2 (9.32e-2) ≈
LIR-CMOP7	1.5879e-1 (8.60e-2) −	2.1200e-2 (2.33e-2) −	2.0310e-2 (5.78e-3) −	1.0805e-2 (1.35e-3) −	8.6365e-3 (3.09e-4) −
LIR-CMOP8	2.7260e-1 (1.27e-1) −	2.8755e-2 (5.32e-2) −	1.8209e-2 (7.29e-3) −	1.0406e-2 (1.11e-3) −	8.7276e-3 (4.08e-4) −
LIR-CMOP9	7.7393e-1 (2.11e-1) −	5.2245e-1 (1.30e-1) −	7.2773e-2 (3.05e-2) ≈	3.2189e-3 (1.27e-4) +	3.2045e-1 (1.27e-1) ≈
LIR-CMOP10	4.7189e-1 (1.88e-2) −	3.2749e-1 (9.87e-2) −	5.3351e-2 (4.92e-2) −	5.2406e-3 (2.96e-4) −	5.5419e-3 (2.14e-4) −
LIR-CMOP11	6.7401e-1 (3.84e-1) −	2.0368e-1 (1.81e-1) −	1.2915e-1 (4.30e-2) −	2.4144e-3 (1.00e-4) +	1.3398e-1 (7.04e-2) −
LIR-CMOP12	4.0064e-1 (2.12e-1) −	1.5019e-1 (9.19e-2) −	1.8690e-2 (7.04e-3) −	3.1229e-3 (1.09e-4) ≈	3.4787e-2 (5.13e-2) −
LIR-CMOP13	3.7022e-1 (6.73e-2) −	6.7847e-2 (1.55e-3) +	5.4180e-2 (6.66e-4) +	6.8434e-2 (1.82e-3) +	7.7436e-2 (1.94e-3) +
LIR-CMOP14	3.3733e-1 (7.20e-2) −	7.0452e-2 (2.09e-3) +	5.5202e-2 (5.93e-4) +	6.7415e-2 (1.30e-3) +	7.0416e-2 (1.06e-3) +
+/ − / ≈	0/14/0	2/12/0	2/11/1	6/5/3	2/10/2
Problem	CCMO	CMOEA-MS	PPS-NSGA-II	PPS-SPEA2	PPS-SPEA2-SDE
LIR-CMOP1	1.9523e-1 (1.14e-1) −	3.6404e-1 (1.19e-1) −	5.8615e-2 (2.71e-2) +	6.6652e-2 (2.60e-2) ≈	7.1902e-2 (2.67e-2)
LIR-CMOP2	7.4052e-2 (3.90e-2) −	2.4700e-1 (9.92e-2) −	1.4110e-2 (2.37e-2) +	1.0965e-2 (1.62e-2) +	1.8512e-2 (3.61e-2)
LIR-CMOP3	1.4087e-1 (5.75e-2) −	3.7948e-1 (1.68e-1) −	6.1746e-2 (5.90e-2) ≈	7.0369e-2 (6.78e-2) ≈	6.3190e-2 (5.96e-2)
LIR-CMOP4	1.4888e-1 (5.81e-2) −	3.1472e-1 (6.22e-2) −	3.7573e-2 (3.84e-2) ≈	5.8888e-2 (5.68e-2) ≈	4.5838e-2 (6.41e-2)
LIR-CMOP5	6.6559e-3 (8.50e-4) −	1.4538e-2 (1.46e-2) −	6.4025e-3 (3.30e-4) −	4.8944e-3 (1.09e-4) −	4.3859e-3 (1.25e-4)
LIR-CMOP6	5.9696e-3 (2.55e-4) +	1.6741e-2 (3.60e-2) ≈	6.4054e-3 (2.32e-4) ≈	5.0147e-3 (1.66e-4) +	6.6161e-3 (7.34e-4)
LIR-CMOP7	7.4989e-3 (5.28e-4) −	9.9978e-3 (1.13e-2) −	8.6728e-3 (3.27e-4) −	7.1968e-3 (1.90e-4) −	7.0463e-3 (1.72e-4)
LIR-CMOP8	7.1835e-3 (2.26e-4) ≈	1.5135e-2 (2.39e-2) ≈	8.7137e-3 (3.91e-4) −	7.1960e-3 (2.43e-4) ≈	7.0915e-3 (1.94e-4)
LIR-CMOP9	5.2432e-3 (2.01e-3) +	2.7109e-1 (1.20e-1) −	1.4441e-1 (2.24e-1) +	2.0447e-1 (2.62e-1) −	1.9521e-1 (2.79e-1)
LIR-CMOP10	4.6869e-3 (1.75e-4) −	6.5069e-2 (4.36e-2) −	5.5581e-3 (2.42e-4) −	4.3195e-3 (1.69e-4) −	3.7496e-3 (1.06e-4)
LIR-CMOP11	2.3872e-3 (4.42e-5) +	7.5616e-2 (6.43e-2) −	7.1308e-3 (2.58e-2) −	1.4166e-2 (3.74e-2) −	4.6128e-3 (1.28e-2)
LIR-CMOP12	3.0359e-3 (1.02e-4) ≈	3.7814e-2 (5.77e-2) ≈	3.4399e-3 (1.69e-3) ≈	2.3794e-2 (4.68e-2) ≈	1.6181e-2 (3.75e-2)
LIR-CMOP13	5.2697e-2 (3.89e-4) +	5.2124e-2 (2.84e-4) +	7.8700e-2 (1.57e-3) +	6.1769e-2 (8.63e-4) +	8.0243e-2 (2.34e-3)
LIR-CMOP14	5.4921e-2 (4.31e-4) +	5.4283e-2 (3.19e-4) +	7.0282e-2 (1.42e-3) +	5.8687e-2 (4.90e-4) +	8.2779e-2 (2.18e-3)
+/ − / ≈	5/7/2	2/9/3	5/5/4	4/5/5	

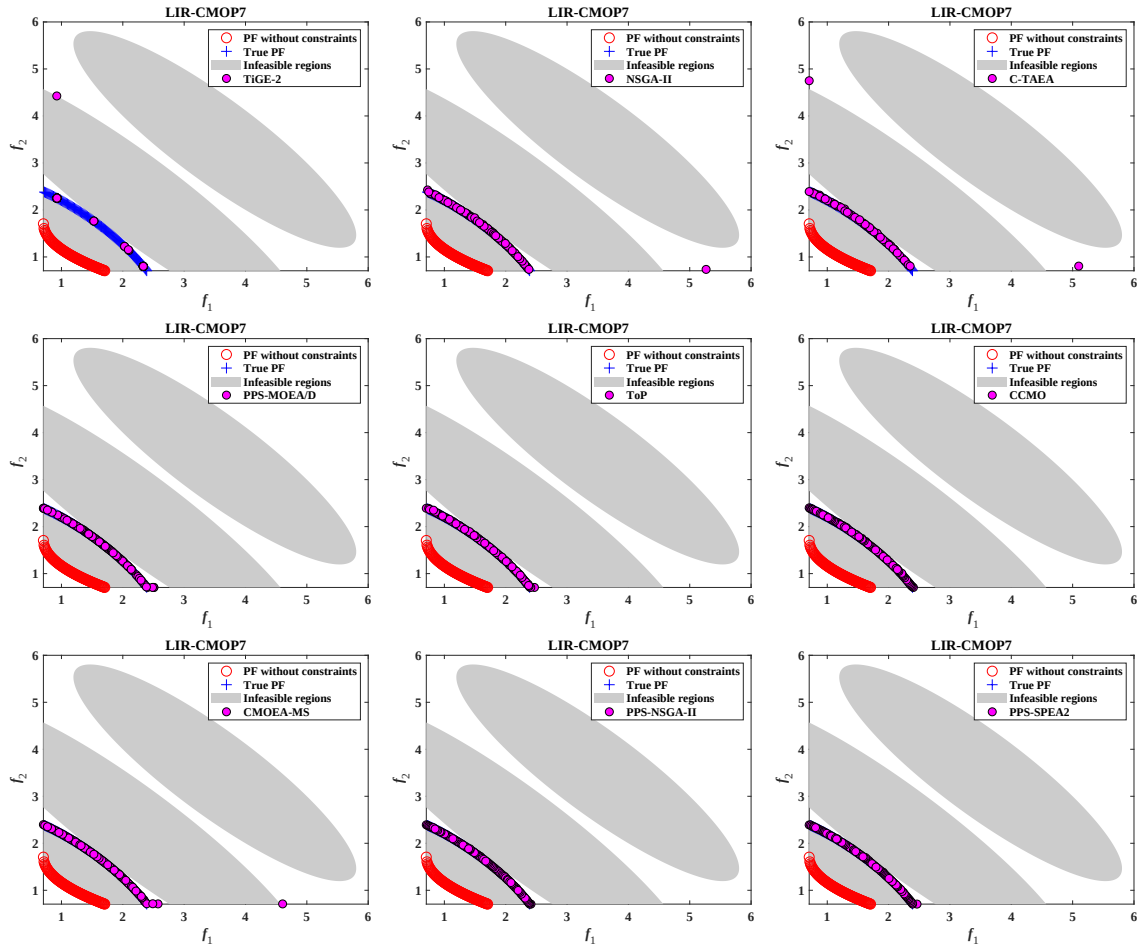


Figure 6.2: The non-dominated solution set with the medium IGD value obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, and PPS-SPEA2 on LIR-CMOP7

Comparisons on DAS-CMOP Suite

The DAS-CMOP suite consists of 9 difficulty-adjustable and scalable CMOPs. These test problems also have large infeasible regions, and the feasible regions are far away from the constrained Pareto fronts. As listed in Table 6.2, CCMO has achieved the best results on three CMOPs and followed by CMOEA-MS, PPS-MOEA/D, PPS-NSGA-II and PPS-SPEA2-SDE. It is necessary to point out that CCMO and PPS-NSGA-II have an obvious advantage over other algorithms on the DAS-CMOP test suite based on the Wilcoxon rank sum test. C-TAEA, PPS-MOEA/D, CMOEA-MS, PPS-SPEA2 and PPS-SPEA2-SDE have similar overall performance and outperforms the other

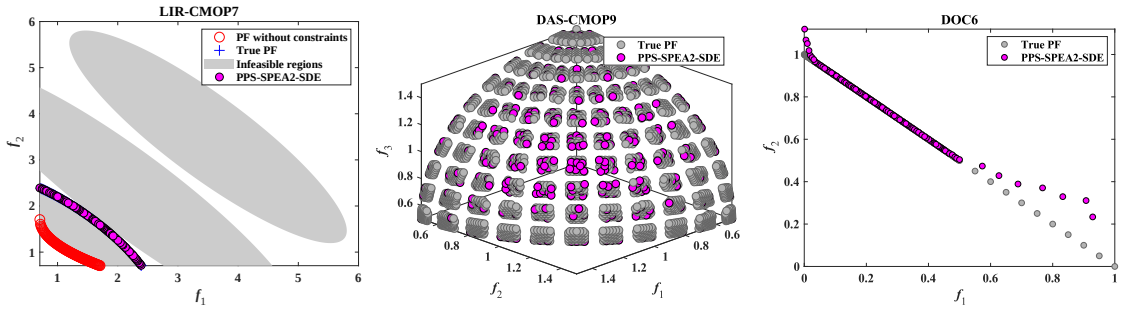


Figure 6.3: The non-dominated solution set with the medium IGD value obtained by PPS-SPEA2-SDE on LIR-CMOP7, DAS-CMOP9, and DOC6

three algorithms (TiGe-2, NSGA-II, and ToP).

Fig. 6.4 shows the final non-dominated solution set with the medium IGD value obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, and PPS-SPEA2 on DAS-CMOP9. And the non-dominated and feasible solutions obtained by PPS-SPEA2-SDE on DAS-CMOP9 is presented in Fig. 6.3. Obviously, TiGE-2 can only find some sparse solutions. NSGA-II, CCMO, CMOEA-MS, and PPS-SPEA2-SDE can find a set of solutions distributed in the center part of the Pareto front. C-TAEA and PPS-MOEA/D can find some solutions with better diversity performance since there exists a special diversity archive in CCMO for maintaining the diversity of solutions. ToP, PPS-NSGA-II and PPS-SPEA2 can get a large number of uniformly distributed solutions.

Comparisons on DOC Suite

The DOC suite contains 9 CMOPs with joint decision constraints and objective constraints. These test problems have various decision constraints (inequality, equality, linear and nonlinear constraints), which makes the feasible regions in the decision space have diverse characteristics. From Table 6.3, it can be found that PPS-SPEA2 gains the best results on four CMOPs, while Top and PPS-SPEA2-SDE have achieved the best performance on two CMOPs, respectively. As can be seen, PPS-NSGA-II and PPS-SPEA2 exhibit better overall performance than the other eight algorithms on the DOC suite. The results demonstrate that PPS-NSGA-II and PPS-SPEA2 can deal with more complicated CMOPs since the DOC suite involves complex constraints in both decision and objective spaces.

Fig. 6.5 plots the final non-dominated solution set with the medium IGD value obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, and PPS-SPEA2

Table 6.2: The IGD values obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on the DAS-CMOP benchmark suite. The best result in each row is highlighted.

Problem	TiGE-2	NSGA-II	C-TAEA	PPS-MOEA/D	ToP
DAS-CMOP1	1.0360e-1 (9.57e-2) −	1.3806e-1 (1.98e-1) −	1.0428e-2 (2.32e-3) +	2.3338e-3 (5.83e-4) +	1.2084e-2 (4.92e-2) −
DAS-CMOP2	2.5659e-2 (5.26e-3) −	3.9072e-2 (2.18e-2) −	7.6687e-3 (5.66e-4) −	3.9502e-3 (1.11e-4) −	3.8178e-3 (1.22e-4) −
DAS-CMOP3	1.2183e-1 (4.55e-2) −	1.7979e-1 (6.74e-2) −	3.2096e-2 (1.11e-2) +	1.8648e-2 (2.09e-3) +	1.2209e-1 (1.03e-1) −
DAS-CMOP4	1.3661e-1 (2.22e-1) −	4.7427e-2 (1.29e-1) −	1.1204e-2 (1.71e-3) −	1.7091e-3 (7.87e-5) +	5.3641e-1 (1.74e-1) −
DAS-CMOP5	2.4390e-2 (4.84e-3) −	3.4839e-3 (1.14e-4) −	7.8244e-3 (5.68e-4) −	4.0922e-3 (1.49e-4) −	5.4637e-1 (2.45e-1) −
DAS-CMOP6	1.0086e-1 (1.12e-1) −	4.1587e-2 (7.76e-2) −	2.6768e-2 (4.49e-3) −	2.1863e-2 (6.10e-3) ≈	7.0519e-1 (1.16e-1) −
DAS-CMOP7	5.4695e-2 (3.56e-3) −	2.3389e-2 (1.01e-3) +	2.9083e-2 (8.55e-4) +	3.9192e-2 (4.63e-3) ≈	5.2238e-1 (2.22e-1) −
DAS-CMOP8	6.8960e-2 (3.04e-3) −	2.8454e-2 (1.24e-3) +	4.4571e-2 (2.18e-3) +	6.0822e-2 (1.25e-2) ≈	7.5145e-1 (1.90e-1) −
DAS-CMOP9	6.1361e-2 (2.91e-3) −	3.1214e-2 (1.24e-3) −	4.5581e-2 (2.70e-3) −	3.2068e-2 (2.00e-3) −	2.8497e-2 (9.37e-4) +
+/ − / ≈	0/9/0	2/7/0	4/5/0	3/3/3	1/8/0
Problem	CCMO	CMOEA-MS	PPS-NSGA-II	PPS-SPEA2	PPS-SPEA2-SDE
DAS-CMOP1	1.7668e-2 (3.81e-2) ≈	1.3268e-1 (1.74e-1) −	4.5949e-3 (5.61e-3) +	6.5184e-3 (7.66e-3) +	1.0679e-2 (9.89e-3)
DAS-CMOP2	2.6768e-2 (2.57e-2) −	2.0677e-2 (1.85e-2) −	3.9454e-3 (1.28e-4) −	3.1447e-3 (1.04e-4) ≈	3.1145e-3 (1.23e-4)
DAS-CMOP3	5.6668e-2 (4.25e-2) −	1.3579e-1 (5.59e-2) −	4.0409e-2 (4.69e-2) ≈	3.9968e-2 (4.68e-2) ≈	4.4131e-2 (5.04e-2)
DAS-CMOP4	1.1382e-3 (1.74e-5) +	1.9129e-2 (6.85e-2) −	1.6106e-3 (1.18e-4) +	2.1404e-2 (6.16e-2) −	1.0109e-2 (3.64e-2)
DAS-CMOP5	2.8517e-3 (1.41e-4) +	2.6534e-3 (3.04e-5) +	3.8773e-3 (1.81e-4) −	3.0523e-3 (1.06e-4) +	3.2926e-3 (2.56e-4)
DAS-CMOP6	1.9788e-2 (4.30e-3) +	2.6165e-2 (2.32e-2) ≈	1.7381e-2 (3.30e-3) +	2.5254e-2 (2.24e-2) ≈	1.9884e-2 (2.62e-3)
DAS-CMOP7	1.7724e-2 (2.92e-4) +	1.7300e-2 (2.18e-4) +	2.9693e-2 (4.10e-3) +	3.9982e-2 (1.37e-2) ≈	4.3865e-2 (1.50e-2)
DAS-CMOP8	2.2393e-2 (3.39e-4) +	2.3227e-2 (8.25e-4) +	3.8502e-2 (1.14e-2) +	6.0143e-2 (1.83e-2) ≈	6.1326e-2 (1.48e-2)
DAS-CMOP9	2.2234e-2 (3.10e-4) +	2.2888e-2 (2.65e-4) +	2.8611e-2 (8.23e-4) +	2.3668e-2 (3.98e-4) +	3.0362e-2 (1.59e-3)
+/ − / ≈	6/2/1	4/4/1	6/2/1	3/1/5	

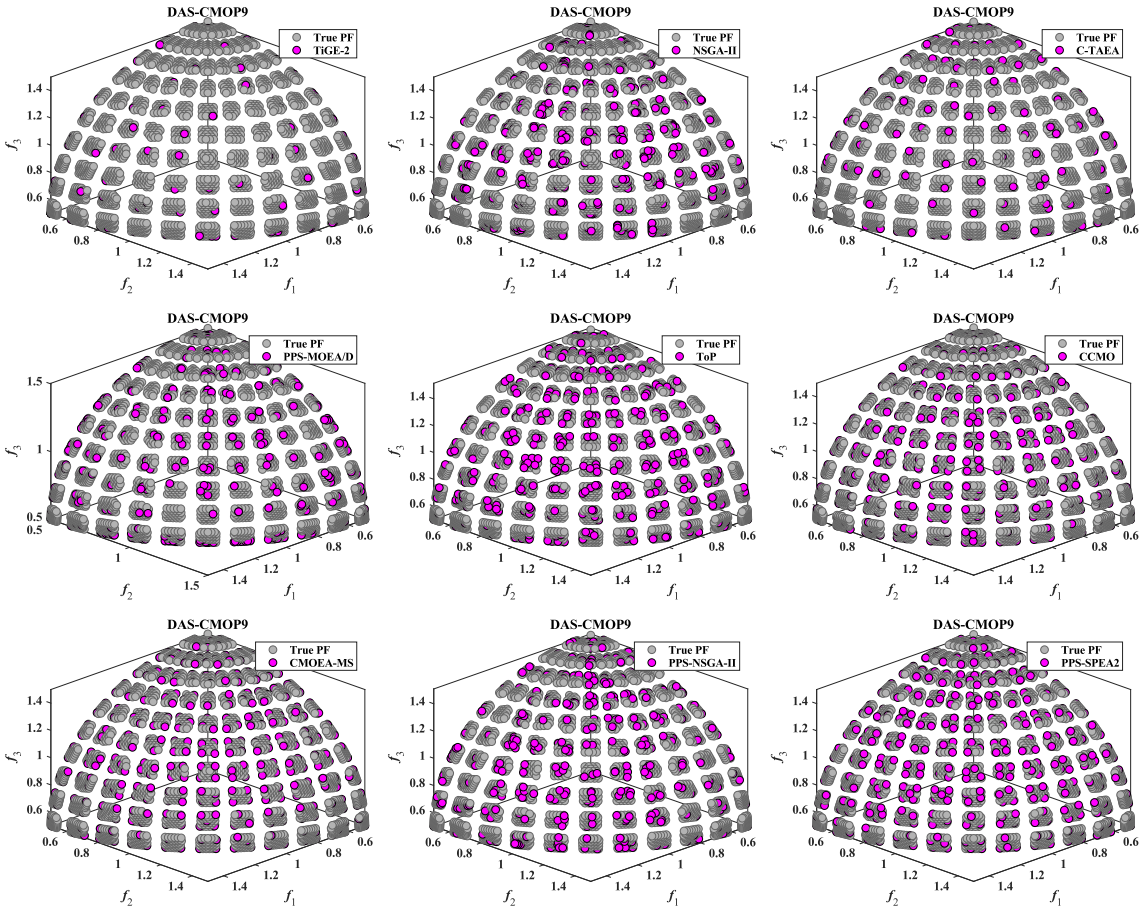


Figure 6.4: The non-dominated solution set with the medium IGD value obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, and PPS-SPEA2 on DAS-CMOP9

on DOC6. And the non-dominated and feasible solutions obtained by PPS-SPEA2-SDE on DOC6 is presented in Fig. 6.3. For a given DOC6 with a disconnected landscape, it is obvious that the proposed three algorithms PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE have better convergence and diversity performance than other compared seven algorithms. TiGE-2, C-TAEA, PPS-MOEA/D and ToP can only find a few feasible solutions since they encounter difficulties in both decision and objective constraints. NSGA-II, CCMO and CMOEA-MS can find a set of solutions far away from the constrained Pareto front. It is further demonstrated that the three proposed algorithms have a good and stable performance for solving various CMOPs.

Table 6.3: The IGD values obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on the DOC benchmark suite. The best result in each row is highlighted. 'N/A' indicates that no feasible solution is found.

Problem	TiGE-2	NSGA-II	C-TAEA	PPS-MOEA/D	ToP
DOC1	2.1126e+0 (1.73e+0) −	3.7007e+0 (2.44e+0) −	5.2585e+2 (2.80e+2) −	6.9749e-2 (4.79e-2) −	6.0467e-3 (4.13e-4) +
DOC2	N/A	N/A	N/A	3.3691e-1 (1.71e-1) −	N/A
DOC3	5.4385e+2 (1.87e+2) ≈	7.0845e+2 (2.43e+2) ≈	N/A	2.4354e+2 (3.09e+2) ≈	1.8863e+2 (1.56e+2) +
DOC4	2.2657e+0 (1.62e+0) −	1.0049e+0 (1.29e+0) −	2.4092e+2 (2.67e+2) −	2.9778e-1 (7.09e-2) −	1.0287e-1 (8.61e-2) ≈
DOC5	8.8488e+1 (3.68e+1) ≈	N/A	N/A	3.2705e+1 (1.01e+2) ≈	2.5523e+1 (5.04e+1) ≈
DOC6	9.8179e-1 (5.95e-1) −	2.3144e+0 (2.25e+0) −	2.5379e+1 (1.89e+1) −	5.0652e-1 (9.01e-2) −	4.8518e+0 (1.80e+0) −
DOC7	3.5848e+0 (2.07e+0) −	5.4283e+0 (2.24e+0) −	N/A	5.0754e-1 (2.09e-1) −	9.3749e-1 (6.23e-1) −
DOC8	1.2315e+2 (7.42e+1) −	8.0406e+1 (6.61e+1) −	4.1052e+2 (1.79e+2) −	7.3879e+1 (1.52e+1) −	5.1458e+1 (2.09e+1) −
DOC9	3.1946e-2 (1.28e-2) −	8.3123e-2 (6.01e-2) −	9.9191e-1 (1.65e-1) −	2.4100e-1 (1.93e-2) −	1.8848e-1 (3.41e-2) −
+/ − / ≈	0/6/2	0/6/1	0/5/0	0/7/2	2/4/2
Problem	CCMO	CMOEA-MS	PPS-NSGA-II	PPS-SPEA2	PPS-SPEA2-SDE
DOC1	4.8668e+0 (3.97e+0) −	3.7414e+0 (3.16e+0) −	6.1709e-3 (2.41e-4) +	5.4443e-3 (3.23e-4) +	9.2780e-3 (1.05e-3)
DOC2	N/A	N/A	3.6254e-2 (1.10e-1) +	2.9815e-2 (9.92e-2) +	4.5584e-2 (1.14e-1)
DOC3	6.1836e+2 (1.58e+2) ≈	6.6583e+2 (2.18e+2) ≈	1.0703e+3 (4.32e+2) ≈	8.5551e+2 (5.73e+2) ≈	7.9078e+2 (6.11e+2)
DOC4	1.1935e+0 (6.98e-1) −	9.4098e-1 (7.97e-1) −	1.9586e-2 (2.39e-3) +	2.0794e-2 (2.74e-3) ≈	2.4830e-2 (7.57e-3)
DOC5	N/A	9.0087e+1 (3.18e+1) ≈	3.0733e+1 (5.71e+1) ≈	2.6195e+1 (5.41e+1) ≈	5.6742e+1 (8.42e+1)
DOC6	3.3319e+0 (4.05e+0) −	1.7154e+0 (1.54e+0) −	3.6421e-3 (2.31e-4) +	3.1091e-3 (2.74e-4) +	5.1402e-2 (2.66e-1)
DOC7	5.6675e+0 (2.20e+0) −	5.8488e+0 (3.01e+0) −	7.1706e-2 (3.01e-1) −	2.4397e-3 (1.34e-4) +	7.8216e-3 (2.23e-2)
DOC8	6.7197e+1 (5.11e+1) −	1.2301e+2 (6.89e+1) −	8.1841e-2 (1.20e-2) −	4.1026e-2 (2.13e-3) −	2.4695e-2 (2.16e-3)
DOC9	9.7203e-2 (9.76e-2) −	4.6108e-2 (6.75e-2) −	9.0207e-2 (8.91e-3) −	5.4600e-2 (7.23e-3) −	1.1604e-2 (9.67e-3)
+/ − / ≈	0/6/1	0/6/2	4/3/2	4/2/3	

6.4 Summary

In this chapter, three constrained multi-objective evolutionary algorithms (CMOEAs) have been developed to solve a variety of CMOPs. NSGA-II, SPEA2 and SPEA2-SDE are embedded into PPS framework for solving CMOPs, and then PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE are realized. In the push search stage, PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE algorithms search for the unconstrained solutions without considering any constraints. In the pull search stage, the constraint handling mechanism is integrated into these algorithms to pull the unconstrained solutions to approximate the constrained Pareto fronts. Three challenging constrained benchmark suites (LIR-CMOP, DAS-CMOP, and DOC) were used to test the performance of the proposed algorithms by comparing them to the other state-of-the-art CMOEAs. The experimental results demonstrate the efficiency and good versatility of the proposed algorithms.

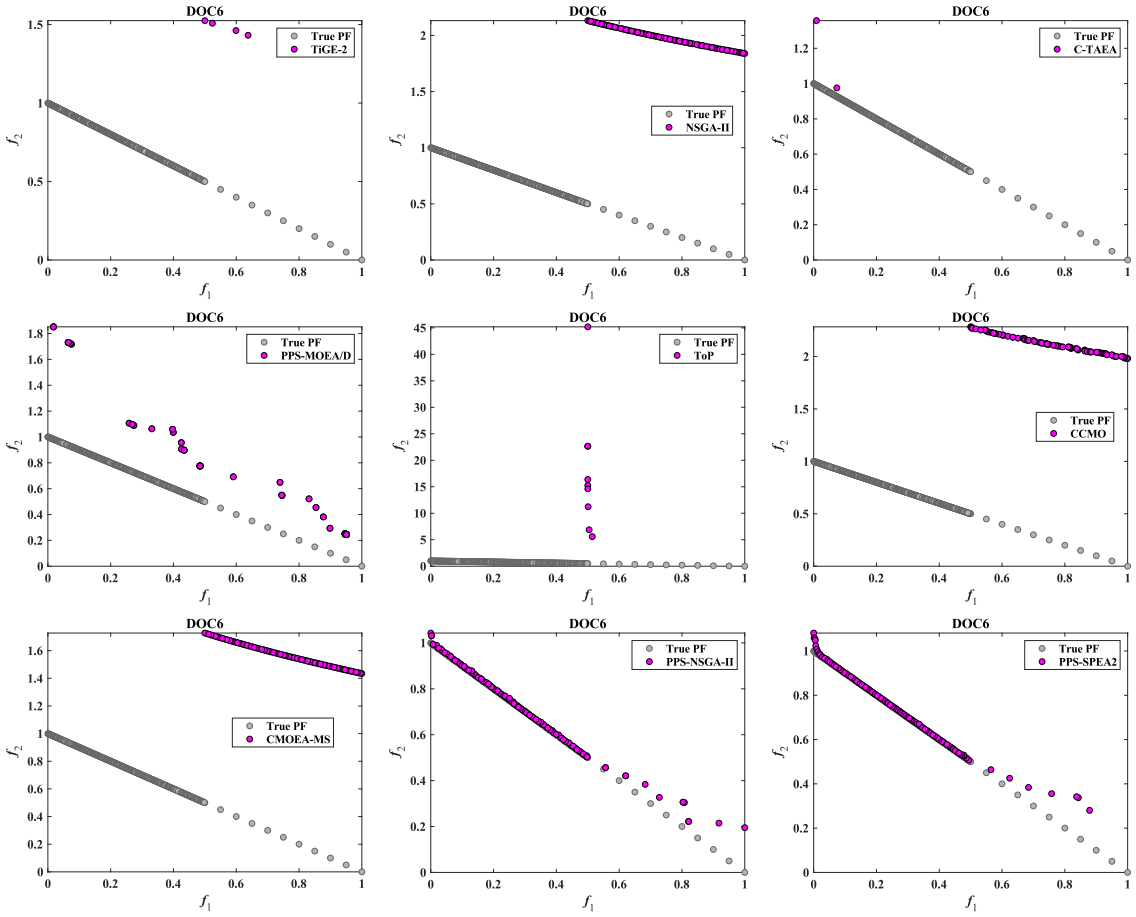


Figure 6.5: The non-dominated solution set with the medium IGD value obtained by TiGE-2, NSGA-II, C-TAEA, PPS-MOEA/D, ToP, CCMO, CMOEA-MS, PPS-NSGA-II, and PPS-SPEA2 on DOC6

Part III

Optimization in Offloading

Chapter 7

Constrained Multi-objective Optimization for Offloading

In this chapter, we adopt the proposed three constrained optimization algorithms PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE of **Chapter 6** to solve IoT-enabled constrained computation offloading problems in collaborative edge and cloud computing.

Computation offloading problems are often constrained optimization problems and NP-hard [6, 104]. However, there are few studies that combine constrained multi-objective optimization with computation offloading in the collaborative mobile cloud and edge computing (MCC and MEC). The motivation of this chapter is to treat the computation offloading problem as a constrained multi-objective optimization problem (CMOP) and then we focus on the state-of-the-art constrained multi-objective evolutionary algorithms (CMOEAs) for solving that.

At first, a constrained multi-objective computation offloading model considering time and energy consumption is established in the mobile environment. Then a multi-server multi-user multi-task computation offloading experimental scenario with a different number of IoT devices is used to evaluate the performance of three proposed algorithms and other compared algorithms as well as representative offloading schemes. The three designed CMOEAs can solve the constrained computation offloading problems well and achieve good results.

7.1 Constrained Offloading Model

In this section, we consider a collaborative MEC and MCC network with multiple mobile devices (MDs), multiple edge servers and multiple cloud servers. The computation tasks in the MDs can be

executed locally or offloaded to the edge/cloud servers.

7.1.1 System Model

Fig. 7.1 presents the system model composed by L cloud servers, K edge servers, and N mobile devices (MDs). Each MD can communicate with the edge server with a wireless link, whereas the edge server and cloud server are connected through a wired link. Without loss of generality, we assume that each mobile device has M independent tasks. We denote the set of MDs as $\mathcal{N} = \{1, 2, \cdots, N\}$ and the set of tasks as $\mathcal{M} = \{1, 2, \cdots, M\}$, and the set of servers as $\mathcal{K} = \{0, 1, 2, \cdots, K, K+1, \cdots, K+L\}$, where server 0 denotes MD itself and servers $\{1, 2, \cdots, K\}$ denote the edge servers and servers $\{K+1, \cdots, K+L\}$ denote the cloud servers. In each MD, different tasks can decide to be processed by MD itself or remotely processed by edge/cloud servers. We denote $a_{nm} \in \{0, 1, 2, \cdots, K, K+1, \cdots, K+L\}$ as the offloading decision that MD n's m-th task is assigned to mobile device or cloud/edge servers, where $n \in \mathcal{N}$ and $M \in \mathcal{M}$. Especially, $a_{nm} = 0$ means that MD n chooses to locally execute its m-th task, $a_{nm} \in \{1, 2, \cdots, K\}$ indicates that MD n's m-th task is offloaded to the edge servers and $a_{nm} \in \{K+1, K+2, \cdots, K+L\}$ represents that MD n's m-th task is offloaded to the cloud servers. Overall, every task must be processed locally or by the edge/cloud servers, whose offloading decision depends on:

$$a_{nm} = \begin{cases} 0, & \text{local computing,} \\ \in \{1, 2, \cdots, K\}, & \text{edge computing,} \\ \in \{K+1, K+2, \cdots, K+L\}, & \text{cloud computing.} \end{cases} \tag{7.1}$$

where $n \in \mathcal{N}$ and $M \in \mathcal{M}$. Since both response time and energy consumption play a significant role in the performance of computation offloading for MDs, we consider these two objectives as QoS metrics. The detailed operations of the communication and computation process are illustrated in Sections 7.1.2 and 7.1.3, respectively. The key notations used in this chapter are listed in Table 7.1.

7.1.2 Communication Model

Considering the communication cost between the MDs and edge/cloud servers, we first analyze the transmission time and energy consumption in the communication model. We set a tuple $(\alpha_{nm}, \gamma_{nm})$ to represent MD n's m-th task, where α_{nm} is the data size and γ_{nm} is the required number of CPU cycles to finish the task. When one of the MD n's task m is offloaded to the edge server $k \in \{1, 2, \cdots, K\}$, the whole processing of task m includes transmitting and edge comput-

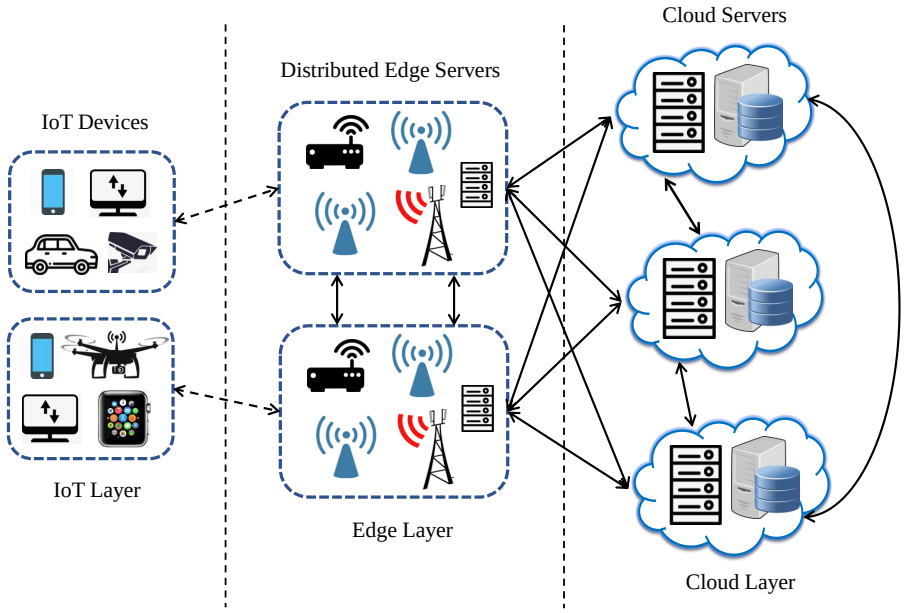


Figure 7.1: System model of local-edge-cloud computation offloading

ing phase. Let B_{nk}^{UE} denote the allocated upload bandwidth between the MD n and the edge server k. We neglect the influence of the process when the edge server returns the results back to MDs since the data size of feedback information is small in general [28]. The upload transmission time for offloading MD n's m-th task to the edge server k can be calculated as:

$$T_{nm}^{UE} = \frac{\alpha_{nm}}{B_{nk}^{UE}} \tag{7.2}$$

The energy consumption for uploading MD n's m-th task to the edge server k can be quantified as:

$$E_{nm}^{UE} = P_n^{TX} T_{nm}^{UE} \tag{7.3}$$

where P_n^{TX} is the transmission energy consumption power of the MD n.

When one of the MD n's task is offloaded to the cloud server $k \in \{K+1, K+2, \cdots, K+L\}$, one of the edge servers is selected as a relay node between the MD and the cloud server. We assume that the task is first transmitted to the edge server $\widetilde{k}$ through a wireless link, then the edge server $\widetilde{k}$ will forward the task to the central cloud server k via a wired link. The upload transmission time for

Table 7.1: Key notations

Notation	Description
a_{nm}	The offloading decision of m-task of n-th MD
α_{nm}	Input data size of the task m of MD n
γ_{nm}	Total CPU cycles of the task m of MD n
B_{nk}^{UE}	The transmission bandwidth between MD n and edge server k
P_n^{TX}	The transmission power consumption of MD n
T_{nm}^{UE}	The transmission time for offloading task m of MD n to edge server k
E_{nm}^{UE}	The transmission energy consumption for offloading task m of MD n to edge server k
τ	The propagation latency between an edge server and a cloud server
T_n^{CommE}	The transmission latency from MD n to edge servers
T_n^{CommC}	The transmission latency from MD n to cloud servers
T_n^{Comm}	The total communication delay of MD n for completing all M tasks
E_n^{CommE}	The communication energy consumption from MD n to edge servers
E_n^{CommC}	The communication energy consumption from MD n to cloud servers
E_n^{Comm}	The total communication energy consumption of MD n for completing all M tasks
f_l, f_e, f_c	The CPU frequency in mobile devices, edge servers and cloud servers
T_{nm}^{Comp}	The computation latency of m task of MD n
T_n^{CompL}	The total computation latency of MD n in mobile devices
T_n^{CompE}	The total computation latency of MD n in edge servers
T_n^{CompC}	The total computation latency of MD n in cloud servers
E_n^{Comp}	The total computation energy consumption of MD n
T_n	The overall completion time of executing all M tasks of MD n
T	The overall completion time of executing all tasks of all MDs
E_n	The energy consumption of executing all M tasks of MD n
E	The total energy consumption of executing all tasks of all MDs
T^{Cons}	The response time constraint
E^{Cons}	The energy consumption constraint

offloading MD n's m-th task to the cloud server k can be calculated as:

$$T_{nm}^{UC} = \frac{\alpha_{nm}}{B_{\widetilde{nk}}^{UE}} + \tau \tag{7.4}$$

where τ denotes the propagation delay between edge servers and cloud servers. We focus on the energy consumption of MDs, thus the energy consumption for uploading MD n's m-th task to the

cloud server k can be quantified as:

$$E_{nm}^{UC} = P_n^{TX} \times \left(T_{nm}^{UC} - \tau\right) \tag{7.5}$$

When the task is executed locally, there is no communication latency. Hence, the total communication delay of MD n for completing all M tasks can be expressed as:

$$T_n^{Comm} = T_n^{CommE} + T_n^{CommC} \tag{7.6}$$

where

$$\begin{cases} T_n^{CommE} = \sum\limits_{m=1}^{M} T_{nm}^{UE}, \ a_{nm} \in \{1, 2, \cdots, K\}, \\ T_n^{CommC} = \sum\limits_{m=1}^{M} T_{nm}^{UC}, \ a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{7.7}$$

Then the overall communication energy consumption of MD n for completing all M tasks can be calculated as:

$$E_n^{Comm} = E_n^{CommE} + E_n^{CommC} \tag{7.8}$$

where

$$\begin{cases} E_n^{CommE} = \sum\limits_{m=1}^{M} E_{nm}^{UE}, \ a_{nm} \in \{1, 2, \cdots, K\}, \\ E_n^{CommC} = \sum\limits_{m=1}^{M} E_{nm}^{UC}, \ a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{7.9}$$

7.1.3 Computation Model

We denote f_l, f_e, f_c as the number of CPU cycles for the mobile devices, the edge servers and the cloud servers, respectively. In general, the computation capability of the cloud servers is more powerful than the edge servers, and the edge servers have better computation capability than the mobile devices, as $f_l \ll f_e \ll f_c$.

When each task is determined to be offloaded to edge or cloud servers, the edge or cloud servers start to process it after all the input data has been received by the edge or cloud servers. The computation latency of MD n's m-th task in MDs, the edge servers and cloud servers are calculated as:

$$T_{nm}^{Comp} = \begin{cases} \frac{\gamma_{nm}}{f_l}, & a_{nm} = 0, \\ \frac{\gamma_{nm}}{f_e}, & a_{nm} \in \{1, 2, \cdots, K\}, \\ \frac{\gamma_{nm}}{f_c}, & a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{7.10}$$

Thus, the total computation latency of MD n for completing all M tasks can be expressed as:

$$\begin{cases} T_n^{CompL} = \sum_{m=1}^{M} \frac{\gamma_{nm}}{f_l}, a_{nm} = 0, \\ T_n^{CompE} = \sum_{m=1}^{M} \frac{\gamma_{nm}}{f_e}, a_{nm} \in \{1, 2, \cdots, K\}, \\ T_n^{CompC} = \sum_{m=1}^{M} \frac{\gamma_{nm}}{f_c}, a_{nm} \in \{K+1, \cdots, K+L\}. \end{cases} \tag{7.11}$$

In this model, we only consider the energy consumption at MDs. Specially, we use P_n^L to denote the local energy consumption power of MD n. Then MD n's energy consumption for executing its task m locally is given by:

$$E_{nm}^{Comp} = P_n^L \times \frac{\gamma_{nm}}{f_l} \tag{7.12}$$

Hence, the total computation energy consumption of MD n can be expressed as:

$$E_n^{Comp} = P_n^L \times T_n^{CompL} \tag{7.13}$$

7.1.4 Problem Formulation

The processing latency consists of communication and computation latency, and the total delay of executing all M tasks of MD n can be given by:

$$T_n = \max\{T_n^{CompL}, T_n^{CompE} + T_n^{CommE}, T_n^{CompC} + T_n^{CommC}\} \tag{7.14}$$

The total completion time of executing all tasks of all MDs can be expressed:

$$T = \max\left\{\sum_{n=1}^{N} T_n^{CompL}, \sum_{n=1}^{N}\left(T_n^{CompE} + T_n^{CommE}\right), \sum_{n=1}^{N}\left(T_n^{CompC} + T_n^{CommC}\right)\right\}$$

The energy consumption of executing all M tasks of MD n can be given by:

$$E_n = E_n^{Comp} + E_n^{Comm} \tag{7.15}$$

The total energy consumption of executing all tasks of all MDs can be expressed as:

$$E = \sum_{n=1}^{N}\left(E_n^{Comp} + E_n^{Comm}\right) \tag{7.16}$$

Hence, the computation offloading problem can be formalized as follows:

$$\min: [T, E], \tag{7.17}$$

$$s.t.: a_{nm} \in \{0, 1, 2, \cdots, K, K+1, \cdots, K+L\}, \tag{7.18}$$

$$|a_{nm}| = 1, \tag{7.19}$$

$$T \leq T^{\text{Cons}}, \tag{7.20}$$

$$E \leq E^{\text{Cons}}, \tag{7.21}$$

where the first and second constraints indicate that each task is assigned to one server, the third constraint denotes that MDs have constraints of response time deadline, and the last constraint represents the energy consumption limits. Hence, we set up a local-edge-cloud constrained multi-objective computation offloading model.

7.2 Performance Evaluation

In this section, we use the proposed three algorithms PPS-NSGA-II (**Algorithm 7**), PPS-SPEA2 and PPS-SPEA2-SDE (**Algorithm 8**) of **Chapter 6** to solve constrained multi-objective computation offloading optimization problems.

7.2.1 Experimental Setup

We set up the multi-server multi-user multi-task computation offloading scenario in the local-edge-cloud environment. The number of mobile devices is selected between 10 and 100. The number of independent tasks of each MD is $M = 5$. We set the number of edge servers $K = 5$ and the number of cloud servers $L = 2$. In the following scenarios, we consider the CPU frequencies of each MD, each edge server, and each cloud server are 0.6 GHz, 10 GHz and 1 THz, respectively [27]. The transmitting power P_n^{TX} of all MDs is 0.2 W. The power consumption of all MDs is 0.7 W. The round-trip propagation delay between edge servers and cloud servers is $\tau = 15$ ms. The bandwidth between MDs and edges is randomly selected from [8,15] MBps. The data size of each task is uniformly distributed between 10 MB and 30 MB. The total CPU cycles for finishing the task are assumed to be proportional to the input data size [43], i.e., $\gamma_{nm} = \rho\alpha_{nm}$. Here the parameter ρ denotes the computation to data radio for different types of applications. Table 7.2 lists some values of ρ for various applications [26, 71]. For example, the label A represents the gzip application and $\rho = 330$ cycles/byte. By default, the type A application is taken as an example.

Table 7.2: Application complexity

Application	Labels	ρ (cycles/byte)
gzip	A	330
pdf2text (N900 data sheet)	B	960
x264 CBR encode	C	1900
html2text	D	5900
pdf2text (E72 data sheet)	E	8900

To test the performance of the proposed algorithms, we compare PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE with other five algorithms (TiGE-2 [117], constrained NSGA-II [24], PPS-MOEA/D [33], ToP [66] and CMOEA-MS [91]) to solve five offloading problems, which consider the number of MDs N = [10, 30, 50, 70, 100]. For a fair comparison, the population size of all algorithms is set to 100, and the number of iterations is 1000. The solution encoding style adopts the real-encoding method, which means that each task is assigned to a specific server including edge and cloud servers. We apply the HV as the performance metric to evaluate the performance of these compared algorithms. Each algorithm is executed 30 times independently and the average and standard deviation of the metric values are recorded. The Wilcoxon rank sum test is also operated.

7.2.2 Convergence Analysis

As listed in Table 7.3, the proposed PPS-NSGA-II has achieved the best performance on four offloading problems, while only CMOEA-MS gets one best result among other algorithms. It is necessary to point out that CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE share similar overall performance based on the Wilcoxon rank sum test, which outperform other four compared algorithms (TiGE-2, NSGA-II, PPS-MOEA/D and ToP). We can also observe that PPS-MOEA/D may obtain good performance for solving the benchmark suites, while encountering difficulties in solving discrete computation offloading problems.

We can observe that ToP cannot find any feasible solutions on N = 50 and 100 offloading problems as shown in Figs. 7.2b and 7.2c. TiGE-2, NSGA-II and PPS-MOEA/D can obtain a few feasible and non-dominated solutions. NSGA-II, CMOEA-MS, PPS-SPEA2 and PPS-SPEA2-SDE may get good results about the small-scale offloading problems (e.g., N = 10), while their performance deteriorates with the growth of the number of mobile devices, especially for the algorithm NSGA-II. PPS-NSGA-II can always obtain a set of well-distributed and well-converged feasible solutions for different offloading problems.

Table 7.3: The HV values obtained by TiGE-2, NSGA-II, PPS-MOEA/D, ToP, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on five offloading problems. The best result in each row is highlighted. 'N/A' indicates that no feasible solution is found.

Problem	N	TiGE-2	NSGA-II	PPS-MOEA/D	ToP
Offloading1	10	3.2142e-1 (1.60e-2) $-$	3.4539e-1 (9.90e-3) $-$	3.2459e-1 (8.59e-3) $-$	2.8488e-1 (2.14e-2) $-$
Offloading2	30	2.7354e-1 (6.73e-3) $-$	1.5740e-1 (1.36e-1) $-$	7.6545e-2 (1.23e-1) $-$	5.1039e-2 (1.08e-1) $-$
Offloading3	50	2.6492e-1 (7.76e-3) $-$	1.8942e-1 (1.31e-1) $-$	1.0244e-1 (1.32e-1) $-$	N/A
Offloading4	70	2.4061e-1 (8.49e-2) $-$	2.4333e-1 (8.63e-2) $-$	1.2822e-1 (1.35e-1) $-$	5.0319e-2 (1.06e-1) $-$
Offloading5	100	1.8015e-1 (1.24e-1) $-$	1.2960e-1 (1.37e-1) $-$	5.1053e-2 (1.08e-1) $-$	N/A
$+/-/\approx$		0/5/0	0/5/0	0/5/0	0/3/0
Problem	N	CMOEA-MS	PPS-NSGA-II	PPS-SPEA2	PPS-SPEA2-SDE
Offloading1	10	3.4966e-1 (9.82e-3) $\approx$	3.5209e-1 (7.42e-3) $\approx$	3.5051e-1 (7.43e-3) $\approx$	3.4799e-1 (7.17e-3)
Offloading2	30	2.9020e-1 (7.39e-3) $\approx$	2.8669e-1 (7.55e-3) $\approx$	2.8761e-1 (4.38e-3) $\approx$	2.8279e-1 (9.65e-2)
Offloading3	50	2.8118e-1 (1.17e-2) $\approx$	2.8837e-1 (5.51e-3) $\approx$	2.8482e-1 (8.87e-3) $\approx$	2.8182e-1 (1.18e-2)
Offloading4	70	2.8290e-1 (7.49e-3) $\approx$	2.8896e-1 (3.41e-3) $\approx$	2.8520e-1 (7.26e-3) $\approx$	2.8635e-1 (2.46e-3)
Offloading5	100	2.7111e-1 (9.13e-3) $\approx$	2.7197e-1 (1.11e-2) $\approx$	2.7020e-1 (1.35e-3) $\approx$	2.7055e-1 (5.87e-3)
$+/-/\approx$		0/0/5	0/0/5	0/0/5	

7.2.3 Performance of Different Offloading Schemes

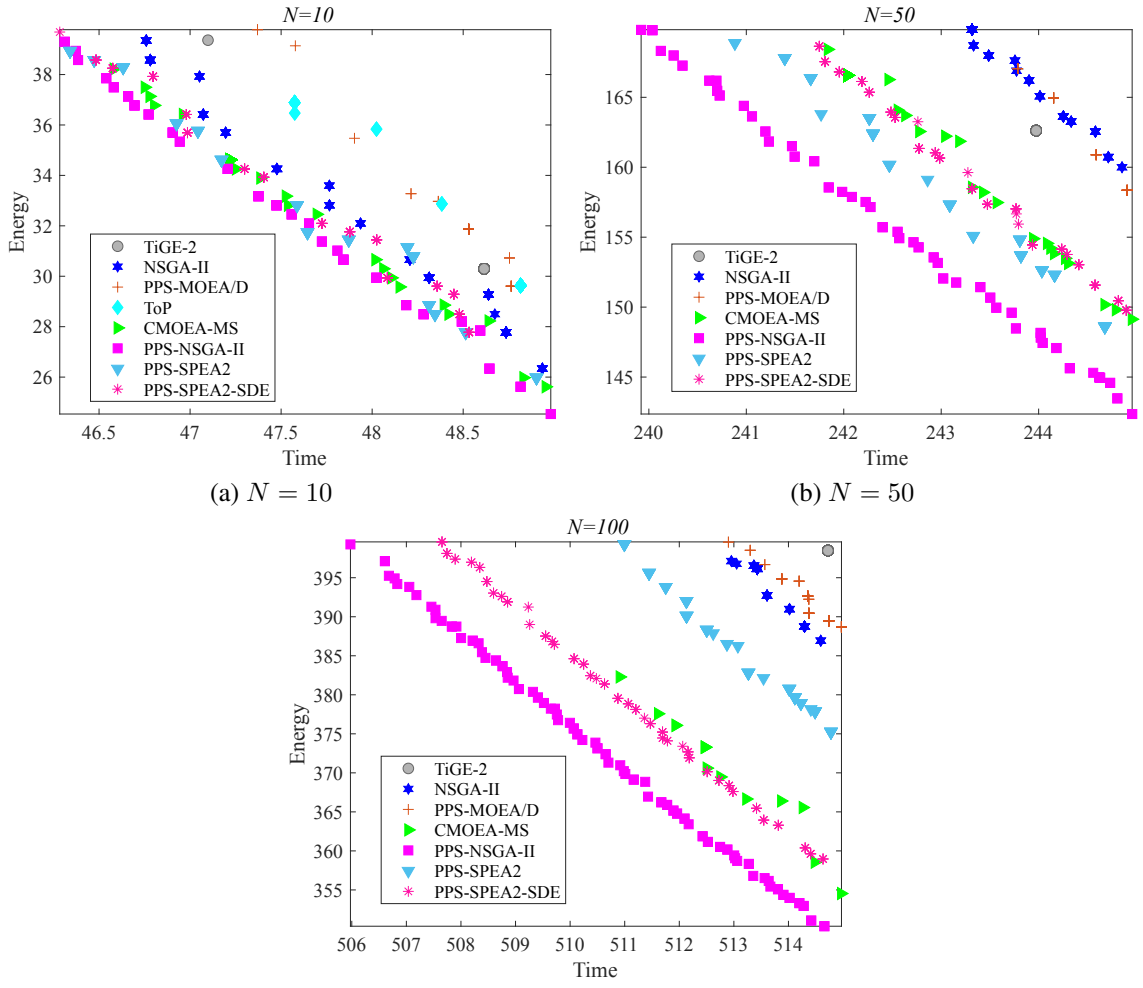
It has been demonstrated that PPS-NSGA-II has a good and stable performance in terms of both convergence and diversity on different offloading problems. To further evaluate the performance of PPS-NSGA-II for reducing response time and energy consumption, we compare PPS-NSGA-II with other four offloading schemes, which are Local Offloading Scheme (LOS), Edge Offloading Scheme (EOS), Cloud Offloading Scheme (COS), and Random Offloading Scheme (ROS). LOS, EOS, and COS represent that all tasks are executed locally, offloaded to edge servers and central cloud servers. ROS denotes that offloading decisions of all tasks are generated randomly. In order to better compare the effectiveness of different algorithms, we can design system cost and offloading gain of a weighted sum of time and energy as follows:

$$\text{SystemCost} = w \times T_{offloading} + (1 - w) \times E_{offloading} \tag{7.22}$$

$$\text{OffloadingGain} = \left[w \times \frac{T_{LOS} - T_{offloading}}{T_{LOS}} + (1 - w) \times \frac{E_{LOS} - E_{offloading}}{E_{LOS}} \right] \times 100\% \tag{7.23}$$

where $T_{offloading}$ and $E_{offloading}$ denote overall time and energy consumption of one specific offloading scheme, respectively. T_{LOS} and E_{LOS} denote the time and energy consumption of LOS, respectively. w is the weight trade-off parameter between time and energy, which can be set by the decision-maker. The larger w is, the more sensitive the response time is.

Figs. 7.3, 7.4, and 7.5 present the offloading gain of different offloading schemes under different

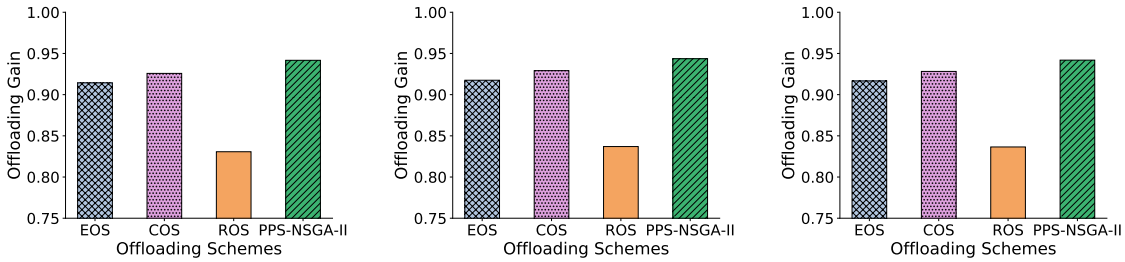


(a) $N = 10$ (b) $N = 50$

(c) $N = 100$

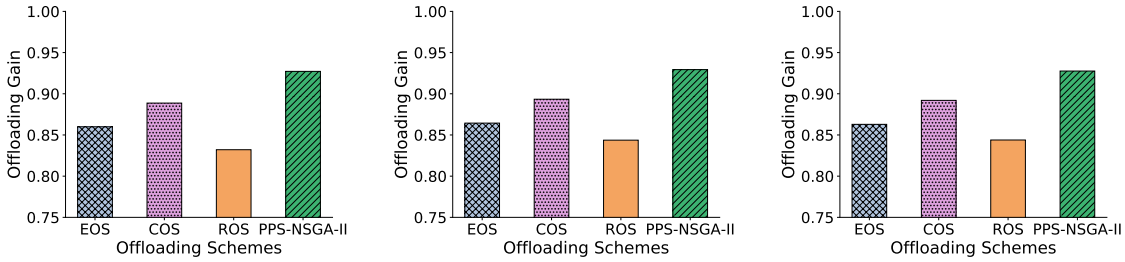
Figure 7.2: The non-dominated solution set with the medium HV value obtained by TiGE-2, NSGA-II, PPS-MOEA/D, ToP, CMOEA-MS, PPS-NSGA-II, PPS-SPEA2 and PPS-SPEA2-SDE on different offloading problems

weights. Compared with LOS, all the other offloading schemes benefit a lot with regard to time consumption and energy consumption. PPS-NSGA-II can obtain the best offloading gain compared with other offloading schemes among all the different offloading problems with different weights. COS achieves a better offloading gain performance than EOS since the cloud servers take advantage of powerful cloud resources. Offloading is more beneficial when focusing on energy usage than time consumption.



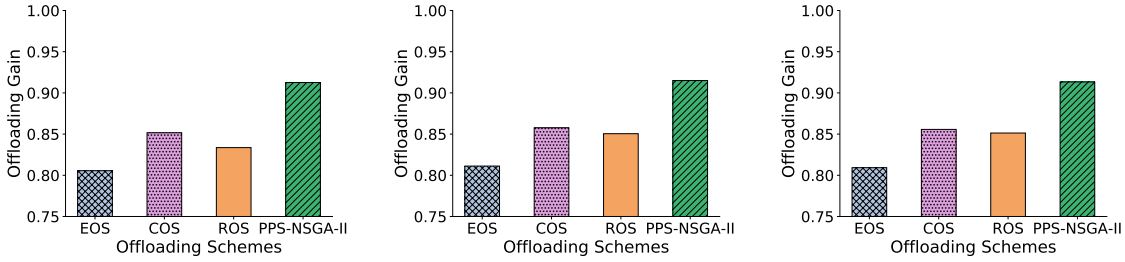
(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Figure 7.3: Offloading gain of different offloading schemes for $w = 0.2$



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Figure 7.4: Offloading gain of different offloading schemes for $w = 0.5$



(a) $N = 10$ (b) $N = 50$ (c) $N = 100$

Figure 7.5: Offloading gain of different offloading schemes for $w = 0.8$

7.2.4 Impact of Different Parameters

In this section, we analyze the impact of different parameters in collaborative edge-cloud computing networks, and w is set to 0.5 as well as N is equal to 10. Fig. 7.6 illustrates the performance of system cost and offloading gain on different offloading schemes under the different number of tasks

of each MD. PPS-NSGA-II gains the best performance compared with other offloading schemes. With the increasing number of tasks, the system cost of LOS grows much faster than EOS, COS, ROS, and PPS-NSGA-II. The offloading gain of the different offloading schemes stays stable since the system cost of EOS, COS, ROS, and PPS-NSGA-II belongs to a small relevant proportion of LOS.

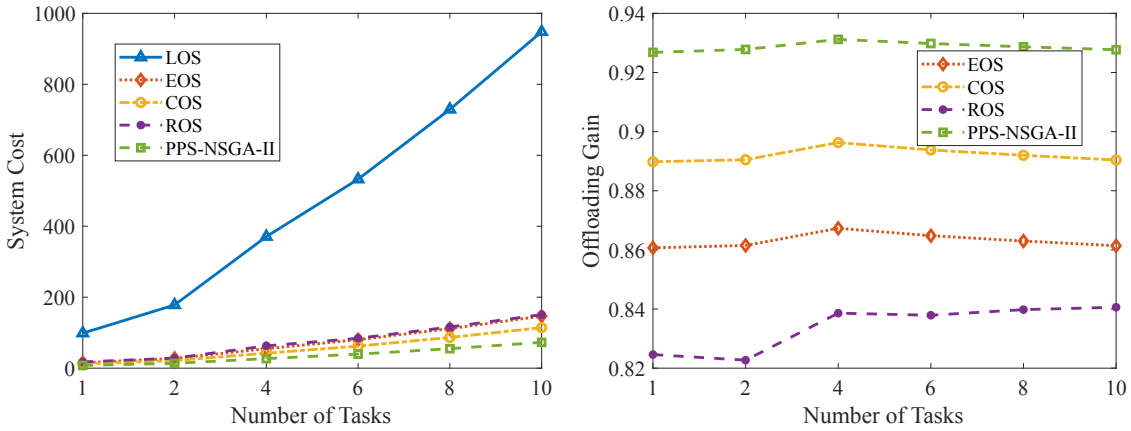


Figure 7.6: System cost and offloading gain on different offloading schemes under different number of tasks

Fig. 7.7 shows the performance of system cost and offloading gain on different offloading schemes under the different wireless bandwidth between MDs and edge servers. LOS does not change with the increment of wireless bandwidth. Both the performance of the system cost as well as offloading gain of the other four offloading schemes (EOS, COS, ROS, and PPS-NSGA-II) improve due to larger wireless bandwidth. In addition, with the increment of wireless bandwidth, the performance improves very fast at the beginning and then becomes small. It is worth noting that the offloading gain of EOS and COS may be negative when the wireless bandwidth is small, which means that a computing task should not be offloaded to edge or cloud servers due to large communication cost in the case wireless bandwidth is small enough.

Fig. 7.8 presents the performance of system cost and offloading gain on different offloading schemes under different edge server CPU frequency. The performance of LOS and COS do not change no matter what the CPU frequency of the edge servers'. With the increment of edge server CPU frequency, the performance of system cost and offloading gain of EOS grows faster than ROS and PPS-NSGA-II. However, PPS-NSGA-II still achieves the best results among all the offloading schemes.

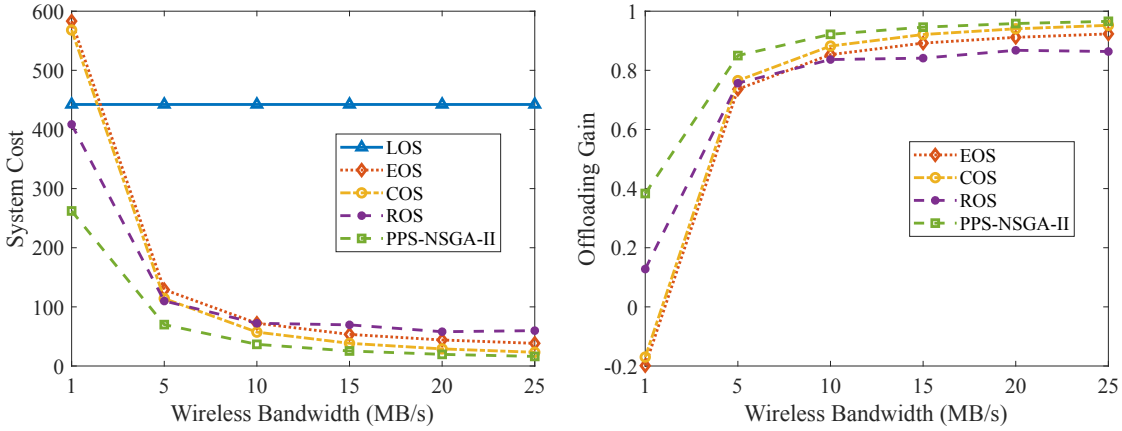


Figure 7.7: System cost and offloading gain on different offloading schemes under different wireless bandwidth

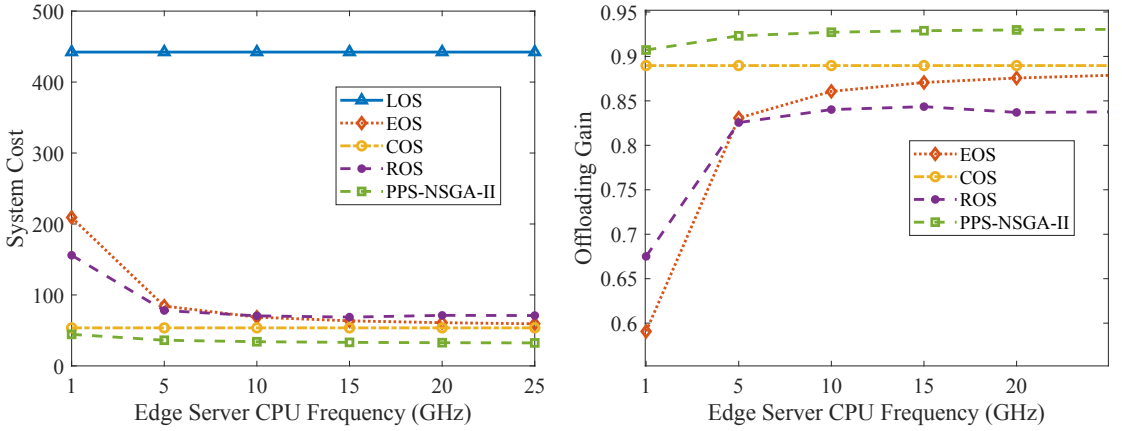


Figure 7.8: System cost and offloading gain on different offloading schemes under different edge server CPU frequency

7.2.5 Impact of Different Types of Applications

Fig. 7.9 illustrates the performance of system cost and offloading gain on different offloading schemes under different types of applications. With the increment of parameter ρ of different applications, the computing delay increases directly. The system cost of LOS increases very fast due to the poor computing capability of MDs, while COS and PPS-NSGA-II grow slowly due to the powerful computing resources at the cloud servers. PPS-NSGA-II will make more offloading decisions to offload the tasks to cloud servers. In addition, the system cost of EOS and ROS grow gradually

and the increasing speed of EOS is slower than ROS. Furthermore, the performance of offloading gain of EOS, COS and PPS-NSGA-II is much better than ROS, and the COS and PPS-NSGA-II achieve the best and similar results due to the increment of parameter ρ of different applications.

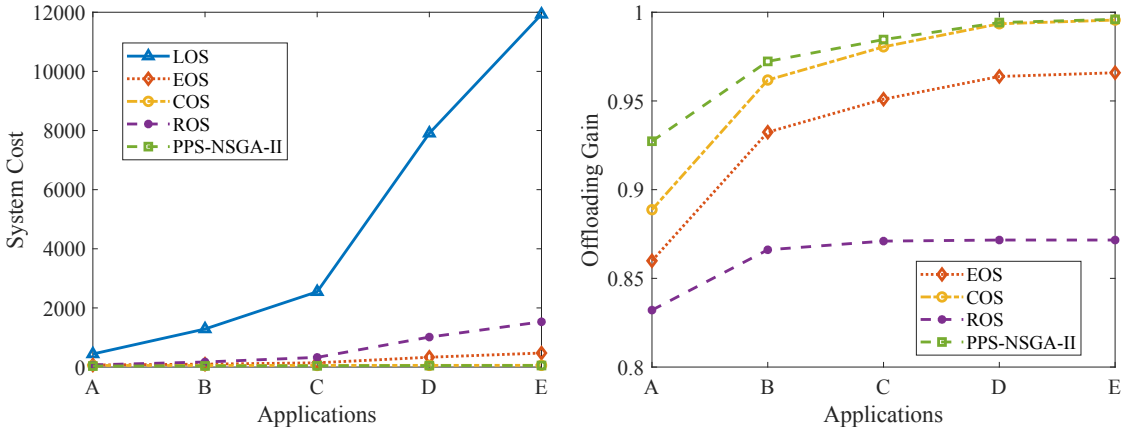


Figure 7.9: System cost and offloading gain on different offloading schemes under different types of applications

7.3 Summary

In this chapter, the three designed CMOEAs were applied to solve IoT-edge-cloud constrained multi-objective computation offloading problems. We establish a constrained multi-objective computation offloading model for minimizing time and energy consumption of IoT devices. The proposed PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE are compared with other state-of-the-art CMOEAs and different offloading schemes to deal with different computation offloading problems. The experimental results show the proposed algorithms can achieve better performance than other compared representative algorithms, and outperform other different offloading policies.

Chapter 8

Large-scale Offloading in Edge-Cloud Computing

In this chapter, we propose and compare two evolutionary large-scale sparse multi-objective optimization algorithms (called ELSMO) to solve collaborative edge-cloud large-scale computation offloading problems.

The traditional multi-objective evolutionary algorithms (MOEAs) will encounter difficulties in dealing with large-scale computation offloading problems when there exists a large number of mobile devices having a great many computation tasks. To remedy this issue, in this chapter we set up a collaborative edge-cloud computation offloading large-scale sparse multi-objective optimization model with binary encoding. Focus on large-scale optimization methods, based on the dimensionality reducing and decision variable analysis methods [88, 90], two evolutionary large-scale sparse multi-objective optimization algorithms (called ELSMO) are applied and compared to solve the large-scale offloading problems. Compared with other MOEAs and offloading schemes, the proposed algorithms are competitive on different large-scale test problems.

8.1 Restricted Boltzmann Machine

Restricted Boltzmann Machine (RBM) is a stochastic neural network, which consists of an input layer and a hidden layer, as shown in Fig. 8.1. The nodes in the two layers are binary variables obeying binomial distribution. RBM can be used to reduce the dimensionality through unsupervised learning. Given an input vector x, the value of each node h_j in the hidden layer is set to 1 with a

probability:

$$p\left(h_j = 1 \,|x\right) = \sigma\left(a_j + \sum_{i=1}^{D} x_i w_{ij}\right) \tag{8.1}$$

where a_j is the basis, w_{ij} is the weight, D is the dimensionality of input layer, and $\sigma\left(x\right) = 1/\left(1+\exp\left(-x\right)\right)$ is the sigmoid function. Through comparing the probability $p\left(h_j = 1 \,|x\right)$ with a uniformly distributed random value in $[0, 1]$, the binary value of each node h_j can be obtained. In the same way, the reconstructed value of each node x'_i in the input layer is set to 1 with a probability:

$$p\left(x'_i = 1 \,|h\right) = \sigma\left(a'_j + \sum_{j=1}^{K} h_j w_{ij}\right) \tag{8.2}$$

where K is the dimensionality of the hidden layer. Hinton [42] proposed the contrastive divergence algorithm to train RBM, which aims to minimize the reconstruction error between the reconstructed vector x' and the original input x by finding the suitable a, a', and w.

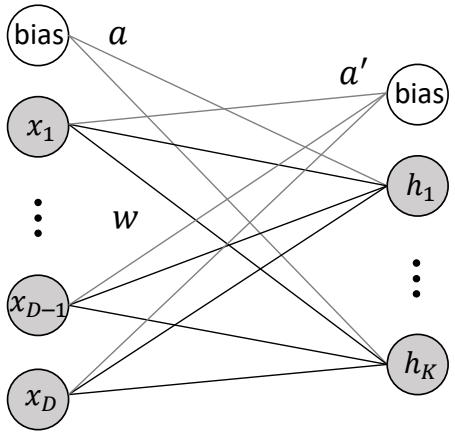


Figure 8.1: RBM structure

8.2 Large-scale Offloading Model

In this section, we consider a collaborative MEC and MCC network with one edge cloud, one central cloud and multiple mobile devices (MDs). Thus, the mobile device can execute its computational tasks locally, or offload its tasks to the edge cloud server through a wireless link and/or to the central cloud server through wireless and backhaul links.

8.2.1 System Model

Fig. 8.2 presents the system model, which consists of one edge cloud server, one central cloud server and multiple MDs, denoted by a set $\mathcal{N} = \{1, 2, \ldots, N\}$. The edge cloud server can be deployed into the base station, which is closer to the MDs. The MDs can communicate with the edge cloud with a wireless link, whereas the edge cloud and the central cloud can be interconnected through a wired link. Each MD has multiple tasks, denoted by a set $\mathcal{M} = \{1, 2, \cdots, M\}$. The size of the m-th task of the n-th MD is denoted by $w_{(n,m)}$. In each MD, these different tasks can choose to be processed locally or offloaded to the edge cloud server and the central cloud server. The offloading decision is represented by two binary variables $x^1_{(n,m)}$ and $x^2_{(n,m)}$.

On one hand, $x^1_{(n,m)} \in \{0, 1\}$ denotes the offloading decision for the m-th task, which means:

$$x^1_{(n,m)} = \begin{cases} 0, & \text{if task is executed locally} \\ 1, & \text{if task is offloaded} \end{cases} \tag{8.3}$$

where $x^1_{(n,m)} = 0$ denotes n-th MD decides to execute the m-th task locally, $x^1_{(n,m)} = 1$ indicates n-th MD decides to offload m-th task to the edge cloud server or central cloud server.

Once the m-th task is decided to be offloaded to the cloud server, $x^2_{(n,m)} \in \{0, 1\}$ represents the specific offloading destination for the m-th task, which means:

$$x^2_{(n,m)} = \begin{cases} 0, & \text{if offloaded to edge cloud \& } x^1_{(n,m)} = 1 \\ 1, & \text{if offloaded to central cloud \& } x^1_{(n,m)} = 1 \end{cases} \tag{8.4}$$

where $x^2_{(n,m)} = 0$ denotes the n-th MD decides to offload the m-th task to the edge cloud server, $x^2_{(n,m)} = 1$ denotes m-th task is offloaded to the central cloud server.

The detailed operations of a local computing model, edge computing model and cloud computing model are illustrated as follows, respectively. The important notations used in this chapter are listed in Table 8.1.

8.2.2 Local Computing Model

We first establish the local computing model when the task is decided to be executed locally. The task execution time of mobile device can be calculated as:

$$Tl_{(n,m)} = \frac{w_{(n,m)}}{f_l} \tag{8.5}$$

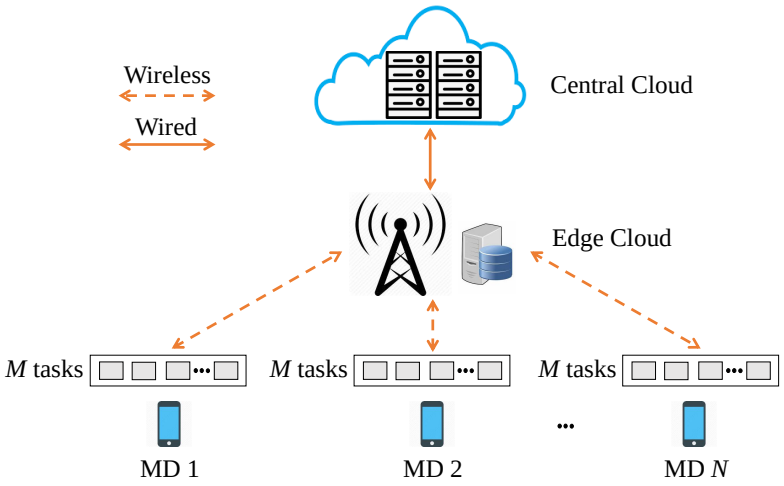


Figure 8.2: System model of computation offloading with heterogeneous cloud

where f_l denotes the task processing rate of local device.

The energy consumption for executing task m at the n-th device can be calculated as:

$$El_{(n,m)} = \theta_l w_{(n,m)} \tag{8.6}$$

where θ_l denotes the energy consumption per unit of workload of local device.

Therefore, the total time and energy consumption of the n-th MD can be expressed as:

$$Tl_{(n)} = \sum_{m=1}^{M} \left(1 - x^1_{(n,m)}\right) Tl_{(n,m)} \tag{8.7}$$

$$El_{(n)} = \sum_{m=1}^{M} \left(1 - x^1_{(n,m)}\right) El_{(n,m)} \tag{8.8}$$

8.2.3 Edge Computing Model

For the edge computing model, the MDs can communicate with the edge server via the cellular link. When the task is decided to be offloaded to the edge cloud, the MD needs to transmit the workload of the task to the edge cloud server and then to be processed. In general, the time and energy consumption are often neglected when the cloud servers return the computing results back to MDs, because the data size of feedback result is small [7].

Table 8.1: Important notations

Notation	Description
$w_{(n,m)}$	The m-th task workload of the n-th MD
$x^1_{(n,m)}$	$x^1_{(n,m)} = 0$ if m-th task is executed locally, $x^1_{(n,m)} = 1$ if m-th task is offloaded to the cloud
$x^2_{(n,m)}$	$x^2_{(n,m)} = 0$ if m-th task is offloaded to edge cloud, $x^2_{(n,m)} = 1$ if m-th task is offloaded to central cloud
$El_{(n,m)}$	The energy consumption of the m-th task of the n-th MD
θ_l	The local device energy consumption per unit of workload
$Tl_{(n,m)}$	The execution time of the m-th task of the n-th MD
f_l	The task processing rate of the MD
$T^e_{t\,(n,m)}$	The transmission time of offloading m-th task to edge cloud via wireless link
$b_{(n,e)}$	The bandwidth between n-th MD and edge cloud
$E^e_{t\,(n,m)}$	The energy consumption for transmission to the edge cloud
σ	The energy consumption per unit of workload for transmission to the edge cloud
$Te_{(n,m)}$	The time delay of offloading the m-th task of the n-th MD to the edge cloud
f_e	The task processing rate of the edge cloud
$Ee_{(n,m)}$	The energy consumption of offloading the m-th task of the n-th MD to the edge cloud
θ_e	The energy consumption per unit of workload of edge cloud
$T^c_{t\,(n,m)}$	The transmission time of offloading m-th task to central cloud via wireless link and wired link
$b_{(e,c)}$	The bandwidth between edge cloud and central cloud
$E^c_{t\,(n,m)}$	The energy consumption for transmission to the central cloud
β	The energy consumption per unit of workload for transmission to the central cloud
$Tc_{(n,m)}$	The time delay of offloading the m-th task of the n-th MD to the central cloud
f_c	The task processing rate of the central cloud
$Ec_{(n,m)}$	The energy consumption of offloading the m-th task of the n-th MD to the central cloud
θ_c	The energy consumption per unit of workload of central cloud

The transmission time for offloading the task to the edge cloud server can be calculated as:

$$T^e_{t\,(n,m)} = \frac{w_{(n,m)}}{b_{(n,e)}} \tag{8.9}$$

where $b_{(n,e)}$ denotes the bandwidth between the n-th MD and the edge server.

The energy consumption for the transmission can be given by:

$$E^e_{t\,(n,m)} = \sigma w\,(n,m) \tag{8.10}$$

where σ denotes the energy consumption per unit of workload for transmission to the edge cloud server.

After the task is transmitted to the edge cloud, it will be executed at the edge cloud server. The computation delay of the whole process can be expressed as:

$$Te_{(n,m)} = T^e_{t\,(n,m)} + \frac{w_{(n,m)}}{f_e} \tag{8.11}$$

where f_e denotes the task processing rate of the edge cloud server.

The energy consumption of whole process can be expressed as:

$$Ee_{(n,m)} = E^e_{t\,(n,m)} + \theta_e w_{(n,m)} \tag{8.12}$$

where θ_e denotes the energy consumption per unit of workload of edge server.

Therefore, the total time and energy consumption of the n-th MD can be expressed as:

$$Te_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} \left(1 - x^2_{(n,m)}\right) Te_{(n,m)} \tag{8.13}$$

$$Ee_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} \left(1 - x^2_{(n,m)}\right) Ee_{(n,m)} \tag{8.14}$$

8.2.4 Cloud Computing Model

For the cloud computing model, MDs can communicate with the central cloud via wireless and wired links. The central cloud servers can provide more powerful computing capacity for the MDs, but it might cause more delays for the transmission between MDs and the central cloud. The transmission time and energy consumption for offloading the task to the central cloud server can be calculated as:

$$T^c_{t\,(n,m)} = \frac{w_{(n,m)}}{b_{(n,e)}} + \frac{w_{(n,m)}}{b_{(e,c)}} \tag{8.15}$$

$$E^c_{t\,(n,m)} = \sigma w\,(n,m) + \beta w\,(n,m) \tag{8.16}$$

where $b_{(e,c)}$ denotes the bandwidth between edge cloud and central cloud. β denotes the energy consumption per unit of workload for transmission to the central cloud server.

After the task is transmitted to the central cloud, it will be executed at the central cloud server. The computation delay and energy consumption of the whole process can be expressed as:

$$Tc_{(n,m)} = T^c_{t\,(n,m)} + \frac{w_{(n,m)}}{f_c} \tag{8.17}$$

$$Ec_{(n,m)} = E^c_{t\,(n,m)} + \theta_c w_{(n,m)} \tag{8.18}$$

where f_c denotes the task processing rate of central cloud server, θ_c denotes the energy consumption per unit of workload of central cloud.

Therefore, the total time and energy consumption of the n-th MD can be expressed as:

$$Tc_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} x^2_{(n,m)} Tc_{(n,m)} \tag{8.19}$$

$$Ec_{(n)} = \sum_{m=1}^{M} x^1_{(n,m)} x^2_{(n,m)} Ec_{(n,m)} \tag{8.20}$$

8.2.5 Problem Formulation

The total computation time of executing all tasks can be given by:

$$T = \sum_{n=1}^{N} \max\left\{Tl_{(n)}, Te_{(n)}, Tc_{(n)}\right\} \tag{8.21}$$

The total energy consumption of executing all tasks can be given by:

$$E = \sum_{n=1}^{N} \left(El_{(n)} + Ee_{(n)} + Ec_{(n)}\right) \tag{8.22}$$

Considering a large number of mobile devices having different applications in the mobile environment, the offloading model is the large-scale MOP. To summarize, we establish a local-edge-cloud large-scale two-objective computation offloading optimization model.

8.3 The Proposed ELSMO

This section presents the details of two evolutionary large-scale sparse multi-objective optimization algorithms (called ELSMO) for solving the large-scale computation offloading problems.

8.3.1 General Framework

The general frameworks of two proposed algorithms are presented in Algorithm 9 and Algorithm 10, respectively. In both two algorithms, the non-dominated front number and crowding distance are calculated the same way as in NSGA-II [24], which are also the selection criteria in the environmental selection. In ELSMO-1, the non-dominated solutions in the current population are used to train a RBM, then the offspring solutions are generated from the mating pool. The two parameters ρ and K are updated iteratively. In ELSMO-2, the population initialization is different from ELSMO-1,

which analyzes the contribution score of each decision variable. And the new offspring generation operation is conducted based on the score. More details about the main operations in ELSMO-1 and ELSMO-2 are presented in the following sections.

Algorithm 9: Framework of ELSMO-1

Input: The population size $\widetilde{N}$
Output: The final population P

$P \leftarrow Initialization\left(\widetilde{N}\right)$;
$[F_1, F_2, \cdots] \leftarrow NondominatedSorting\,(P)$;
$CrowdDis \leftarrow CrowdingDistance\,(F_1, F_2, \cdots)$;
$\rho \leftarrow 0.5$; // Ratio of offspring solutions generated in the different search subspace
$K \leftarrow N$; // Size of the hidden layer
while $termination\ criterion\ not\ fulfilled$ **do**
    Train a RBM with K hidden neurons based on non-dominated solutions in P;
    $P' \leftarrow$ Select $\widetilde{N}$ parents via binary tournament selection in P ;
    $O \leftarrow Offspring Generation\,(P, P', \rho, K, RBM)$;
    $P \leftarrow P \cup O$;
    Delete duplicated solutions from P;
    $P \leftarrow EnvironmentalSelection\,(P)$;
    $[\rho, K] \leftarrow UpdateParameter\,(P, \rho)$;
end

8.3.2 The Proposed ELSMO-1

Offspring Generation

Before generating offspring solutions, all the non-dominated solutions in the population are used to train an RBM via the contrastive divergence algorithm. As shown in Fig. 8.3, the RBM can be used to reduce the dimensionality of the original binary vectors, and the reduced binary vectors can be also recovered. Since the non-dominated solutions are used to train the RBM, it can be seen that each solution can be mapped between the Pareto optimal space and the original search space.

After obtaining the trained RBM, the binary tournament selection mechanism is used to select $\widetilde{N}$ parents as the mating pool based on the non-dominated front number and crowding distance. Then two parents are randomly selected from the mating pool and single-point crossover and bitwise mutation operation are used to generate two offspring solutions. The ratio ρ determines the probability that the offspring solutions will be generated in the Pareto-optimal subspace by the trained RBM or generated in the original search space. Specifically, if the probability ρ is larger than a random value

Algorithm 10: Framework of ELSMO-2

Input: The population size $\widetilde{N}$
Output: The final population P

$[P, Score] \leftarrow Initialization\left(\widetilde{N}\right)$;
$[F_1, F_2, \cdots] \leftarrow NondominatedSorting\,(P)$;
$CrowdDis \leftarrow CrowdingDistance\,(F_1, F_2, \cdots)$;
while $termination\ criterion\ not\ fulfilled$ **do**
    $P' \leftarrow$ Select $2\widetilde{N}$ parents via binary tournament selection in P ;
    $O \leftarrow OffspringGeneration\,(P', Score)$;
    $P \leftarrow P \cup O$;
    Delete duplicated solutions from P;
    $P \leftarrow EnvironmentalSelection\,(P)$;
end

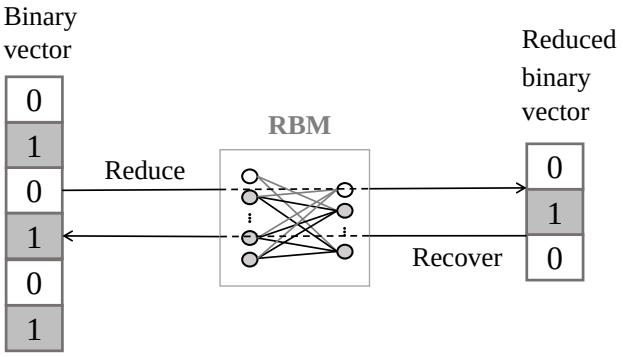


Figure 8.3: Reduce and recover of solutions

in $[0, 1]$, the binary vectors of parents are reduced by Equation 8.1 and the offspring solutions are generated in the Pareto-optimal space, then the reduced offspring solutions are recovered by Equation 8.2. If the probability ρ is smaller than the random value, the offspring solutions are generated in the original search space without RBM.

Update Parameter

In ELSMO-1, there are two parameters needed to be considered, the ratio of offspring solutions generated in the Pareto-optimal subspace or original search space ρ and the size of the hidden layer

K. The parameter ρ is updated iteratively, which is defined as follows:

$$\rho_{t+1} = 0.5 \times \left(\rho_t + \frac{s_{1,t} + 1}{s_{1,t} + s_{2,t} + 1} \right) \tag{8.23}$$

where ρ_t is the value of ρ at the t-th generation and $\rho_0 = 0.5$. $s_{1,t}$ and $s_{2,t}$ denotes the number of successful offspring solutions generated in the Pareto-optimal subspace and in the original search space, respectively. A successful solution means that it survives to the next population.

The sparsity of non-dominated solutions in the current population reflects the setting value of K. Let dec be a binary vector denoting whether each variable should be nonzero, the probability of setting dec_i to 1 is defined according to the non-dominated solution set NP:

$$p\left(dec_i = 1 \,|NP\right) = \frac{1}{|NP|} \sum_{x \in NP} |sign\left(x_i\right)| \tag{8.24}$$

where if $x_i = 0$ then $|sign\left(x_i\right)|$ equals 0 or $|sign\left(x_i\right)|$ is equal to 1 otherwise. Through comparing the probability $p\left(dec_i = 1 \,|NP\right)$ with a uniformly distributed random value in $[0, 1]$, the value of dec_i is obtained. Then the parameter K is defined as follows:

$$K = \sum dec_i \tag{8.25}$$

8.3.3 The Proposed ELSMO-2

Population Initialization

In ELSMO-2, the population initialization process includes two steps, i.e., calculating the scores of decision variables and generating the initial population. First, the population Q is constituted by a $D * D$ identity matrix, where D denotes the number of decision variables. Then the non-dominated front numbers of the solutions in population Q are calculated based on the non-dominated sorting method. The non-dominated front number of the i-th solution in Q can be regarded as the score of the i-th decision variable. The smaller score means the better quality of the decision variable, so the value of the decision variable is set to 1 with a higher probability.

Afterwards, let a binary vector $mask$ represent the offloading decision. According to the scores of decision variables, the binary tournament selection mechanism is used to select the element in $mask$ with a better score and set the element to 1. In each solution, the number of $rand\left(\right) \times D$ elements are selected to be set to 1 and other elements in $mask$ are set to 0, where $rand\left(\right)$ denotes a uniformly distributed random value in $[0, 1]$. In this way, the initialized population with better

convergence and diversity is obtained by selecting decision variables with good quality.

Offspring Generation

The binary tournament selection mechanism is used to select $2\widetilde{N}$ parents as the mating pool based on the non-dominated front number and crowding distance. Then two parents p and q are randomly selected from the mating pool to generate an offspring solution o each time. The binary vector $mask$ of o is first set to the same to p, then two decision variables from the nonzero elements in $p.mask \cap q.\overline{mask}$ are randomly selected with probability 0.5, the decision variable in the $mask$ of o with a larger score is set to 0. Otherwise, selecting two decision variables from nonzero elements in $p.\overline{mask} \cap q.mask$, the decision variable in the $mask$ of o with a smaller score is set to 1.

Afterwards, one mutation operation is conducted on the $mask$ of o to retain diversity. Compare one random probability distributed in $[0, 1]$ with 0.5, if the probability is less than 0.5, randomly select two decision variables from the nonzero elements in $o.mask$, and set the element with a large score in $o.mask$ to 0. Otherwise, randomly select two decision variables from the nonzero elements in $o.\overline{mask}$, and set the element with a small score in $o.mask$ to 1. The main idea of the mutation operation is making decision variables with better quality approach to 1, while decision variables with worse quality approach 0.

8.3.4 Computational Complexity

For the proposed algorithms, the major costs are the iteration process in Algorithm 9 and Algorithm 10. In ELSMO-1, Step 7 needs $O\left(\widetilde{N}EDK\right)$ operations to train the RBM, where $\widetilde{N}$ is the population size, E is the number of epochs for training, D is the number of decision variables, K is the hidden layer size. Step 8 needs $O\left(\widetilde{N}\right)$ operations for the binary tournament selection. Step 9 performs $O\left(\widetilde{N}DK\right)$ to generate offspring solutions. Step 12 performs $O\left(\widetilde{M}\widetilde{N}^2\right)$ operations for the environmental selection, where $\widetilde{M}$ is the number of objectives. Step 13 needs $O\left(\widetilde{N}D\right)$ operations to update the parameters. To summarize, the overall computational complexity at one generation of ELSMO-1 is $\max\left\{O\left(\widetilde{N}EDK\right), O\left(\widetilde{M}\widetilde{N}^2\right)\right\}$. In ELSMO-2, Step 5 performs $O\left(2\widetilde{N}\right)$ operations for selecting mating pool. Step 6 needs $O\left(\widetilde{N}D\right)$ operations to generate offspring solutions. Step 9 performs $O\left(\widetilde{M}\widetilde{N}^2\right)$ operations for the environmental selection. The overall computational complexity at one generation of ELSMO-2 is $\max\left\{O\left(\widetilde{M}\widetilde{N}^2\right), O\left(\widetilde{N}D\right)\right\}$.

8.4 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms and demonstrate the compared results of different MOEAs and offloading strategies.

8.4.1 Experimental Settings

In the experiment, we set up the local-edge-cloud offloading environment. The number of mobile devices is selected between 100 and 1000. The number of independent tasks of each MD is $M = 5$. So the dimension of the offloading problem D is between 1000 and 10000. We set the energy consumption per unit of workloads in local device, edge and central cloud server $\theta_l = 3\mathrm{J/MB}$, $\theta_e = 1.5\mathrm{J/MB}$, and $\theta_c = 1\mathrm{J/MB}$, respectively. The processing rates of local device, edge and central cloud server are $f_l = 2\mathrm{MB/s}$, $f_e = 8\mathrm{MB/s}$, and $f_c = 12\mathrm{MB/s}$, respectively. The energy consumption per unit of workloads for transmission from local device to edge cloud server is a random value within $[0.4, 0.6]$ J/MB, and the same measure from edge to central cloud server is a random value within $[0.3, 0.5]$ J/MB. In addition, the bandwidth between the local device and edge server is chosen from $[80, 100]$ Mbps, whereas the bandwidth between edge cloud and central cloud is fixed $b_{(e,c)} = 150\mathrm{Mbps}$. We assume that the workloads of tasks are randomly distributed between 10MB and 30MB. The related parameters and corresponding values are summarized in Table 8.2.

Table 8.2: Parameter values

Parameter	Value
The number of mobile devices	$N = [100, 1000]$
The number of independent tasks of each MD	$M = 5$
The local energy consumption per unit	$\theta_l = 3\mathrm{J/MB}$
The edge cloud energy consumption per unit	$\theta_e = 1.5\mathrm{J/MB}$
The central cloud energy consumption per unit	$\theta_c = 1\mathrm{J/MB}$
The processing rate of the local device	$f_l = 2\mathrm{MB/s}$
The processing rate of the edge cloud server	$f_e = 8\mathrm{MB/s}$
The processing rate of the central cloud server	$f_c = 12\mathrm{MB/s}$
The energy consumption per unit for transmission from MD to edge cloud	$\sigma = [0.4, 0.6]\,\mathrm{J/MB}$
The energy consumption per unit for transmission from edge to central cloud	$\beta = [0.3, 0.5]\,\mathrm{J/MB}$
The bandwidth between n-th MD and edge cloud	$b_{(n,e)} \in [80, 100]\,\mathrm{Mbps}$
The bandwidth between edge and central cloud	$b_{(e,c)} = 150\mathrm{Mbps}$
The workloads of all tasks	$w_{(n,m)} \in [10, 30]\,\mathrm{MB}$

To verify the performance of the proposed algorithms, we compare the proposed algorithms with other MOEAs and offloading schemes to solve ten different large-scale offloading problems, which

means the number of devices N = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000] and the dimension D = [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]. The compared algorithms are NSGA-II [24], SPEA2 [124], SMS-EMOA [4], and EAG-MOEA/D [10]. NSGA-II, SPEA2, SMS-EMOA are three classical MOEAs which are effective for MOPs, while EAG-MOEA/D is tailored for combinatorial MOPs. For a fair comparison, the population size of all algorithms is set to 50. The number of function evaluations is set with values in the interval from 6.0×10^4 to 15×10^4 for ten test problems. In NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D and ELSMO-1, the single-point crossover and bitwise mutation are applied to generate new offspring, where the probabilities of crossover and mutation are set to 1.0 and $1/D$, respectively. The hypervolume (HV) [125] is adopted as the metric to evaluate the performance of the compared algorithms. For each test instance, each algorithm is executed 30 times independently, and the average and standard deviation of the metric values are recorded. The Wilcoxon rank sum test at a 5% significance level is used to compare the experimental results, where the symbol '$+$', '$-$' and '$\approx$' denotes that the result of another algorithm is significantly better, significantly worse and similar to that obtained by proposed algorithm.

8.4.2 Comparison with Other MOEAs

Table 8.3 presents the HV metric values obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on ten test problems. The proposed ELSMO-2 has achieved the best performance on 9 of 10 test instances, while only EAG-MOEA/D gets 1 of 10 best results for the rest of compared algorithms. When the dimension is not so large (i.e., 1000), the EAG-MOEA/D and ELSMO-2 can obtain similar metric values. It can be observed that ELSMO-2 has a clear advantage over other compared algorithms with the increment of dimension.

Figs. 8.4, 8.5 and 8.6 show the final non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on offloading problems with 1000, 5000, and 10000 binary variables. NSGA-II, SPEA2, SMS-EMOA can only get a small part of the solutions in the Pareto front, which may be worse when dealing with large-dimension problems. EAG-MOEA/D is designed for combinatorial problems, which can obtain good performance compared with other classical algorithms NSGA-II, SPEA2 and SMS-EMOA, whereas its diversity still encounters difficulties for solving large-dimensional offloading problems. ELSMO-1 seems to have the best performance of diversity compared with the other five algorithms, but the RBM may need more iterations to be trained for improving the convergence. It is clear from the figures that ELSMO-2 can always get better performance between convergence and diversity no matter what the dimensional offloading problem is.

Table 8.3: The HV values obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on offloading problems

Problem	D	NSGA-II	SPEA2	SMS-EMOA
Offloading1	1000	2.7834e-1 (1.60e-3) $-$	2.7353e-1 (1.75e-3) $-$	2.6961e-1 (1.64e-3) $-$
Offloading2	2000	2.6303e-1 (1.80e-3) $-$	2.5928e-1 (1.20e-3) $-$	2.5669e-1 (9.36e-4) $-$
Offloading3	3000	2.5533e-1 (1.70e-3) $-$	2.5192e-1 (9.69e-4) $-$	2.5123e-1 (1.44e-3) $-$
Offloading4	4000	2.5263e-1 (1.10e-3) $-$	2.4856e-1 (7.96e-4) $-$	2.4805e-1 (7.14e-4) $-$
Offloading5	5000	2.5009e-1 (1.09e-3) $-$	2.4757e-1 (6.41e-4) $-$	2.4656e-1 (8.32e-4) $-$
Offloading6	6000	2.4780e-1 (6.85e-4) $-$	2.4587e-1 (6.99e-4) $-$	2.4483e-1 (4.67e-4) $-$
Offloading7	7000	2.4759e-1 (1.04e-3) $-$	2.4493e-1 (1.59e-4) $-$	2.4391e-1 (2.20e-4) $-$
Offloading8	8000	2.4542e-1 (6.92e-4) $-$	2.4373e-1 (7.42e-4) $-$	2.4254-1 (6.27e-4) $-$
Offloading9	9000	2.4439e-1 (4.68e-4) $-$	2.4325e-1 (3.76e-4) $-$	2.4218-1 (7.11e-4) $-$
Offloading10	10000	2.4434e-1 (3.23e-4) $-$	2.4267e-1 (5.62e-4) $-$	2.4212-1 (3.56e-4) $-$
$+/-/\approx$		0/10/0	0/10/0	0/10/0
Problem	D	EAG-MOEA/D	ELSMO-1	ELSMO-2
Offloading1	1000	2.9834e-1 (3.45e-4) $+$	2.9541e-1 (8.36e-4) $-$	2.9800e-1 (8.96e-5)
Offloading2	2000	2.9564e-1 (9.80e-4) $-$	2.6861e-1 (2.92e-2) $-$	2.9761e-1 (1.73e-4)
Offloading3	3000	2.8934e-1 (3.76e-3) $-$	2.8580e-1 (2.07e-3) $-$	2.9442e-1 (3.48e-4)
Offloading4	4000	2.8371e-1 (3.79e-3) $-$	2.7427e-1 (1.89e-2) $-$	2.9761e-1 (1.73e-4)
Offloading5	5000	2.8441e-1 (1.35e-3) $-$	2.7836e-1 (3.87e-3) $-$	2.9161e-1 (3.14e-4)
Offloading6	6000	2.8219e-1 (2.73e-3) $-$	2.7564e-1 (2.51e-3) $-$	2.9018e-1 (2.00e-4)
Offloading7	7000	2.7632e-1 (5.08e-3) $-$	2.6431e-1 (1.77e-2) $-$	2.8837e-1 (5.94e-4)
Offloading8	8000	2.7476-1 (5.20e-3) $-$	2.7134e-1 (1.67e-3) $-$	2.8626e-1 (8.37e-4)
Offloading9	9000	2.7607e-1 (3.52e-3) $-$	2.6904e-1 (2.81e-3) $-$	2.8521e-1 (3.74e-4)
Offloading10	10000	2.7024e-1 (7.04e-4) $-$	2.7013e-1 (2.49e-3) $-$	2.8359e-1 (1.17e-3)
$+/-/\approx$		1/9/0	0/10/0	

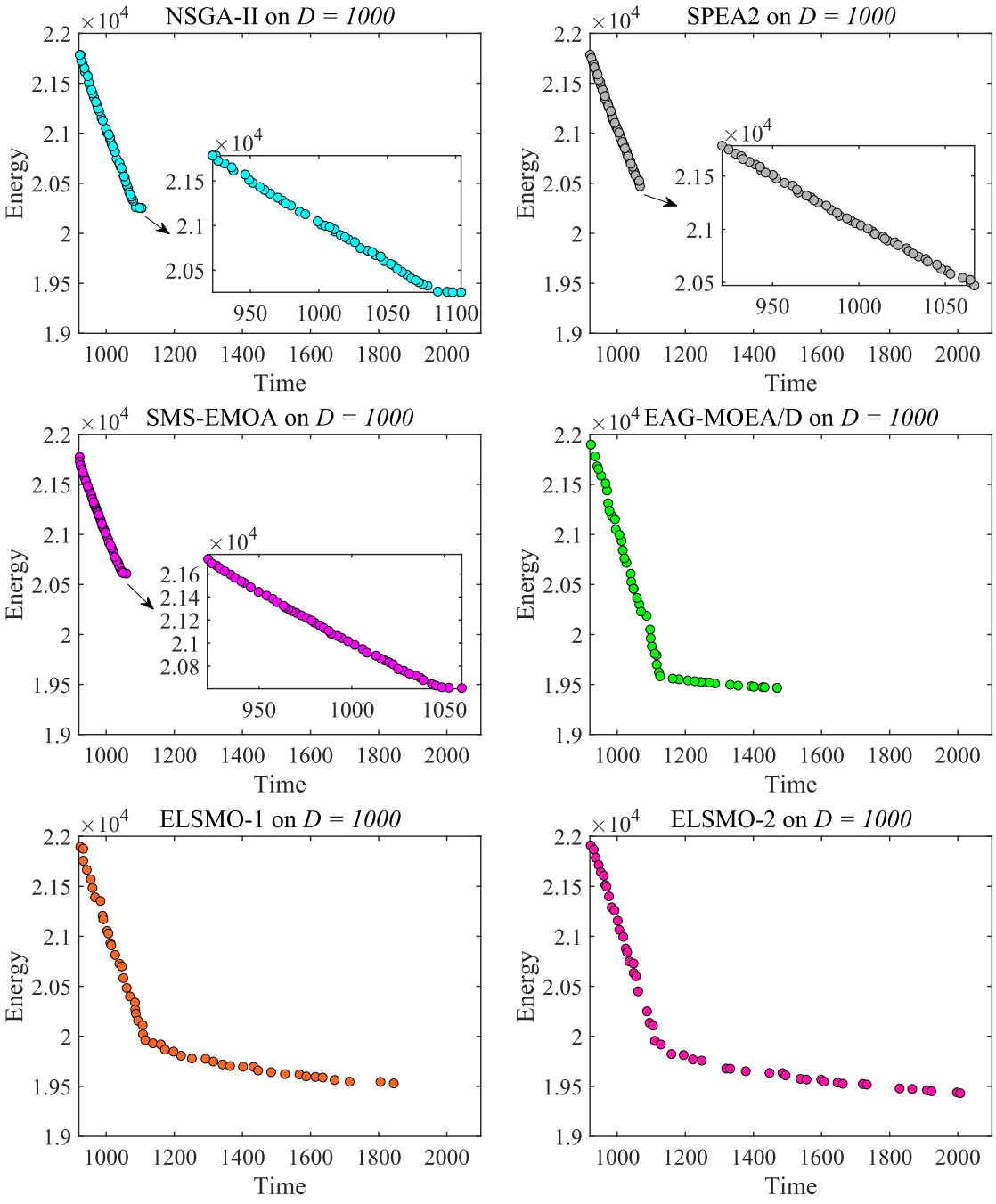


Figure 8.4: The non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on 1000-dimensional offloading problem

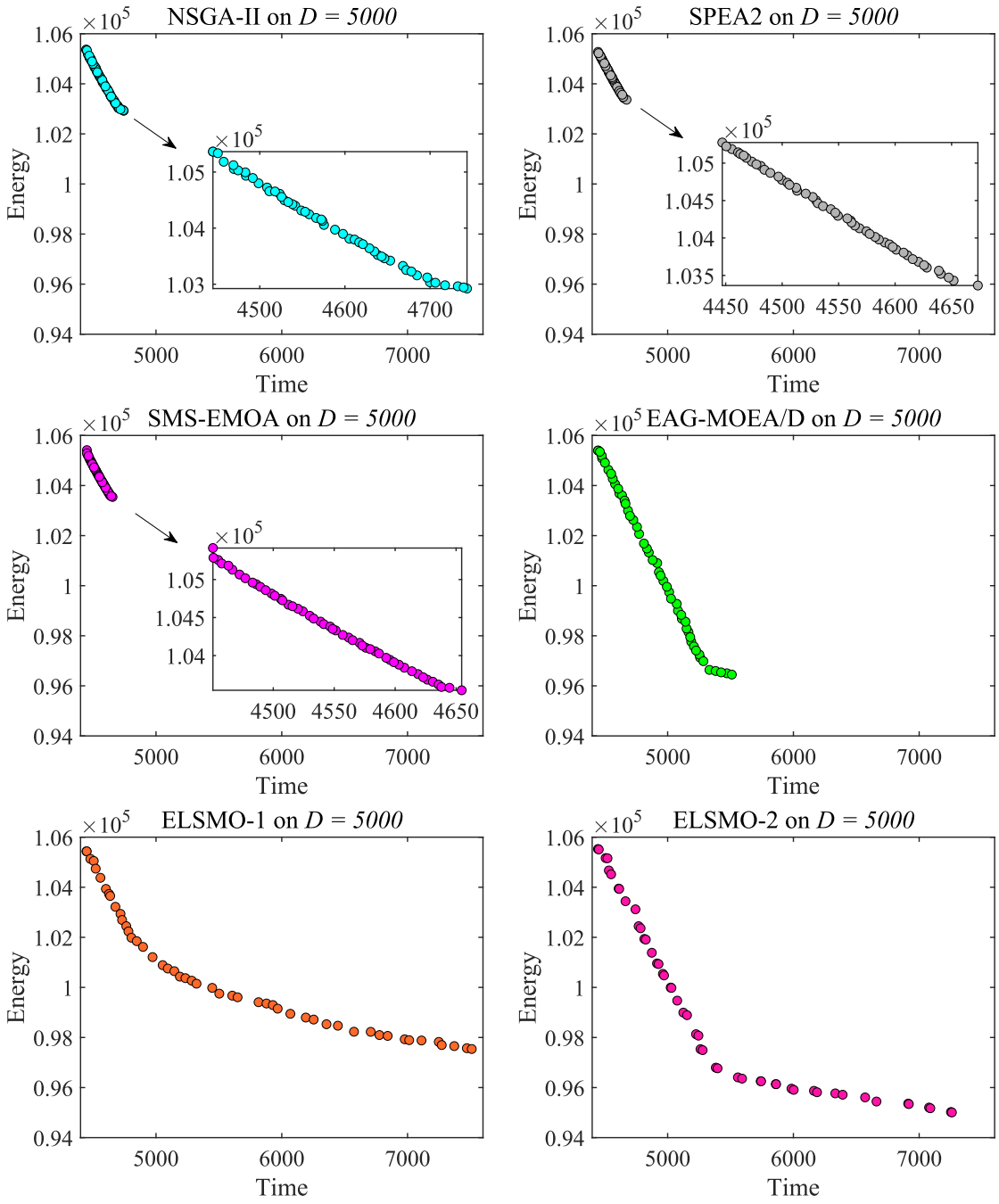


Figure 8.5: The non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on 5000-dimensional offloading problem

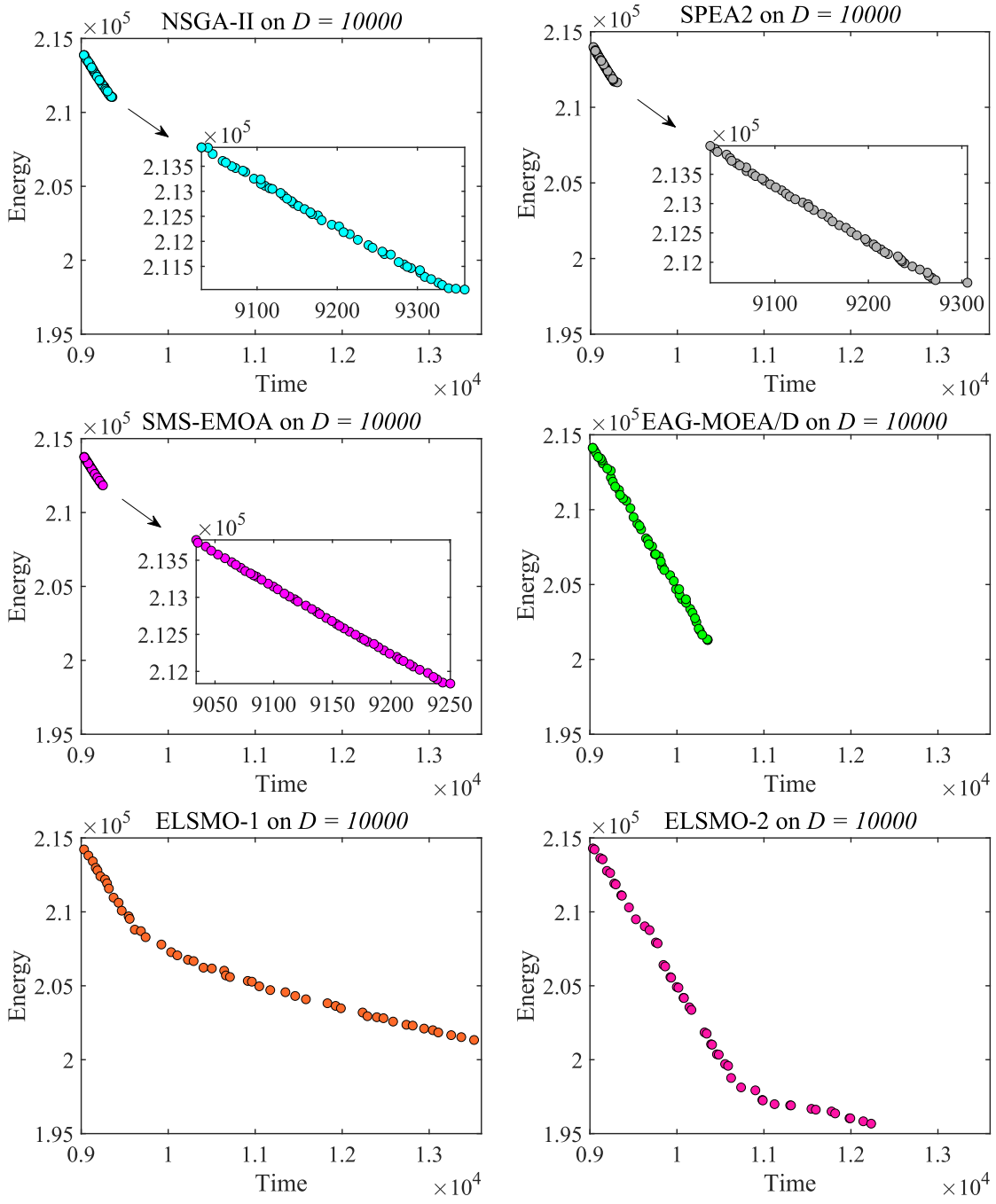
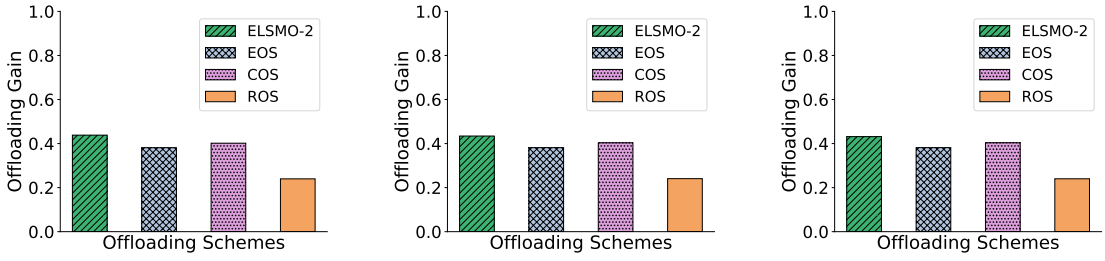


Figure 8.6: The non-dominated solution set with the medium HV value obtained by NSGA-II, SPEA2, SMS-EMOA, EAG-MOEA/D, ELSMO-1 and ELSMO-2 on 10000-dimensional offloading problem

8.4.3 Comparison with Other Offloading Schemes

It has been observed that ELSMO-2 can get the best non-dominated solution set in the Pareto front. The non-dominated solution set can give the decision-maker more choices. To further validate the performance of ELSMO-2, we apply the offloading gain (Equation 7.23) to compare ELSMO-2 with the other four offloading schemes, which are LOS, EOS, COS and ROS in Section 7.2.3. According to the different quality of service, the decision-maker may set different weights of w for the tradeoff between time and energy. If the decision-maker is sensitive to the time consumption, it may set a larger weight for the time consumption, or if the decision-maker focus is on energy performance, it may set a larger weight for the energy consumption.

Figs. 8.7, 8.8 and 8.9 present the offloading gain of different offloading schemes under the different weights on 1000-, 5000-, 10000-dimensional offloading problems. It can be seen that offloading is always beneficial compared with the only local offloading scheme. And the proposed ELSMO-2 can always get the best offloading gain compared with other offloading schemes under different weights on different large-scale offloading problems. On the other hand, ELSMO-2 takes an obvious advantage over other offloading schemes when w is becoming larger, which means that ELSMO-2 can reduce the time delay more efficiently. What's more, compared with ROS, EOS and COS can obtain a better offloading gain, which also demonstrates that edge and cloud offloading can improve performance for both time and energy consumption.

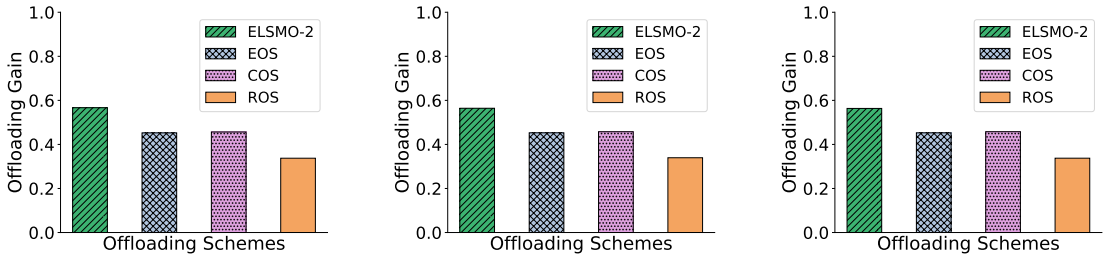


(a) $D = 1000$ (b) $D = 5000$ (c) $D = 10000$

Figure 8.7: Offloading gain of different offloading schemes for $w = 0.2$

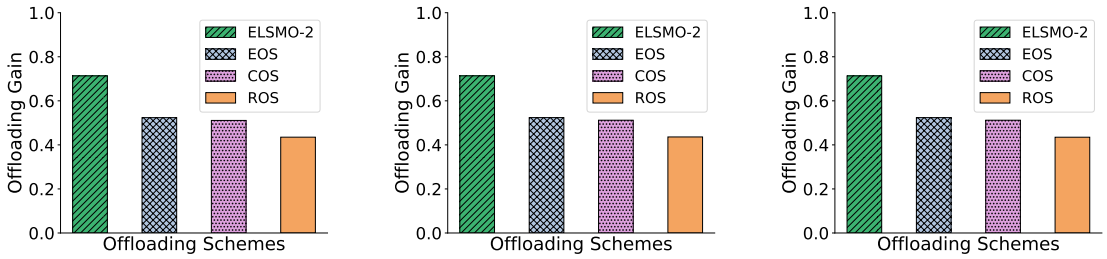
8.5 Summary

In this chapter, two evolutionary large-scale sparse multi-objective optimization (ELSMO) algorithms have been proposed and compared for solving heterogeneous edge-cloud computation of-



(a) $D = 1000$ (b) $D = 5000$ (c) $D = 10000$

Figure 8.8: Offloading gain of different offloading schemes for $w = 0.5$



(a) $D = 1000$ (b) $D = 5000$ (c) $D = 10000$

Figure 8.9: Offloading gain of different offloading schemes for $w = 0.8$

floading problems. Taking into account the large-scale and sparsity properties of the multi-objective offloading model, the RBM is used to reduce the dimensionality and learn from the Pareto optimal subspace. The contribution score is applied to select better decision variables to generate offspring solutions. The proposed algorithms are compared with other MOEAs and offloading schemes to solve the test problems under different scales, the experimental results have demonstrated the effectiveness and efficiency of the proposed algorithms.

Chapter 9

Dynamic and Secure Multi-objective Offloading

In this chapter, we design a novel multi-objective computation offloading evolutionary algorithm (called MCOEA) to solve a dynamic multi-objective offloading problem considering compression, security and mobility.

In the computation offloading process of MCC and MEC, the mobility and security of IoT devices play significant roles in offloading decision making. At first, we establish a multi-server multi-user multi-task multi-objective computation offloading model in IoT-edge-cloud computing networks, with the consideration of compression, security and mobility, which aims to minimize response time and energy consumption of the IoT devices. The compression strategy is used to compress the large data size, and the security layer is adopted to encrypt the private data, as well as the mobility influence is analyzed in dynamic computation offloading.

For solving the proposed model, a novel multi-objective computation offloading evolutionary algorithm (MCOEA) is developed. In MCOEA, a tailored binary crossover operator is designed to improve the convergence, and a hybrid mutation operator is employed to retain the diversity of the solutions. We compare the proposed MCOEA with the other five representative multi-objective evolutionary algorithms (MOEAs) as well as four offloading schemes to solve ten different computation offloading problems. In addition, we analyze the advantages and influence of compression, security and mobility. Furthermore, the impact of different system parameters in IoT-edge-cloud networks are studied with regard to computation offloading performance. The experimental results demonstrate the superiority and efficiency of the proposed algorithms.

9.1 Dynamic and Secure Offloading Model

In this section, we first introduce a collaborative IoT-edge-cloud computing network and then formulate the computation offloading decision model related to task offloading, compression, security and mobility.

9.1.1 System Overview

In order to take advantage of MEC and MCC, we try to leverage heterogeneous computing resources and consider a collaborative MEC and MCC network with multiple IoT devices, multiple edge servers and one cloud server. Fig. 9.1 presents a task offloading framework in the IoT-edge-cloud computing network. The MEC servers can be deployed to the base stations and MCC server can be placed at the central cloud data center. Each IoT device can communicate with edge server through the wireless network while edge server and cloud server are connected with wired links. In addition, the different MEC servers can communicate with each other through wired links. The IoT device has different computing tasks, which can be executed locally at IoT devices or offloaded to the MEC servers and MCC servers.

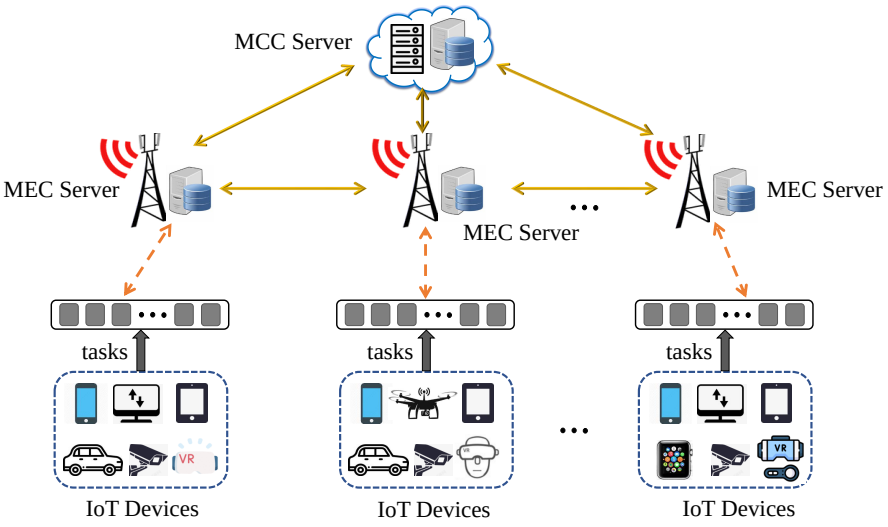


Figure 9.1: A task offloading framework in the IoT-edge-cloud computing network

The IoT-edge-cloud computing model consists of N IoT devices, K MEC servers and one MCC server. In general, we assume that each IoT device has M independent tasks. We denote the set of IoT devices as $\mathcal{N} = \{1, 2, \ldots, N\}$ and the set of tasks as $\mathcal{M} = \{1, 2, \cdots, M\}$. These different

computing tasks can be allocated to IoT devices, MEC or MCC servers. We use two binary variables a_{nm}^1 and a_{nm}^2 to represent the offloading decision of each task [100], which are defined as:

$$a_{nm}^1 = \begin{cases} 0, & \text{if task is executed locally} \\ 1, & \text{if task is offloaded} \end{cases} \tag{9.1}$$

$$a_{nm}^2 = \begin{cases} 0, & \text{if offloaded to MEC server \& } a_{nm}^1 = 1 \\ 1, & \text{if offloaded to MCC server \& } a_{nm}^1 = 1 \end{cases} \tag{9.2}$$

We try to find the optimal offloading decision to make IoT devices reduce response time and energy consumption. The detailed operations of the task offloading decision model as well as compression, security, and mobility are illustrated as follows. Table 9.1 summarizes the key notations and their definitions in this chapter.

9.1.2 Offloading Decision Model

We intend to establish the task offloading decision model to minimize the completion time and energy consumption of IoT devices.

IoT Computing Model

For an IoT computing model, each computing task is defined as a tuple $(\alpha_{nm}, \gamma_{nm})$, where α_{nm} is the data size of m-th task of IoT n and γ_{nm} is the required number of CPU cycles to finish the task. Due to the limited computing resources and battery life, IoT devices can perform lightweight tasks.

The execution time of task m on IoT device can be calculated as:

$$T_{nm}^{IoT\_exec} = \frac{\gamma_{nm}}{f_{IoT}} \tag{9.3}$$

where f_{IoT} is the computing frequency of the IoT device.

Similarly, the energy consumption of the IoT device for executing the task m is formulated as:

$$E_{nm}^{IoT\_exec} = p_{ex} \times T_{nm}^{IoT\_exec} \tag{9.4}$$

where p_{ex} is the CPU power of the IoT device.

Table 9.1: Summary of notations

Notation	Description
a_{nm}^1	a_{nm}^1 represents that task m of IoT device n is executed locally or offloaded
a_{nm}^2	a_{nm}^2 represents that task m of IoT device n is offloaded to MEC or MCC server
α_{nm}	Input data size of the task m of IoT device n
β_{nm}	The total data size of the task m of IoT device n for offloading after compression
γ_{nm}	Total CPU cycles of the task m of IoT device n
p_{ex}	The CPU power of the IoT device
p_{tr}	The transmission power of the IoT device
B_{lan}, B_{wan}	The bandwidth of LAN and WAN
f_{IoT}, f_{edge}, f_{cloud}	The computing capability of the IoT device, MEC server and MCC server
$T_{nm}^{IoT\_exec}$	The execution time for processing task m on the IoT device
$T_{nm}^{edge\_exec}$	The execution time for processing task m on the MEC server
$T_{nm}^{cloud\_exec}$	The execution time for processing task m on the MCC server
$T_{nm}^{edge\_tr}$	The transmission time for offloading task m to the MEC server
$T_{nm}^{cloud\_tr}$	The transmission time for offloading task m to the MCC server
$E_{nm}^{IoT\_exec}$	The energy consumption for processing task m on the IoT device
$E_{nm}^{edge\_tr}$	The energy consumption for transmitting the task m to the MEC server
$E_{nm}^{cloud\_tr}$	The energy consumption for transmitting the task m to the MCC server
$T_{nm}^{edge\_tr+exec}$	The total computation time for processing the task m on the MEC server
$T_{nm}^{cloud\_tr+exec}$	The total computation time for processing the task m on the MCC server
x_{nm}	Compression decision of the task m of the IoT device n
y_{nm}	Security decision of the task m of the IoT device n
z_n	Mobility decision of the IoT device n
σ_{nm}	CPU cycles required to compress the data size of the task m
η_{nm}	CPU cycles required to decompress the data size of the task m
$T_{nm}^{edge\_compr+decompr}$	The compression and decompression time of the task m on the IoT device and MEC server
$T_{nm}^{cloud\_compr+decompr}$	The compression and decompression time of the task m on the IoT device and MCC server
E_{nm}^{compr}	The energy consumption for compressing the task m on the IoT device
$T_{nm}^{edge\_tr\_compr}$	The transmission time for offloading task m to the MEC server after compression
$T_{nm}^{cloud\_tr\_compr}$	The transmission time for offloading task m to the MCC server after compression
$E_{nm}^{tr\_compr}$	The energy consumption for transmitting the task m to the MEC or MCC server after compression
q_{nm}	CPU cycles required to encrypt the data size of the task m
d_{nm}	CPU cycles required to decrypt the data size of the task m
$T_{nm}^{edge\_enc+dec}$	The encryption and decryption time of the task m on the IoT device and MEC server
$T_{nm}^{cloud\_enc+dec}$	The encryption and decryption time of the task m on the IoT device and MCC server
E_{nm}^{enc}	The energy consumption for encrypting the task m on the IoT device
τ_n	The mobility delay of the IoT device n
$T_n^{mob\_edge}$	The mobility delay of the IoT device n on the MEC server
$T_n^{mob\_cloud}$	The mobility delay of the IoT device n on the MCC server
T_{nm}^{IoT}, T_{nm}^{edge}, T_{nm}^{cloud}	The total time consumption for processing the task m on the IoT device, MEC server and MCC server
E_{nm}	The total energy consumption for processing the task m of the IoT device n
T_n	The completion time for processing all M tasks of IoT device n
T	Total completion time for processing all tasks of all IoT devices
E_n	The energy consumption for processing all M tasks of IoT device n
E	Total energy consumption for processing all tasks of all IoT devices

Edge Computing Model

For an edge computing model, IoT devices can communicate with the MEC servers through wireless communication technologies. The tasks can be offloaded to and then processed on MEC servers. The energy consumption and delay are neglected for transmitting the results from MEC servers to IoT devices. This is due to the fact that the size of output result is generally much smaller than input data [104, 107].

By using a Local Area Network (LAN), the transmission time for offloading the task m to the MEC server can be expressed as:

$$T_{nm}^{edge\_tr} = \frac{\alpha_{nm}}{B_{lan}} \tag{9.5}$$

where B_{lan} is the bandwidth of LAN between IoT devices and MEC servers.

The energy consumption for the transmission can be calculated as:

$$E_{nm}^{edge\_tr} = p_{tr} \times T_{nm}^{edge\_tr} \tag{9.6}$$

where p_{tr} is the transmission power of IoT device.

The execution time of processing the task m on the MEC server can be given by:

$$T_{nm}^{edge\_exec} = \frac{\gamma_{nm}}{f_{edge}} \tag{9.7}$$

where f_{edge} is the computing frequency of the MEC server.

Hence, the total computation time for processing the task m on the MEC server can be expressed by:

$$T_{nm}^{edge\_tr+exec} = T_{nm}^{edge\_tr} + T_{nm}^{edge\_exec} \tag{9.8}$$

Cloud Computing Model

For a cloud computing model, IoT devices can communicate with MCC servers with wireless and wired links. MCC servers can provide more powerful computing capabilities than the IoT devices and MEC servers. By taking advantage of MCC via Wide Area Network (WAN), the transmission time and energy consumption for offloading the task m to MCC server can be calculated by:

$$T_{nm}^{cloud\_tr} = \frac{\alpha_{nm}}{B_{wan}} \tag{9.9}$$

$$E_{nm}^{cloud\_tr} = p_{tr}\frac{\alpha_{nm}}{B_{lan}} \tag{9.10}$$

where B_{wan} is the bandwidth of WAN between IoT devices and MCC servers. It is noted that energy consumption for transmitting the task to MCC server only covers the transmission process between IoT devices and MEC server, since we only focus on the energy consumption of IoT devices.

The time consumption for executing the task m on MCC server can be expressed as:

$$T_{nm}^{cloud\_exec} = \frac{\gamma_{nm}}{f_{cloud}} \tag{9.11}$$

where f_{cloud} is the computing frequency of the MCC server.

Hence, the total computation time for processing the task m on the MCC server can be expressed by:

$$T_{nm}^{cloud\_tr+exec} = T_{nm}^{cloud\_tr} + T_{nm}^{cloud\_exec} \tag{9.12}$$

9.1.3 Compression

More and more IoT devices produce ever-increasing amounts of data which could be offloaded to MEC or MCC servers through the wireless channel. Multiple IoT devices will share the radio resource and the wireless network may not always be stable and abundant for IoT devices. The large data size of tasks could cause the communication overhead to exceed the benefits of offloading. To address this problem, a compression layer can be an efficient solution to reduce the communication overhead by compressing the tasks' data before offloading [30].

Let $x_{nm} \in \{0, 1\}$ denote the compression decision for the task m of IoT device n, where $x_{nm} = 0$ indicates that task m does not need to compress the data before offloading; and $x_{nm} = 1$ indicates that the data of task m is compressed before offloading. When the task is decided to be compressed before offloading, the extra overhead in terms of time and energy consumption for executing the task m on MEC servers or MCC servers can be calculated as:

$$\begin{cases} T_{nm}^{edge\_compr+decompr} = \frac{\sigma_{nm}}{f_{IoT}} + \frac{\eta_{nm}}{f_{edge}} \\ T_{nm}^{cloud\_compr+decompr} = \frac{\sigma_{nm}}{f_{IoT}} + \frac{\eta_{nm}}{f_{cloud}} \end{cases} \tag{9.13}$$

$$E_{nm}^{compr} = p_{ex}\frac{\sigma_{nm}}{f_{IoT}} \tag{9.14}$$

where σ_{nm} and η_{nm} denote the number of CPU cycles required to compress and decompress the data of computing task m on IoT device and MEC or MCC server, respectively.

After the data of the task is compressed, the transmission time and energy consumption will be

reduced, which are respectively calculated as:

$$\begin{cases} T_{nm}^{edge\_tr\_compr} = \frac{\beta_{nm}}{B_{lan}} \\ T_{nm}^{cloud\_tr\_compr} = \frac{\beta_{nm}}{B_{wan}} \end{cases} \tag{9.15}$$

$$E_{nm}^{tr\_compr} = p_{tr}\frac{\beta_{nm}}{B_{lan}} \tag{9.16}$$

where β_{nm} denotes data size of the computing task after compression, and $\beta_{nm} = \alpha_{nm} \times Compr\_ratio$. $Compr\_ratio$ is the compression ratio.

9.1.4 Security

In MEC and MCC framework, each IoT device can offload the tasks to the MEC and MCC servers through wireless links. However, this data includes private information such as photos, shopping or payment. Without a security mechanism, the private data are vulnerable to cyber-attack and damage while transmitting to edge or cloud servers. Hence, an efficient and secure layer is required to encrypt the data [29]. We select a cryptographic technique (e.g., AES) to encrypt and protect data.

Let $y_{nm} \in \{0, 1\}$ denote the security decision for the task m of IoT device n, where $y_{nm} = 0$ indicates that the task m of IoT device n will be offloaded without any encryption; $y_{nm} = 1$ indicates that the task m of IoT device n will be encrypted before offloading. MEC or MCC servers will decrypt the data after receiving the encrypted data of the task. When the IoT device considers the security layer, the extra overhead in terms of time and energy consumption for executing on MEC servers or MCC servers can be calculated as:

$$\begin{cases} T_{nm}^{edge\_enc+dec} = \frac{q_{nm}}{f_{IoT}} + \frac{d_{nm}}{f_{edge}} \\ T_{nm}^{cloud\_enc+dec} = \frac{q_{nm}}{f_{IoT}} + \frac{d_{nm}}{f_{cloud}} \end{cases} \tag{9.17}$$

$$E_{nm}^{enc} = p_{ex}\frac{q_{nm}}{f_{IoT}} \tag{9.18}$$

where q_{nm} and d_{nm} denote the number of CPU cycles required to encrypt and decrypt the data of computing task m on IoT device and MEC or MCC server, respectively. Accordingly, the security decision is determined by the IoT user based on private information. We set the security decision randomly in the simulation scenario.

9.1.5 Mobility

The mobility of IoT devices from one base station to another base station during the offloading period may influence the performance of the task offloading. We consider a dynamic offloading scenario that executes the mobility randomly of different IoT devices. Once the IoT device is moving from one base station to another new base station during the offloading period, it is necessary to consider the information hand-over mechanism between these different base stations. We design two basic information hand-over strategies for IoT devices and base stations. First, when the IoT device has finished transmitting all the offloaded data to the current base station, whereas it does not receive the result of the current MEC server and moves to another new base station, the current base station will transmit the result to the new base station and then send the result to the IoT device. Another situation is that the IoT device just transmits some part of data to the current base station and moves to the new base station, the IoT device will retransmit the remaining data to the new base station to guarantee data integrity and security.

Let $z_n \in \{0, 1\}$ denote the mobility decision for IoT device n, where $z_n = 0$ indicates that the IoT device n will transmit and receive the data from current base station; $z_n = 1$ indicates device n will move to another new base station during the offloading period. When device n considers mobility, the extra delay may be caused by moving between different base stations. We denote the mobility delay as τ_n, which is often determined by the moving distance of the IoT device.

$$T_n^{mob\_edge} = z_n \tau_n, \text{ if } \sum_{m=1}^{M} a_{nm}^1 \left(1 - a_{nm}^2\right) > 0, \forall n \in N \tag{9.19}$$

$$T_n^{mob\_\text{clond}} = z_n \tau_n, \text{ if } \sum_{m=1}^{M} a_{nm}^1 a_{nm}^2 > 0, \forall n \in N \tag{9.20}$$

9.1.6 Problem Formulation

With consideration of the collaborative MEC and MCC computation offloading, compression, security, and mobility, the total time and energy consumption of m-th task of IoT n in IoT-edge-cloud framework can be calculated as:

$$T_{nm}^{IoT} = \left(1 - a_{nm}^1\right) T_{nm}^{IoT\_exec} \tag{9.21}$$

$$\begin{aligned} T_{nm}^{edge} &= a_{nm}^1 \left(1 - a_{nm}^2\right) \left(1 - x_{nm}\right) T_{nm}^{edge\_tr+exec} \\ &+ a_{nm}^1 \left(1 - a_{nm}^2\right) x_{nm} T_{nm}^{edge\_compr+decompr} \\ &+ a_{nm}^1 \left(1 - a_{nm}^2\right) x_{nm} \left(T_{nm}^{edge\_tr\_compr} + T_{nm}^{edge\_exec}\right) \\ &+ a_{nm}^1 \left(1 - a_{nm}^2\right) y_{nm} T_{nm}^{edge\_enc+dec} \end{aligned} \tag{9.22}$$

$$\begin{aligned} T_{nm}^{cloud} &= a_{nm}^1 a_{nm}^2 \left(1 - x_{nm}\right) T_{nm}^{cloud\_tr+exec} \\ &+ a_{nm}^1 a_{nm}^2 x_{nm} T_{nm}^{cloud\_compr+decompr} \\ &+ a_{nm}^1 a_{nm}^2 x_{nm} \left(T_{nm}^{cloud\_tr\_compr} + T_{nm}^{cloud\_exec}\right) \\ &+ a_{nm}^1 a_{nm}^2 y_{nm} T_{nm}^{cloud\_enc+dec} \end{aligned} \tag{9.23}$$

$$\begin{aligned} E_{nm} &= \left(1 - a_{nm}^1\right) E_{nm}^{IoT\_exec} \\ &+ a_{nm}^1 \left(1 - a_{nm}^2\right) \left(1 - x_{nm}\right) E_{nm}^{edge\_tr} \\ &+ a_{nm}^1 a_{nm}^2 \left(1 - x_{nm}\right) E_{nm}^{cloud\_tr} \\ &+ a_{nm}^1 x_{nm} E_{nm}^{compr} + a_{nm}^1 x_{nm} E_{nm}^{tr\_compr} \\ &+ a_{nm}^1 y_{nm} E_{nm}^{enc} \end{aligned} \tag{9.24}$$

where T_{nm}^{IoT}, T_{nm}^{edge} and T_{nm}^{cloud} denote time consumption of task m executed on IoT devices, MEC servers and MCC servers, respectively. E_{nm} represents the energy consumption of the IoT devices for processing the task m.

The total execution time and energy consumption of all tasks of the n-th IoT can be expressed by:

$$T_n = \max\left\{T_n^{IoT}, T_n^{edge} + T_n^{mob\_edge}, T_n^{cloud} + T_n^{mob\_cloud}\right\} \tag{9.25}$$

$$E_n = \sum_{m=1}^{M} E_{nm} \tag{9.26}$$

where

$$\begin{cases} T_n^{IoT} = \sum\limits_{m=1}^{M} T_{nm}^{IoT} \\ T_n^{edge} = \sum\limits_{m=1}^{M} T_{nm}^{edge} \\ T_n^{cloud} = \sum\limits_{m=1}^{M} T_{nm}^{cloud} \end{cases} \tag{9.27}$$

The total time and energy consumption for executing all tasks can be given by:

$$T = \max \left\{ \sum_{n=1}^{N} T_n^{IoT}, \sum_{n=1}^{N} \left(T_n^{edge} + T_n^{mob\_edge} \right), \sum_{n=1}^{N} \left(T_n^{cloud} + T_n^{mob\_cloud} \right) \right\} \tag{9.28}$$

$$E = \sum_{n=1}^{N} E_n \tag{9.29}$$

Thus, we establish a dynamic and secure IoT-edge-cloud multi-objective computation offloading optimization model, with consideration of compression, security and mobility.

9.2 The Proposed MCOEA

This section presents the details of the proposed novel multi-objective computation offloading evolutionary algorithm (MCOEA) for solving dynamic and secure computation offloading problems in IoT-edge-cloud orchestrated computing networks.

9.2.1 General Framework

The general framework of the proposed algorithm is presented in Algorithm 11. In the beginning, the number of $\widetilde{N}$ solutions are initialized to form the population P. In each iteration, the offspring solution set O is generated by the crossover operator (Step 3). Then the offspring solution set O is mutated to retain the diversity by the mutation operator (Step 4). The original population P and new offspring O are combined to a new population. The non-dominated front number and crowding distance are calculated by the fast non-dominated sorting (Step 6) and crowding distance calculation (Step 7) methods in NSGA-II [24]. Finally, $\widetilde{N}$ solutions are selected from the combined population based on the non-dominated front number and crowding distance.

9.2.2 Crossover Operator

Considering that the multi-objective computation offloading decision model adopts the binary encoding method, the crossover operator applies a binary crossover idea to generate offspring solutions [103]. First, the mating pool P' randomly selects parents from the global population with a probability 0.8 and from the neighborhood at probability 0.2, respectively [58]. Through the neighborhood selection method, the closer solutions can exchange information to generate similar offspring solutions to improve the convergence. The selected parents from the global population can help the generated offspring solutions to keep the diversity. The neighborhood size T is set

Algorithm 11: Framework of MCOEA

Input: The population size $\widetilde{N}$
Output: The final population P

$P \leftarrow Initialization\left(\widetilde{N}\right)$;
while $termination\ criterion\ not\ fulfilled$ **do**
  $O = CrossoverOperator\left(P\right)$ in **Algorithm 12**;
  $O = MutationOperator\left(O\right)$ in **Algorithm 13**;
  $P \leftarrow P \cup O$;
  $[F_1, F_2, \cdots] \leftarrow NondominatedSorting\left(P\right)$;
  $CrowdDis \leftarrow CrowdingDistance\left(F_1, F_2, \cdots\right)$;
  $k \leftarrow \arg\min_i \left|F_1 \cup \cdots \cup F_i\right| \geq \widetilde{N}$;
  Delete $\left|F_1 \cup \cdots \cup F_k\right| - \widetilde{N}$ solutions from F_k with the smallest $CrowdDis$;
  $P \leftarrow F_1 \cup \cdots \cup F_k$;
end

to 20% of the population size, and the objective values of all solutions are normalized to the scale [0,1]. After that, a $\widetilde{N} * T$ neighborhood matrix Nic is generated, while $Nic\left(i\right)$ consists of T nearest solutions to $P\left(i\right)$ based on the Euclidean distances among normalized objective values.

We use the k-bit crossover method to exchange the decision variable values. First, the offspring set O is initialized to the same as P, we compare decision variable values of the parent $P'\left(i\right)$ with $P\left(i\right)$ and find the index j where the bit values of the two parents are different ($s.t.\ P\left(i,j\right) \neq P'\left(i,j\right)$). From the total $\left|j\right|$ indices, we select at least one but no more than $\left|j\right| - 1$ decision variables to cross between offspring solution $O\left(i\right)$ and the parent solution $P'\left(i\right)$. Compared with the traditional single-point crossover method [24], the k-bit crossover method can exchange information frequently to improve the convergence speed. The procedure of the crossover operator are shown in Algorithm 12.

9.2.3 Mutation Operator

The main idea of the mutation operator is to retain the diversity of the solutions. The mutation operator is detailed in Algorithm 13. Here we adopt the hybrid mutation operator to mutate the solutions [103]. The modified mutation method (Steps 2-8) and a traditional mutation method (Steps 9-15) are randomly applied with a probability of 0.2 and 0.8. In the modified mutation method, we find the indices t^1 and t^0 to satisfy $O\left(i,t^1\right) == 1$ and $O\left(i,t^0\right) == 0$, respectively. According to Steps 5-8, the specific actions are applied to turn the decision variable value 1 to 0 or 0 to 1.

Algorithm 12: $CrossoverOperator(P)$

Input: The population P
Output: The offspring set O

The population size $\widetilde{N} = |P|$;
Set the neighborhood size $T = \left\lceil \widetilde{N} * 0.2 \right\rceil$;
Normalize the objective values of all solutions in P to the scale [0,1];
Generate a neighborhood matrix Nic, while $Nic\,(i)$ consists of T nearest solutions to $P\,(i)$ based on the normalized Euclidean distances among solutions;
Generate an empty parent set P';
for $i = 1, \ldots, \widetilde{N}$ **do**
    if $rand < 0.8$ **then**
        Randomly select a parent from $Nic\,(i)$ into P';
    else
        Randomly select a parent from P into P';
    end
end
Initialize $O = P$;
for $i = 1, \ldots, \widetilde{N}$ **do**
    Find index j $s.t.$ $P\,(i,j) \neq P'\,(i,j)$;
    if $|j| > 1$ **then**
        $k = j(randperm\,(|j|, randi\,(\{1, \ldots, |j| - 1\}, 1)))$;
        $O\,(i,k) = P'\,(i,k)$;
    end
end

On the other hand, the traditional mutation method adopts the bitwise mutation [24]. The modified mutation method aims to balance the probabilities of exchanging the value of 1 and 0, respectively. We will give an example to show the advantage of the modified mutation method.

Assuming that the decision space dimension (denoted as D) is 10, the number of decision variables whose value equals 1 is 2 while the number of decision variables whose value equals 0 is 8. In the traditional bitwise mutation method, the mutation probability of each bit in the genetic algorithm is equal to the reciprocal of the encoding length (i.e., $1/D$). Therefore, in the traditional bitwise mutation method, the probability of transforming at least one value 1 to 0 is $1 - (1 - 1/10)^2 = 0.19$, while that of transforming at least one value 0 to 1 is $1 - (1 - 1/10)^8 \approx 0.57$. The probability of turning 0 to 1 is about three times higher than transforming 1 to 0, which is not fair to mutate 0 and 1 decision variables in the offspring solutions. By contrast, in the modified mutation method,

the probability of transforming at least one value 1 to 0 is $1-(1-1/(1+2))^2 \approx 0.56$, while that of transforming at least one value 0 to 1 is $1-(1-1/(1+8))^8 \approx 0.61$. It can be seen that the probability of turning binary variables to each other is similar, which can improve the diversity of the offspring solutions.

Algorithm 13: $MutationOperator(O)$

Input: The offspring set O
Output: The offspring set O

for $i = 1, \ldots, \widetilde{N}$ **do**
  if $rand < 0.2$ **then**
    Find index $t^1 \ \ s.t.\ O\left(i, t^1\right) == 1$;
    Find index $t^0 \ \ s.t.\ O\left(i, t^0\right) == 0$;
    $k1 = \left(rand\left(1, \left|t^1\right|\right) < 1/\left(1+\left|t^1\right|\right)\right)$;
    $k0 = \left(rand\left(1, \left|t^0\right|\right) < 1/\left(1+\left|t^0\right|\right)\right)$;
    $O\left(i, t^1\left(k1\right)\right) = \widetilde{O}\left(i, t^1\left(k1\right)\right)$;
    $O\left(i, t^0\left(k0\right)\right) = \widetilde{O}\left(i, t^0\left(k0\right)\right)$;
  else
    for $l = 1, \ldots, D$ **do**
      if $rand < 1/D$ **then**
        $O\left(i, l\right) = \widetilde{O}\left(i, l\right)$;
      end
    end
  end
end

9.2.4 Computational Complexity

In the proposed algorithm, the major costs are the iteration process in Algorithm 11. The crossover operator (Step 3) needs $O\left(m\widetilde{N}\right)$ operations to select parents from the mating pool and requires $O\left(\widetilde{N}D\right)$ operations to generate the offspring set, where m is the number of objectives, $\widetilde{N}$ is the population size, D is the number of decision variables. The mutation operator (Step 4) performs $O\left(\widetilde{N}D\right)$ operations to mutate the offspring solutions. The environment selection using fast non-dominated sorting and crowding distance methods (Steps 6-10) needs $O\left(m\widetilde{N}^2\right)$ to select the new solutions. To summarize, the overall computational complexity of MCOEA within one generation is $O\left(m\widetilde{N}^2\right)$.

9.3 Performance Evaluation

In this section, we evaluate the performance of the proposed MCOEA.

9.3.1 Experiment Profile

In the IoT-edge-cloud environment, we set up the multi-user multi-task computation offloading scenario with consideration of compression, security, and mobility. The number of IoT devices is selected between 20 and 200. The number of independent tasks of each IoT device is $M = 5$. The number of edge servers is $K = 3$. As an example, the CPU frequencies of IoT device, MEC server and MCC server are $f_{IoT} = 0.5$ GHz, $f_{edge} = 5$ GHz, and $f_{cloud} = 10$ GHz, respectively. The CPU computation power and data transmission power of IoT device are $p_{ex} = 0.7$ W, and $p_{tr} = 0.2$ W, respectively. The bandwidth of LAN and WAN are set to $B_{lan} \in (10, 15)$ MB/s and $B_{wan} \in (5, 10)$ MB/s. The data size of each task is uniformly distributed between 10 MB and 30 MB. Due to the security decision is based on the private information of task, while mobility decision is related to the mobility of IoT device, then we set the security decision and mobility decision randomly using uniform distribution in the following simulations.

Here we adopt the different types of applications in Table 7.2 to analyze the impact of application complexity. Furthermore, we add an augmented reality application as label F with $\rho = 12000$ cycles/byte. In addition, the label C represents the x264 CBR encode application and $\rho = 1900$ cycles/byte. The type C application is taken as an example of the offloading problems.

We compare MCOEA with five other algorithms to solve ten different offloading problems, which consider the number of IoT devices N = [20, 40, 60, 80, 100, 120, 140, 160, 180, 200], thus D = [300, 600, 900, 1200, 1500, 1800, 2100, 2400, 2700, 3000]. The comparison algorithms are NSGA-II [24], NSGA-III [22], NSGA-II/SDR [87], AREA [47], and PREA [108]. NSGA-II, NSGA-III, NSGA-II/SDR are representative Pareto dominance-based algorithms. AREA uses decomposition-based framework and PREA applies indicator-based method to solve MOPs, respectively.

In all algorithms, the population size and the number of function evaluations is set to 50 and $50 * D$, respectively. The solution encoding uses the binary-encoding method. The crossover and mutation operators in NSGA-II, NSGA-III, NSGA-II/SDR, AREA, and PREA adopt the single-point and bitwise mutation operators. We utilize the HV [96] as the performance metric. Each algorithm is executed 30 times independently and the average and standard deviation of the metric values are recorded. The Wilcoxon rank sum test at a 5% significance level is used to analyze the experimental results.

9.3.2 Convergence Analysis

Table 9.2 lists the HV metric values obtained by NSGA-II, NSGA-III, NSGA-II/SDR, AREA, PREA and MCOEA for solving ten computation offloading problems. The proposed MCOEA has achieved the best performance on 9 of 10 test instances, while PREA only gets 1 of 10 best results for the rest of compared algorithms. Compared with the other five comparison algorithms NSGA-II, NSGA-III, NSGA-II/SDR, AREA, and PREA, MCOEA can get better results in terms of HV metric. With the increment of the dimension, MCOEA and PREA have an obvious advantage over other compared algorithms.

Fig. 9.2 presents the non-dominated solution set with the medium HV value obtained by NSGA-II, NSGA-III, NSGA-II/SDR, AREA, PREA, and MCOEA on N = 20, 100, and 200 computation offloading problems. We can observe that MCOEA can always obtain a set of well-distributed and well-converged solutions for different dimensional computation offloading problems. With the growth of the number of the IoT devices, the performance of convergence or diversity of NSGA-II, NSGA-III, NSGA-II/SDR, AREA, and PREA deteriorates, while PREA can obtain a better solution set than NSGA-II, NSGA-III, NSGA-II/SDR and AREA.

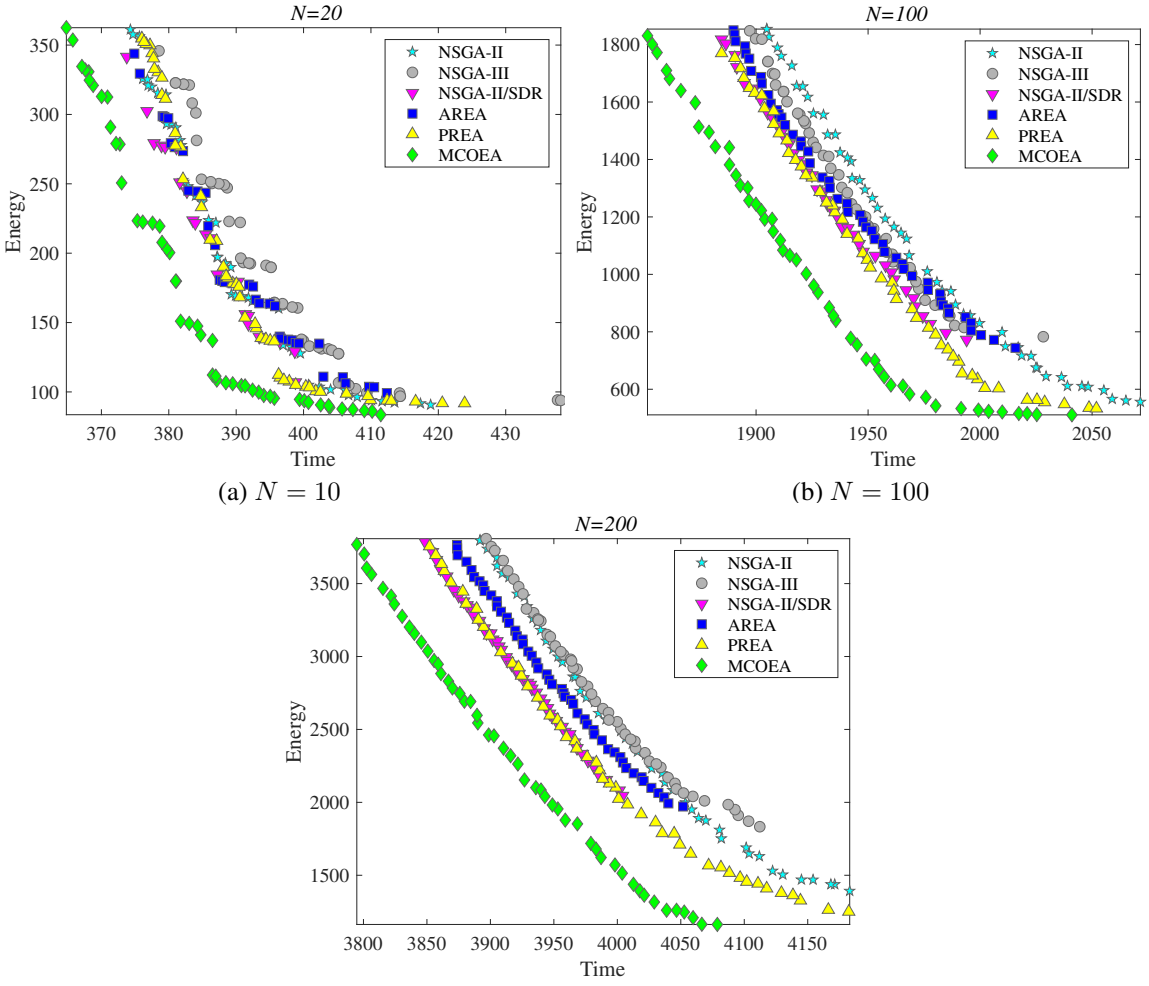
9.3.3 Compression Security Mobility Analysis

Compression, security and mobility models play a significant role in the computation offloading problems. We take $N = 100$ computation offloading instance as an example to analyze the performance. Fig. 9.3a presents the Pareto fronts of the $N = 100$ computation offloading model with and without compression. As shown in Fig. 9.3a, the offloading model with compression can save time and energy compared with the model without compression to some extent. Since the compression strategy makes the data size of tasks become small, the offloading decision model can reduce transmission time and transmission energy consumption. However, the compression process can cause extra energy consumption in the IoT device. Hence, most of the non-dominated solutions in the offloading model with compression have better convergence than that in the model without compression, whereas the model without compression can still get the least energy consumption.

Fig. 9.3b shows the Pareto fronts of the $N = 100$ computation offloading model with and without security. Considering the security layer, the encryption and decryption of data size of tasks will increase the time and energy consumption of IoT devices and edge/cloud servers, respectively. The non-dominated solution set in the model without security has better convergence than that in the model with security. Fig. 9.3c illustrates the Pareto fronts of the $N = 100$ computation offloading model with and without mobility. It can be seen that when the model considers the mobility of

Table 9.2: The HV values obtained by NSGA-II, NSGA-III, NSGA-II/SDR, AREA, PREA and MCOEA on ten computation offloading problems

Problem	N	NSGA-II	NSGA-III	NSGA-II/SDR
Offloading1	20	4.2722e-1 (7.03e-3) $-$	4.1602e-1 (8.32e-3) $-$	4.1653e-1 (4.54e-3) $-$
Offloading2	40	4.2494e-1 (1.07e-2) $-$	4.0896e-1 (1.10e-2) $-$	4.0367e-1 (9.64e-3) $-$
Offloading3	60	4.2757e-1 (6.63e-3) $-$	4.0125e-1 (1.30e-2) $-$	3.9600e-1 (1.18e-2) $-$
Offloading4	80	4.3037e-1 (6.30e-3) $-$	3.9308e-1 (1.80e-2) $-$	3.9747e-1 (7.23e-3) $-$
Offloading5	100	4.1932e-1 (7.65e-3) $-$	3.7703e-1 (1.40e-2) $-$	3.9042e-1 (1.46e-2) $-$
Offloading6	120	2.4780e-1 (6.85e-4) $-$	2.4587e-1 (6.99e-4) $-$	2.4483e-1 (4.67e-4) $-$
Offloading7	140	4.2395e-1 (8.49e-3) $-$	3.8467e-1 (1.68e-2) $-$	3.8032e-1 (1.11e-2) $-$
Offloading8	160	4.0963e-1 (9.93e-3) $-$	3.6636e-1 (1.04e-2) $-$	3.6394e-1 (9.95e-3) $-$
Offloading9	180	4.0951e-1 (6.38e-3) $-$	3.7475e-1 (1.19e-2) $-$	3.5894e-1 (9.35e-3) $-$
Offloading10	200	4.0680e-1 (7.98e-3) $-$	3.6521e-1 (8.73e-3) $-$	3.5369e-1 (1.09e-2) $-$
$+/-/\approx$		0/10/0	0/10/0	0/10/0
Problem	N	AREA	PREA	MCOEA
Offloading1	20	4.1892e-1 (4.69e-3) $-$	4.3319e-1 (2.91e-3) $-$	4.3859e-1 (3.90e-3)
Offloading2	40	4.0481e-1 (1.36e-2) $-$	4.3293e-1 (4.44e-3) $\approx$	4.3598e-1 (2.55e-3)
Offloading3	60	4.0034e-1 (1.76e-2) $-$	4.3522e-1 (3.59e-3) $\approx$	4.3591e-1 (2.60e-3)
Offloading4	80	4.0213e-1 (1.10e-2) $-$	4.3895e-1 (5.52e-3) $\approx$	4.4069e-1 (4.50e-3)
Offloading5	100	3.9075e-1 (1.13e-2) $-$	4.3270e-1 (7.33e-3) $\approx$	4.3652e-1 (4.20e-3)
Offloading6	120	2.8219e-1 (2.73e-3) $-$	2.7564e-1 (2.51e-3) $-$	2.9018e-1 (2.00e-4)
Offloading7	140	3.7507e-1 (1.52e-2) $-$	4.3404e-1 (6.98e-3) $\approx$	4.3936e-1 (3.27e-3)
Offloading8	160	3.6943e-1 (1.40e-2) $-$	4.2445e-1 (6.01e-3) $\approx$	4.2853e-1 (5.05e-3)
Offloading9	180	3.6307e-1 (9.46e-3) $-$	4.3081e-1 (6.23e-3) $\approx$	4.2785e-1 (4.79e-3)
Offloading10	200	3.6055e-1 (1.09e-2) $-$	4.2403e-1 (3.77e-3) $\approx$	4.2723e-1 (4.69e-3)
$+/-/\approx$		0/10/0	0/2/8	



(a) $N = 10$ (b) $N = 100$

(c) $N = 200$

Figure 9.2: The non-dominated solution set with the medium HV value obtained by NSGA-II, NSGA-III, NSGA-II/SDR, AREA, PREA, and MCOEA on $N = 10, 100, 200$ offloading problems

the IoT devices, the extra delay will be compared with the model without mobility, while energy consumption will not change so much.

9.3.4 Comparison with Different Offloading Schemes

We use the other four offloading schemes, i.e., LOS, EOS, COS and ROS of Section 7.2.3 to further compare the performance with the proposed MCOEA. The system cost and offloading gain in Equations 7.22 and 7.23 are adopted as performance metrics. It is noted that the proposed MCOEA

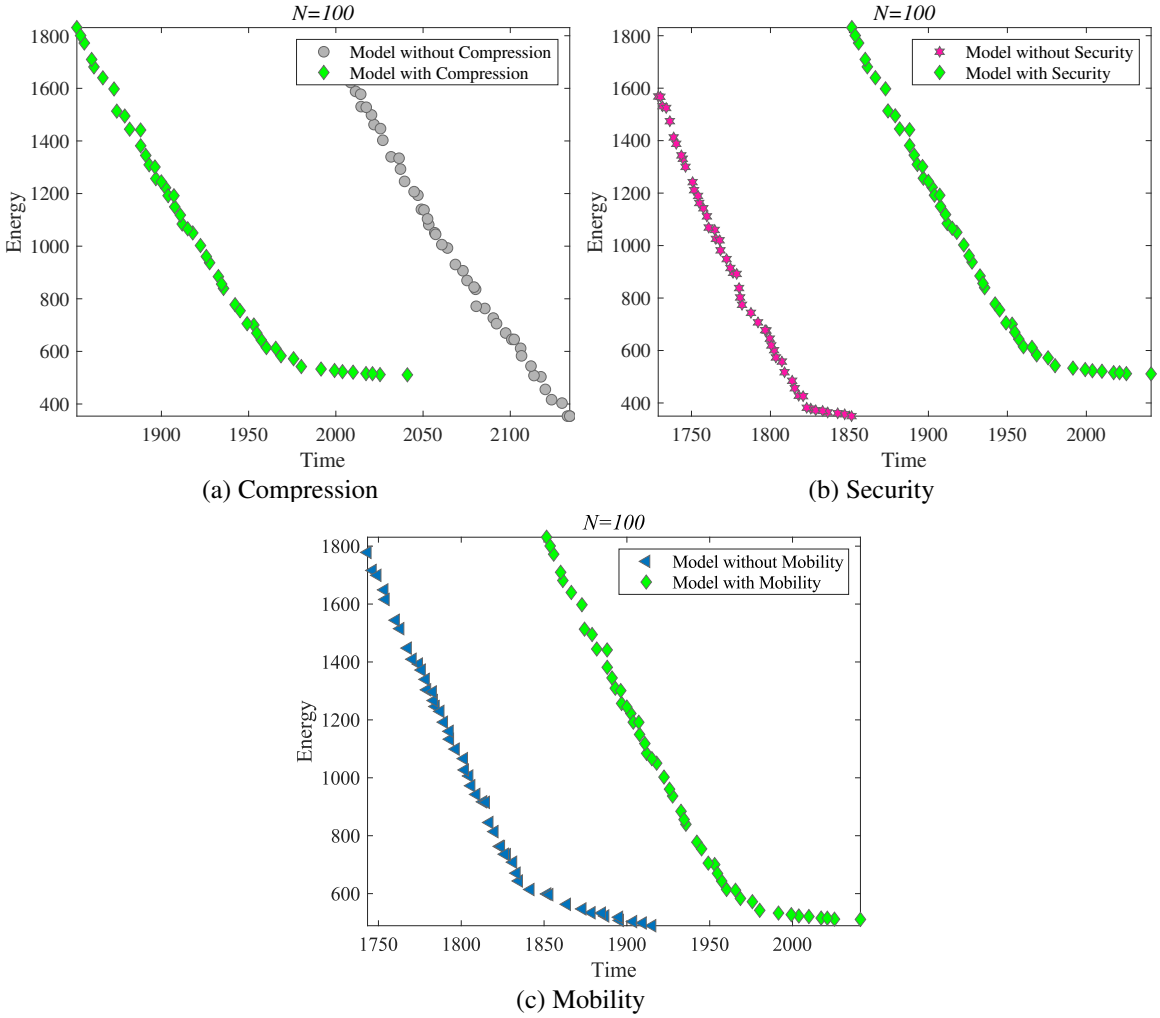
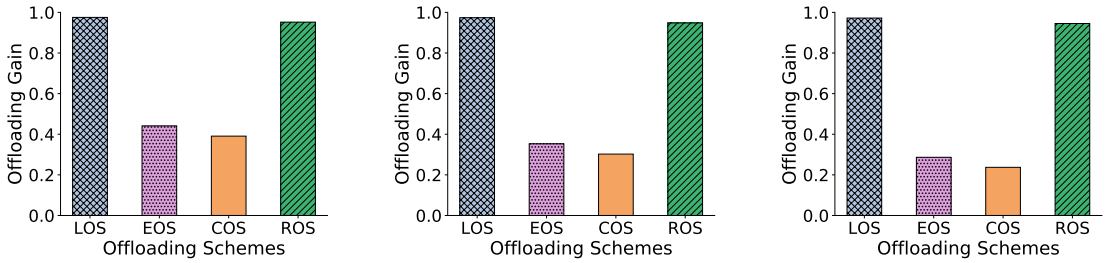


Figure 9.3: The Pareto fronts of the $N = 100$ computation offloading model with regard to compression, security and mobility

is adopted as the baseline in the offloading gain metric, which measures the benefits of the MCOEA compared with LOS, EOS, COS and ROS, respectively.

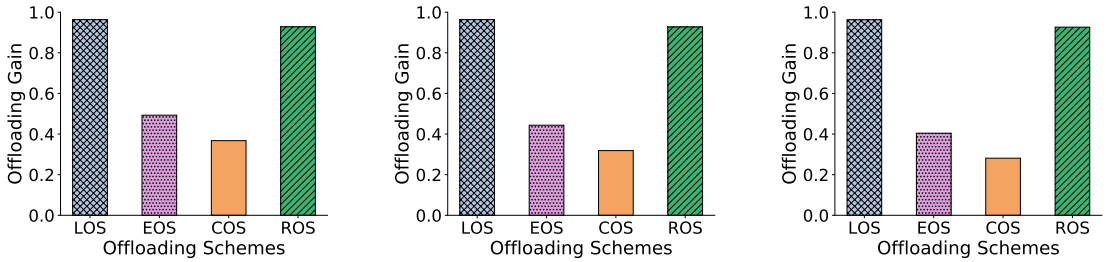
Figs. 9.4, 9.5, and 9.6 present the offloading gain of different offloading schemes under different weights. Compared with LOS, EOS, COS and ROS, MCOEA can always benefit a lot with regard to time and energy consumption, especially for LOS and ROS. COS can get better performance than EOS. With the growing number of IoT devices, more and more tasks will be chosen to be offloaded to MCC servers since cloud servers have more powerful computation capability. Hence,

the offloading gain between MCOEA and COS will decrease due to the growing number of IoT devices. In addition, with the increment of the tradeoff parameter w, the offloading gain of EOS and COS will be improved, which means the MCOEA is more sensitive to time latency than energy consumption.



(a) $N = 20$ (b) $N = 100$ (c) $N = 200$

Figure 9.4: Offloading gain of different offloading schemes for $w = 0.2$

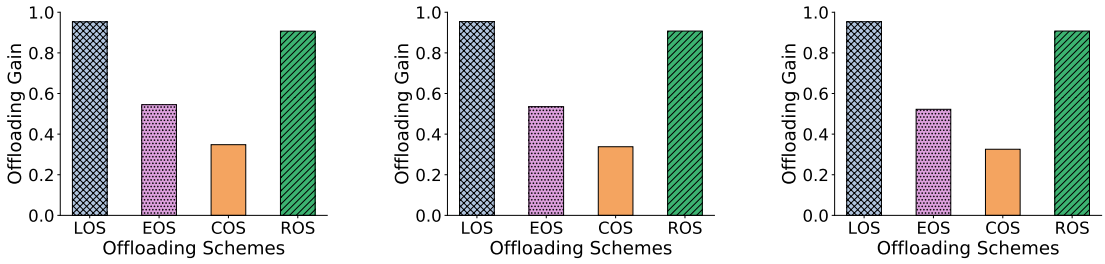


(a) $N = 20$ (b) $N = 100$ (c) $N = 200$

Figure 9.5: Offloading gain of different offloading schemes for $w = 0.5$

9.3.5 Impact of System Parameters

The impact of different system parameters are analyzed in the IoT-edge-cloud computation offloading model, where $w = 0.5$ and $N = 100$. Fig. 9.7 presents the performance of system cost and offloading gain on different offloading schemes under the different average data size. MCOEA can achieve the best results with respect to system cost and offloading gain. With the increment of average data size of tasks, the system cost of LOS and ROS increase much faster than EOS, COS and MCOEA, while the offloading gain of EOS and COS will decrease since that the complicated tasks will be more likely to be processed on MEC or MCC servers.



(a) $N = 20$ (b) $N = 100$ (c) $N = 200$

Figure 9.6: Offloading gain of different offloading schemes for $w = 0.8$

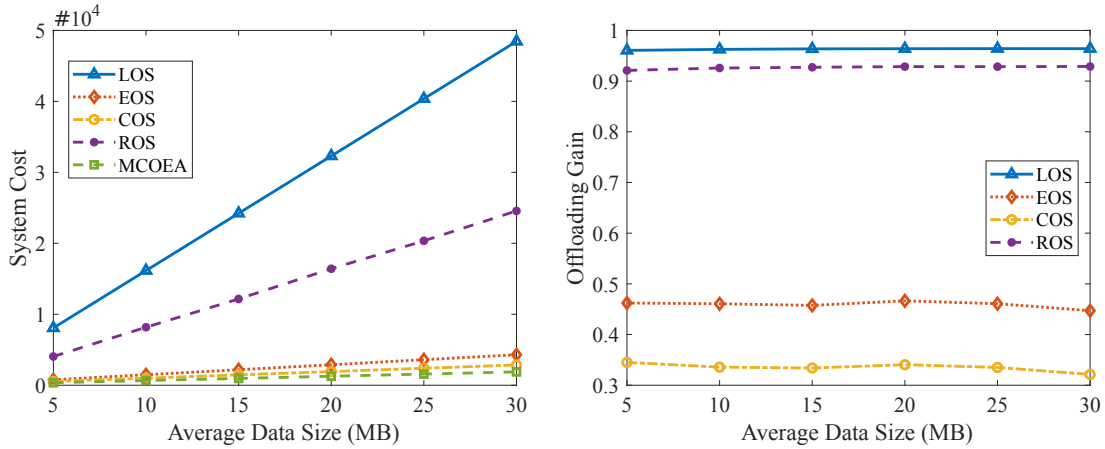


Figure 9.7: System cost and offloading gain on different offloading schemes under different average data size

Fig. 9.8 illustrates the performance of system cost and offloading gain on different offloading schemes under different edge server CPU frequency. The system cost of LOS and COS stay the same. With the increment of edge server CPU frequency, the system cost of EOS will decrease obviously in the beginning and then decrease slowly. Since the edge server CPU frequency is increasing, more and more tasks can be offloaded to MEC servers, and the offloading gain of EOS will decrease while COS is on the contrary.

Fig. 9.9 shows the performance of system cost and offloading gain on different offloading schemes under different types of applications. With the increment of parameter ρ, the total CPU cycles for processing tasks are increasing, causing more time and energy consumption for finishing tasks. The increasing speed of system cost of LOS is much faster than EOS and COS due to the fact that the computing capability of MEC and MCC servers are more powerful than IoT devices. The offloading

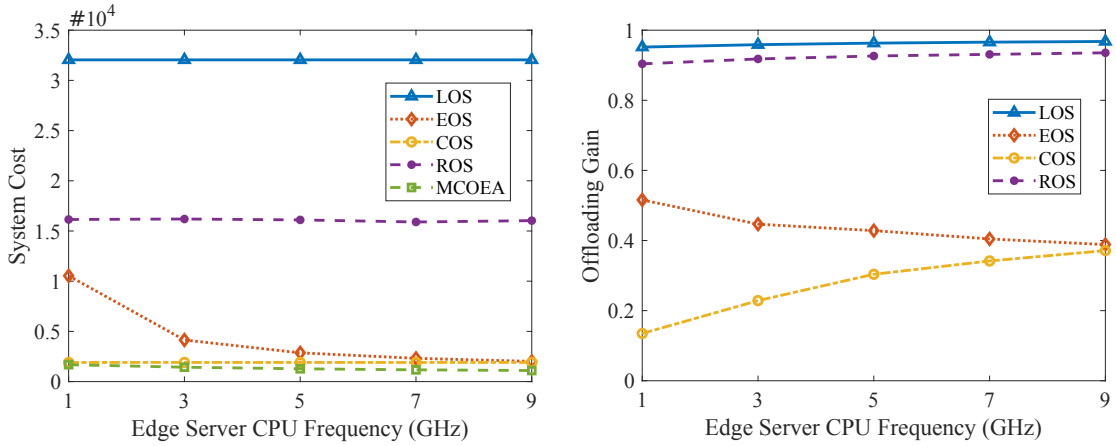


Figure 9.8: System cost and offloading gain on different offloading schemes under different edge server CPU frequency

gain of EOS and COS is increasing with the increment of ρ, while they begin to decrease after ρ is increasing to a certain extent, that is because the system cost of EOS and COS belongs to an increasingly relevant proportion of MCOEA.

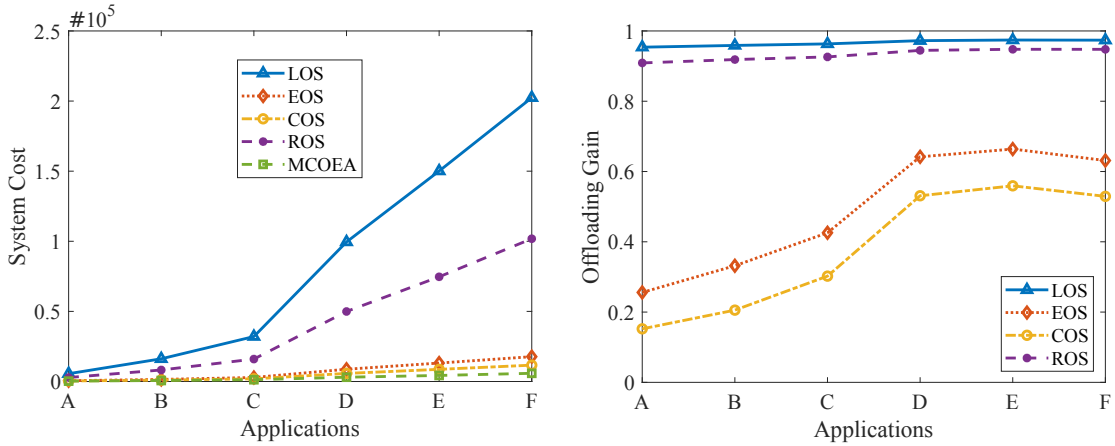


Figure 9.9: System cost and offloading gain on different offloading schemes under different types of applications

9.4 Summary

In this chapter, we proposed a novel multi-objective computation offloading evolutionary algorithm (MCOEA) for solving dynamic and secure computation offloading problems in collaborative IoT-edge-cloud computing networks. We establish the multi-objective computation offloading model with the consideration of compression, security and mobility. The compression model can compress the data size of tasks to reduce the transmission overhead. The security layer is used to encrypt private data for protecting personal information. The mobility model reflects the dynamic positions of IoT devices in the offloading period and will cause extra delay. Furthermore, the binary k-bit crossover and hybrid mutation can be used to improve the convergence and diversity of the non-dominated solutions. The proposed MCOEA is compared with the other five state-of-the-art multi-objective evolutionary algorithms and four offloading strategies on ten different offloading problems. The experiment results demonstrate the efficiency and superiority of the proposed algorithm.

Part IV

Concluding Remarks

Chapter 10

Conclusions and Outlook

10.1 Conclusions

In this thesis, the main goal has been to design effective and efficient multi-objective evolutionary algorithms (MOEAs) to solve multi- and many-objective optimization problems (MOPs and MaOPs), constrained and large-scale optimization problems. Then we apply and develop multi-objective optimization methods to deal with different computation offloading problems in collaborative edge-cloud computing. We discuss the contributions as follows:

- *Multi-objective Optimization Algorithm*: A multi-objective artificial bee colony algorithm using the decomposition approach is proposed to solve normalized and scaled MOPs (**Chapter 3**). We substitute the artificial bee colony operator for the original genetic operator to improve the convergence. The modified Tchebycheff approach is used to retain diversity. An adaptive normalization method is adopted to solve scaled problems. The proposed algorithm exhibits better convergence and diversity than other compared algorithms on most instances.
- *Many-objective Optimization Algorithm*: A decomposition-based evolutionary algorithm with adaptive weight vectors is presented for solving the normalized and scaled MaOPs (**Chapter 4**). We use an adaptive weight vector method to tune weight vectors for solving scaled problems instead of using normalization approaches. We compare and analyze the performance of the existing six decomposition approaches and select the best one. Further, one novel replacement strategy is adopted to keep the balance between convergence and diversity for MaOPs. The algorithm is reliable for dealing with different normalized and scaled MaOPs.
- *Adaptive Algorithm for Irregular Pareto fronts*: An adaptive decomposition-based evolutionary algorithm is developed to solve different MOPs and MaOPs with irregular Pareto fronts

(**Chapter 5**). The archive in the algorithm is used to store non-dominated solutions. We design a novel archive maintenance strategy to avoid the dominance resistant solutions as well as retain the good diversity. An adaptive adding and deleting weight vector method is adopted to solve MOPs and MaOPs with regular and irregular Pareto fronts. The proposed algorithm can achieve good performance of both convergence and diversity on different MOPs and MaOPs with various Pareto front shapes (the simplex-like, the inverted, the disconnected, the degenerated, the scaled, the mixed, the high dimensional).

- *Constrained Optimization in Optimization*: Three tailored constrained multi-objective optimization algorithms are designed to solve different constrained MOPs (CMOPs) selected from widely used benchmark suites (**Chapter 6**). Then we apply these three constrained algorithms to deal with constrained computation offloading optimization problems in mobile edge-cloud computing (**Chapter 7**). The proposed algorithms use two search stages (without considering constraint handling first and then with constraint handling). The two search stages can help the algorithms to jump over the infeasible regions. The proposed constrained optimization algorithms can work well for the complicated benchmark test problems as well as constrained applications offloading optimization problems.
- *Large-scale Offloading Optimization*: Two evolutionary large-scale sparse multi-objective optimization algorithms are developed to solve large-scale computation offloading problems in collaborative edge-cloud computing (**Chapter 8**). First we establish a large-scale multi-objective computation offloading optimization model. The restricted Boltzmann machine (RBM) is used to reduce the dimensionality of the problem. The contribution scores of decision variables are applied to select better decision variables for generating offspring solutions. The proposed algorithms can achieve good performance for large-scale offloading optimization problems.
- *Dynamic and Secure Offloading Optimization*: A novel multi-objective computation offloading evolutionary algorithm (MCOEA) is proposed to tackle dynamic and secure computation offloading problems (**Chapter 9**). We set up a dynamic multi-objective computation offloading model considering compression, security and mobility. The new binary crossover and mutation operators are designed in the proposed algorithm to improve convergence and diversity for solving the model. Furthermore, we analyze the impact of compression, security and mobility in the offloading optimization.

The source codes of this work are available on GitHub: https://github.com/Guangfu17. The proposed multi-objective optimization algorithms in the thesis can be also used to solve two and three-objective optimization problems in other research areas. The designed many-objective

optimization algorithm can be applied to deal with optimization problems with more than three objectives. In addition, the tailored constrained optimization algorithms are able to tackle constrained multi-objective optimization problems having a set of constraints. The large-scale optimization algorithms can be applied to solve large-scale multi-objective optimization problems with large-scale binary decision variables.

10.2 Outlook

Several directions for future work are worth investigated. The main ideas of them are summarized as follows:

- In order to improve the performance of decomposition-based multi-objective evolutionary algorithms for solving MOPs and MaOPs with irregular Pareto fronts, adjusting weight vectors to be suitable for various Pareto front shapes is one of the key factors in the designed algorithm. However, the weight vector adaptation methods may bring two shortcomings. The first one is that it may deteriorate the performance of dealing with problems with regular Pareto fronts. The second one is that it can increase the computational complexity of the algorithm. Some other more efficient and effective adaptation methods of decomposition-based algorithms will be studied.
- For large-scale multi-objective optimization, some other more efficient methods that can reduce the dimensionality will be considered. The relationship between different decision variables as well as relevant groups for the decision variables will be investigated. Furthermore, the efficient large-scale offloading decision algorithms will be applied to a real software deployment framework in mobile edge/cloud computing.
- The different computing tasks in IoT devices are assumed to be independent in this work. However, a user often needs to execute multiple related tasks, the input of one task may require the output of another. In the future, the dependencies between the tasks in various applications will be considered.
- During the process of task offloading, the transmitted data is vulnerable. To remedy this issue, the blockchain technology can be employed in mobile edge computing to ensure the data integrity. In addition, a decentralized, transparent and trustworthy toll collection based on blockchain technology is worth investigating to motivate the heterogeneous edge platforms to share their vacant resource.
- Adaptive deep learning methods for tackling task offloading problems is another interest-

ing research topic. On one hand, by aggregating the decision-making ability of reinforcement learning, and the rapid environment learning ability of meta-learning, it is possible to quickly and flexibly obtain the near-optimal offloading strategy from the dynamic environment. What's more, federated learning can be applied to solve computation offloading problems for guaranteeing data privacy and security.

Bibliography

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018.

[2] M. Asafuddoula, T. Ray, and R. Sarker. A decomposition-based evolutionary algorithm for many objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(3):445–460, 2014.

[3] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1):45–76, 2011.

[4] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[5] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. An empirical assessment of the properties of inverted generational distance on multi- and many-objective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 31–45, Cham, 2017. Springer International Publishing.

[6] S. Bi, L. Huang, and Y. A. Zhang. Joint optimization of service caching placement and computation offloading in mobile edge computing systems. *IEEE Transactions on Wireless Communications*, 19(7):4947–4963, 2020.

[7] S. Bi and Y. J. Zhang. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Transactions on Wireless Communications*, 17(6):4177–4190, 2018.

[8] C. Blum. m. dorigo, t. stützle, ant colony optimization (2004) mit press, cambridge, ma 300 pp. *Artificial Intelligence*, 165(2):261–264, 2005.

[9] P. A. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):174–188, 2003.

[10] X. Cai, Y. Li, Z. Fan, and Q. Zhang. An external archive guided multiobjective evolutionary

algorithm based on decomposition for combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 19(4):508–523, 2015.

[11] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

[12] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. Test problems for large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*, 47(12):4108–4121, 2017.

[13] R. Cheng, M. Li, Y. Tian, X. Zhang, and S. Yang. A benchmark test suite for evolutionary many-objective optimization. *Complex & Intelligent Systems*, 3(1):67–81, 2017.

[14] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[15] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 283–290. Morgan Kaufmann Publishers Inc., 2001.

[16] C. Darwin. *On the Origin of Species by Means of Natural Selection: Or the Preservation of Favoured Races in the Struggle for Life*. D. Appleton, 1869.

[17] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, 8(3):631–657, 1998.

[18] L. R. de Farias, P. H. Braga, H. F. Bassani, and A. F. Araújo. Moea/d with uniformly randomly adaptive weights. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 641–648. ACM, 2018.

[19] K. Deb. An introduction to genetic algorithms. *Sadhana*, 24(4):293–315, 1999.

[20] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[21] K. Deb and M. Goyal. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and informatics*, 26:30–45, 1996.

[22] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[23] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[25] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001.

[26] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin. Learning for computation offloading in mobile edge computing. *IEEE Transactions on Communications*, 66(12):6353–6367, 2018.

[27] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, 65(8):3571–3584, 2017.

[28] J. Du, L. Zhao, J. Feng, and X. Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2018.

[29] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li. Resource allocation and computation offloading with data security for mobile edge computing. *Future Generation Computer Systems*, 100:531–541, 2019.

[30] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang. Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile iot networks. *IEEE Transactions on Network and Service Management*, 17(4):2410–2422, 2020.

[31] Z. Fan, Y. Fang, W. Li, J. Lu, X. Cai, and C. Wei. A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 209–216. IEEE, 2017.

[32] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, and E. Goodman. An improved epsilon constraint-handling method in moea/d for cmops with large infeasible regions. *Soft Computing*, 23(23):12491–12510, 2019.

[33] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman. Push and pull search for solving constrained multi-objective optimization problems. *Swarm and Evolutionary Computation*, 44:665–679, 2019.

[34] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman. Difficulty adjustable and scalable constrained multiobjective test problem toolkit. *Evolutionary computation*, 28(3):339–378, 2020.

[35] Z. Fan, Z. Wang, W. Li, Y. Yuan, Y. You, Z. Yang, F. Sun, and J. Ruan. Push and pull search embedded in an m2m framework for solving constrained multi-objective optimization problems. *Swarm and Evolutionary Computation*, 54:100651, 2020.

[36] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming. Towards understanding the cost of adaptation in decomposition-based optimization algorithms. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 615–620. IEEE, 2013.

[37] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya. An application placement technique for concurrent iot applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, 2020.

[38] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung. An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. *IEEE/ACM Transactions on Networking*, 26(6):2651–2664, 2018.

[39] D. Han, W. Du, W. Du, Y. Jin, and C. Wu. An adaptive decomposition-based evolutionary algorithm for many-objective optimization. *Information Sciences*, 491:204–222, 2019.

[40] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, and X. Yao. Accelerating large-scale multiobjective optimization via problem reformulation. *IEEE Transactions on Evolutionary Computation*, 23(6):949–961, 2019.

[41] Z. He, G. G. Yen, and J. Zhang. Fuzzy-based pareto optimality for many-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 18(2):269–285, 2013.

[42] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[43] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu. Multi-server multi-user multi-task computation offloading for mobile edge computing networks. *Sensors*, 19(6):1446, 2019.

[44] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[45] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima. Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190, 2017.

[46] H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2014.

[47] S. Jiang, H. Li, J. Guo, M. Zhong, S. Yang, M. Kaiser, and N. Krasnogor. Area: An adaptive reference-set based evolutionary algorithm for multiobjective optimisation. *Information Sciences*, 515:365–387, 2020.

[48] S. Jiang and S. Yang. A strength pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization. *IEEE Transactions on Evolutionary*

Computation, 21(3):329–346, 2017.

[49] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, Engineering Faculty Computer Engineering Department, 2005.

[50] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, 1995.

[51] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219 – 235, 2019.

[52] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[53] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher. *Computational Intelligence - A Methodological Introduction Second Edition*. Springer, 2016.

[54] L. Kuang, T. Gong, S. OuYang, H. Gao, and S. Deng. Offloading decision methods for multiple users with structured tasks in edge computing for smart cities. *Future Generation Computer Systems*, 105:717–729, 2020.

[55] B. Li, J. Li, K. Tang, and X. Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 48(1):13, 2015.

[56] B. Li, K. Tang, J. Li, and X. Yao. Stochastic ranking algorithm for many-objective optimization based on multiple indicators. *IEEE Transactions on Evolutionary Computation*, 20(6):924–938, 2016.

[57] H. Li, J. Sun, Q. Zhang, and Y. Shui. Adjustment of weight vectors of penalty-based boundary intersection method in moea/d. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 91–100. Springer, 2019.

[58] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

[59] K. Li, R. Chen, G. Fu, and X. Yao. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315, 2018.

[60] K. Li, K. Deb, Q. Zhang, and S. Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716, 2015.

[61] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 18(6):909–923, 2013.

[62] M. Li, S. Yang, and X. Liu. Shift-based density estimation for pareto-based algorithms in

many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):348–365, 2013.

[63] M. Li and X. Yao. What weights work for you? adapting weights for any pareto front shape in decomposition-based evolutionary multiobjective optimisation. *Evolutionary Computation*, 28(2):227–253, 2020.

[64] H.-L. Liu, F. Gu, and Q. Zhang. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation*, 18(3):450–455, 2013.

[65] Y. Liu, Y. Hu, N. Zhu, K. Li, J. Zou, and M. Li. A decomposition-based multiobjective evolutionary algorithm with weights updated adaptively. *Information Sciences*, 2021.

[66] Z.-Z. Liu and Y. Wang. Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces. *IEEE Transactions on Evolutionary Computation*, 23(5):870–884, 2019.

[67] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Transactions on Evolutionary Computation*, 20(2):275–298, 2016.

[68] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi. Optimal joint scheduling and cloud offloading for mobile applications. *IEEE Transactions on Cloud Computing*, 7(2):301–313, 2019.

[69] S. Mane and M. R. N. Rao. Many-objective optimization: Problems and evolutionary algorithms–a short review. *International Journal of Applied Engineering Research*, 12(20), 2017.

[70] P. M. Mell and T. Grance. *SP 800-145. The NIST Definition of Cloud Computing*. National Institute of Standards & Technology, 2011.

[71] A. P. Miettinen and J. K. Nurminen. Energy efficiency of mobile clients in cloud computing. *HotCloud*, 10(4-4):19, 2010.

[72] M. Patel, B. Naughton, C. Chan, et al. Mobile-edge computing—introductory technical white paper. *White Paper, Mobile-edge Computing (MEC) Industry Initiative*, 2014.

[73] K. Peng, H. Huang, S. Wan, and V. C. Leung. End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment. *Wireless Networks*, pages 1–12, 2020.

[74] R. C. Purshouse and P. J. Fleming. On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, 11(6):770–784, 2007.

[75] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu. Moea/d with adaptive weight adjustment. *Evolutionary computation*, 22(2):231–264, 2014.

[76] M. Reyes-Sierra, C. C. Coello, et al. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[77] G. Rozenberg, T. H. W. Back, and J. N. Kok. Handbook of natural computing. *Kybernetes*, 40(3/4):20–69, 2012.

[78] I. Sahu and U. S. Pandey. Mobile cloud computing: Issues and challenges. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 247–250, 2018.

[79] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello. Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504–522, 2012.

[80] I. Sheikh and O. Das. Modeling the effect of parallel execution on multi-site computation offloading in mobile cloud computing. In *Computer Performance Engineering*, pages 219–234, 2018.

[81] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[82] D. Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[83] Y. Sun, G. G. Yen, and Z. Yi. Igd indicator-based evolutionary algorithm for many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, 23(2):173–187, 2018.

[84] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin. An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Transactions on Evolutionary Computation*, 22(4):609–622, 2017.

[85] Y. Tian, R. Cheng, X. Zhang, and Y. Jin. Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017.

[86] Y. Tian, R. Cheng, X. Zhang, M. Li, and Y. Jin. Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [research frontier]. *IEEE Computational Intelligence Magazine*, 14(3):61–74, 2019.

[87] Y. Tian, R. Cheng, X. Zhang, Y. Su, and Y. Jin. A strengthened dominance relation considering convergence and diversity for evolutionary many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):331–345, 2018.

[88] Y. Tian, C. Lu, X. Zhang, K. C. Tan, and Y. Jin. Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks. *IEEE Transactions on Cybernetics*, pages 1–14, 2020.

[89] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin. A coevolutionary framework for constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 25(1):102–116, 2021.

[90] Y. Tian, X. Zhang, C. Wang, and Y. Jin. An evolutionary algorithm for large-scale sparse multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 24(2):380–393, 2020.

[91] Y. Tian, Y. Zhang, Y. Su, X. Zhang, K. C. Tan, and Y. Jin. Balancing objective optimization and constraint satisfaction in constrained evolutionary multi-objective optimization. *IEEE Transactions on Cybernetics*, 2020.

[92] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2017.

[93] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis, 1998.

[94] P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016.

[95] R. Wang, Q. Zhang, and T. Zhang. Decomposition-based algorithms using pareto adaptive scalarizing methods. *IEEE Transactions on Evolutionary Computation*, 20(6):821–837, 2016.

[96] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, 2006.

[97] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, 2009.

[98] H. Wu. *Analysis of offloading decision making in mobile cloud computing*. PhD thesis, Freien Universität Berlin, 2015.

[99] H. Wu. Multi-objective decision-making for mobile cloud offloading: A survey. *IEEE Access*, 6:3962–3976, 2018.

[100] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu. Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. *IEEE Internet of Things Journal*, 7(9):8099–8110, 2020.

[101] Y. Xiang, Y. Zhou, M. Li, and Z. Chen. A vector angle-based evolutionary algorithm for unconstrained many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 21(1):131–152, 2016.

[102] Y. Xiang, Y. Zhou, X. Yang, and H. Huang. A many-objective evolutionary algorithm with pareto-adaptive reference points. *IEEE Transactions on Evolutionary Computation*, 24(1):99–113, 2020.

[103] H. Xu, B. Xue, and M. Zhang. A duplication analysis based evolutionary algorithm for biobjective feature selection. *IEEE Transactions on Evolutionary Computation*, 2020.

[104] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi. A computation offloading method over big data for iot-enabled cloud-edge computing. *Future Generation Computer Systems*, 95:522–533, 2019.

[105] S. Yang, S. Jiang, and Y. Jiang. Improving the multiobjective evolutionary algorithm based on decomposition with new penalty schemes. *Soft Computing*, 21(16):4677–4691, 2017.

[106] S. Yang, M. Li, X. Liu, and J. Zheng. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(5):721–736, 2013.

[107] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, and J. Zhang. Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading. *IEEE Wireless Communications*, 27(1):92–99, 2020.

[108] J. Yuan, H.-L. Liu, F. Gu, Q. Zhang, and Z. He. Investigating the properties of indicators and an evolutionary many-objective algorithm based on a promising region. *IEEE Transactions on Evolutionary Computation*, 2020.

[109] Y. Yuan, H. Xu, and B. Wang. An improved nsga-iii procedure for evolutionary many-objective optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 661–668, New York, NY, USA, 2014. ACM.

[110] Y. Yuan, H. Xu, B. Wang, and X. Yao. A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):16–37, 2015.

[111] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao. Balancing convergence and diversity in decomposition-based many-objective optimizers. *IEEE Transactions on Evolutionary Computation*, 20(2):180–198, 2015.

[112] J. Zhang and L. Xing. A survey of multiobjective evolutionary algorithms. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, pages 93–100, July 2017.

[113] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[114] X. Zhang, Y. Tian, R. Cheng, and Y. Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):97–112, 2018.

[115] X. Zhang, Y. Tian, and Y. Jin. A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(6):761–776, 2014.

[116] A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm & Evolutionary Computation*, 1(1):32–49, 2011.

[117] Y. Zhou, M. Zhu, J. Wang, Z. Zhang, Y. Xiang, and J. Zhang. Tri-goal evolution framework for constrained many-objective optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(8):3086–3099, 2020.

[118] Q. Zhu, Q. Zhang, and Q. Lin. A constrained multiobjective evolutionary algorithm with detect-and-escape strategy. *IEEE Transactions on Evolutionary Computation*, 24(5):938–947, 2020.

[119] H. Zille. *Large-scale Multi-objective Optimisation: New Approaches and a Classification of the State-of-the-Art*. PhD thesis, Otto von Guericke University Magdeburg, 2019.

[120] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima. A framework for large-scale multiobjective optimization based on problem transformation. *IEEE Transactions on Evolutionary Computation*, 22(2):260–275, 2018.

[121] H. Zille and S. Mostaghim. Linear search mechanism for multi- and many-objective optimisation. In *EMO*, 2019.

[122] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[123] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*, pages 832–842. Springer, 2004.

[124] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[125] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132, 2003.

List of Figures

List of Tables

Glossary

CDTLZ convex DTLZ
CMOEA constrained multi-objective evolutionary algorithm
CMOP constrained multi-objective optimization problem
COS cloud offloading scheme

DAS-CMOP difficulty-adjustable and scalable CMOP
DOC decision and objective constraints
DTLZ Deb, Thiele, Laumanns and Zitzler

EA evolutionary algorithm
EOS edge offloading scheme

GD generational distance

HV hypervolume

IDTLZ inverted DTLZ
IGD inverted generational distance
IoT Internet of Things

LAN local area network
LIR-CMOP CMOP with large infeasible regions
LOS local offloading scheme
LSMOP large-scale multi-objective optimization problem

MCC mobile cloud computing
MD mobile device

MEC mobile edge computing
MOP multi-objective optimization problem
MaOP many-objective optimization problem
MOEA multi-objective evolutionary algorithm

PF Pareto front
PPS push and pull search

QoS quality of service

RAN radio access network
RBM restricted Boltzmann machine
ROS random offloading scheme

SDTLZ scaled DTLZ
SMOP sparse multi-objective optimization problem
SZDT scaled ZDT

WAN wide area network

ZDT Zitzler, Deb and Thiele

List of Publications

Peng, G., Wu, H., Wu, H., & Wolter, K. Constrained Multi-objective Optimization for IoT-enabled Computation Offloading in Collaborative Edge and Cloud Computing. *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2021.3067732, 2021.

Peng, G., Wu, H., Wu, H., & Wolter, K. Evolutionary Large-scale Sparse Multi-objective Optimization for Collaborative Edge-cloud Computation Offloading. In *Proceedings of the 12th International Joint Conference on Computational Intelligence*, 1:100-111, 2020.

Peng, G., & Wolter, K. A Novel Archive Maintenance for Adapting Weight Vectors in Decomposition-based Multi-objective Evolutionary Algorithms. In *2020 IEEE Congress on Evolutionary Computation* (pp. 1-8). IEEE, 2020.

Peng, G., & Wolter, K. A Decomposition-Based Evolutionary Algorithm with Adaptive Weight Vectors for Multi- and Many-objective Optimization. *Applications of Evolutionary Computation-23rd European Conference, EvoApplications 2020*. Springer, 2020. (Outstanding Students of the EvoStar 2020)

Wu, H., Shang, Z., **Peng, G.**, & Wolter, K. A Reactive Batching Strategy of Apache Kafka for Reliable Stream Processing in Real-time. In *31st IEEE International Symposium on Software Reliability Engineering (ISSRE 2020)* (pp. 207-217). IEEE, 2020.

Shang, Z., Wu, H., **Peng, G.**, & Wolter, K. Dynamic Load Balancing in the Control Plane of Software-Defined Networks. In *19th IEEE International Conference on Communication Technology (ICCT 2019)* (pp. 947-953). IEEE, 2019.

Peng, G.. Multi-objective Optimization Research and Applied in Cloud Computing. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 97-99). IEEE, 2019.

Peng, G., Shang, Z., & Wolter, K. A Multiobjective Artificial Bee Colony Algorithm based on Decomposition. In *Proceedings of the 11th International Joint Conference on Computational Intelligence* (pp. 188-195). SciTePress, 2019.

Peng, G. & Wolter, K. Efficient Task Scheduling in Cloud Computing Using an Improved Particle Swarm Optimization Algorithm. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)* (pp. 58-67). SciTePress, 2019.

Peng, G., Wu, H., Wu, H., & Wolter, K. Dynamic and Secure Multi-Objective Computation Offloading for IoT-Edge-Cloud Orchestrated Computing Networks. *Unpublished*, 2021.

About the Author

Guang Peng received both his bachelor and master degrees from Xi'an Jiaotong University, China in 2014 and 2016, respectively. He joined in Dependable Distributed Systems group of Freie Universität Berlin, Germany, supervised by Prof. Dr. Katinka Wolter since 2018. His research interests include multi- and many-objective optimization, intelligent computation, deep learning and mobile cloud/edge computing.