Exact and Approximation Algorithms for the Planning and Design of Fog Networks

by

Decheng Zhang

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
January, 2018

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Exact and Approximation Algorithms for the Planning and Design of Fog Networks

submitted by **Decheng Zhang**

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Professor Marc St-Hilaire, Thesis Supervisor

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
December, 2018

Abstract

As a promising computing architecture, fog computing has attracted lots of attention from industry and research communities in recent years. With the huge number of heterogeneous and distributed devices performing computational and storage tasks between the cloud and users, fog computing can be an answer to the surging challenges in today's networks. However, due to the decentralized and heterogeneous nature of fog networks, planning a fog network can be a complicated and challenging task. To our best knowledge, little work has been done on the planning and designing of fog computing networks. To deal with this problem, we first propose a multi-objective mathematical model that simultaneously deals with the fog node placement, fog node dimensioning and demand routing. The model optimizes the tradeoff front (Pareto front) between capital expenditure and network delay in dual objective functions. Then, we analyze the performance of an exact algorithm (branch and bound) and two evolutionary algorithms (genetic algorithm and particle swarm algorithm) on this problem, showing that the evolutionary algorithms offer a good balance between the Pareto optimality and computation time efficiency. Inspired by the existing evolutionary algorithms, we proposed a new evolutionary algorithm, named PSONSGA, which combines the convergence efficiency from NSGA-II and the searching efficiency from SMPSO. The results demonstrate that the evolutionary algorithms are highly efficient compared to the exact algorithm. Among the three evolutionary algorithms, the algorithm we proposed (PSONSGA) gives the best Pareto front solutions which shows the good convergence to the true optimal front and the evenly distribution character. The proposed algorithm can be a valuable planning tool for real-world fog network planning.

I would like to dedicate my thesis to my parents and my grandparents.

Acknowledgments

I would like to express my deep appreciation to my supervisors Prof. Marc St-Hilaire for his guidance, support, generosity, motivation and encouragement during the past two years. Without Prof. Marc St-Hilaire's appreciation, I would not have had the chance to be a member of a prestigious research group that I have benefited so much from.

I would like to thank Dr. Christian Makaya for his guidance, suggestions and support throughout the whole period of my research.

Sincere thanks to my colleague Faisal Haider for his support in both my research and my personal life.

I would also like to thank my friends Dr. Ammar Alhosainy, Dr. Muhammad Raisul Alam and Zhe Zhang for giving me consistent help, guidance and encouragement.

On the personal side, I want to thank my family for their unconditional love.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
CC	Cloud Computing
CDI	Crossover Distribution Index
CFLP	Capacitated Facility Location Problem
CP	Crossover Probability
CPE	Count-Preserving Encoding
DC	Distributed Cloud
EMO	Evolutionary Multi-objective Optimization
FC	Fog Computing
FPP	Fog Planning Problem
GA	Genetic Algorithm
HV	HyperVolume
IGD	Inverted Generational Distance
IoT	Internet of Things

IT	Information Technology
MC	Multi-Cloud
MDI	Mutation Distribution Index
MDC	Micro Datacenters
MIP	Mixed-Integer Programming
MO	Multi-Objective
MOO	Multi-Objective Optimization
MOEA	Multi-Objective Evolutionary Algorithm
MOCO	Multi-Objective Combinatorial Optimization
MP	Mutation Probability
NE	Non-supported Efficient
NSGA-II	Non-dominated Sorting Genetic Algorithm II
PSO	Particle Swarm Optimization
PSONSGA	Particle Swarm Optimized Non-dominated Sorting Genetic Algorithm
QoS	Quality of Service
SDN	Software-Defined Networking
SE	Supported Efficient
SLA	Service-Level Agreement
SMPSO	Speed constrained Multi-objective Particle Swarm Optimization
TCP/IP	Transmission Control Protocol/Internet Protocol

UFLP	Uncapacitated Facility Location Problem
vCPU	virtual Central Processing Unit
VM	Virtual Machine

Chapter 1

Introduction

Cloud computing has been dominating the Information Technology (IT) industry for several years. As a computing model, cloud computing enables convenient, on-demand network access to a shared pool of computing resources. This centralized computing model has been used as an efficient way to process data and deploy business services. However, the rapid evolution in the fields of mobile applications and wearable devices imposes stringent delay and reliability requirements in today's cloud systems [1]. In addition, the Internet of Things (IoT) introduces new challenges that cannot be addressed by the current cloud system and could lead to serious issues in terms of performance, reliability and security. For instance:

- *Device Ubiquity Challenge*: Cisco estimates that there will be 50 billion connected devices by 2020 [2]. The traffic and data generated by these new devices will burden the backbone and access networks.

- *Service/Network Management Challenge*: Immersion in billions of devices can be helpful to improve application procedures. Nonetheless, managing a network of billions of heterogeneous devices would be complex and challenging.

- *Connectivity Challenge*: When there are billions of devices consuming and generating data at the edge of the network, the current network standards may fall short [3]. Resource contention may happen on the networks due to the huge amount of traffic generated by these devices.

- *Security Challenge*: Information leaking across platforms and interfaces has been the primary concern in cloud computing to date. Moreover, in some countries, regulations prevent organizations from storing their information in foreign

locations.

In order to tackle these challenges, Cisco proposed the paradigm of fog computing as a potential solution [2]. The idea of fog computing is to bring the computation, communication, control and storage closer to the end-users; therefore, the data transfer time and the amount of network transmissions will be greatly reduced. Typically, the fog architecture can be described as shown in Figure 1.1. As can be seen, the fog architecture is deployed as a large number of Micro-DataCenters (MDCs), which are often distributed over a geographical region of interest. These fog nodes connect to access networks and remote clouds to provide storage and computing services without the intervention of third-parties [4].

The proximity offered by fog networks addresses the challenges facing today's cloud centre, and enables low-latency applications such as augmented reality, gaming, video streaming, etc.

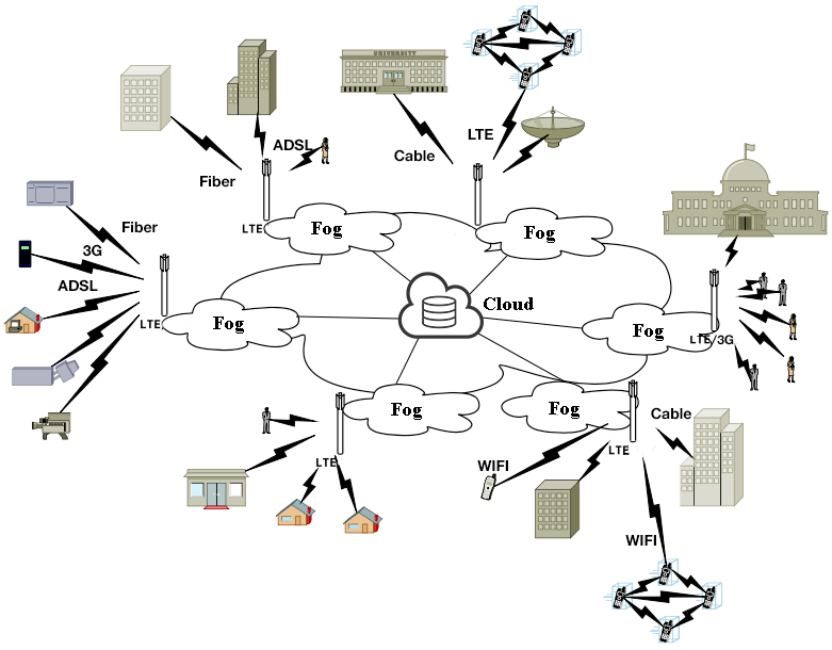

Figure 1.1: Typical fog network architecture

However, planning fog networks can be a complicated task due to the following reasons:

- Fog facilities need to be highly decentralized.

- The fog architecture needs to take into account heteromerous hardware requirements.
- User demands need to be distributed across multiple facilities.
- The center cloud needs to be able to connect and cooperate with the fog facilities.

With the difficulties mentioned above and the complexity of future networks, service providers need efficient planning tools to implement fog computing networks. These efficient network planning tools can help design optimal fog networks guaranteeing better service to the end users.

1.1 Problem Statement

The primary goal of the fog network planning is to generate an optimal fog network design fulfilling the Quality of Service (QoS) requirements of latency-critical services and IoT devices. Without loss of generality, it can be recognized that the fog nodes will be the distributed fog facilities delivering computation, storage, and networking services at the edge of the network. Considering the decentralized and heterogeneous nature of the fog network combined with a vast number of edge devices, obtaining the optimal network design can be challenging. In fact, several factors have to be taken into consideration such as the decentralized resources, heterogeneous facilities, ubiquitous IoT devices and so on. Essentially, the fog network planning problem can be decomposed into three subproblems: the fog placement problem, the fog dimensioning problem, and the demand routing problem.

- **Fog placement** is the process of selecting a subset of potential locations from a given candidate set and placing fog facilities at these locations. The geographical factors could significantly affect the propagation delay in the fog network.
- **Fog dimensioning** is the process of selecting the capacity levels of fog devices and links for each candidate site. This selection depends on the fog node price, link price and demand requirements.
- **Demand routing** is the process of directing requests to an appropriate fog node. This problem can be modelled as an assignment problem. Once the fog

placement and dimensioning decisions are made, the assignment problem must be solved to optimally direct requests. Since the location of the users will also have an impact on the fog placement, all these three problems are dependent on each other.

All these problems are of particular importance to the implementation of fog networks, and have significant impacts on the fog network performance. The fog placement and fog dimensioning problems correspond to the facility placement problem, while the demand routing problem constitutes the resource allocation problem. Unfortunately, each of these problems has already been proven to be NP-hard [5, 6].

Different approximation algorithms have been proposed to solve these optimization problems. However, due to the complexity, most of them are problem dependent and lacking flexibility to the real-world network planning. For example: the traditional Facility Location Problem (FLP) aims to determine the minimum capital expense to achieve the specific network performance. Without a priori knowledge of network objectives, it is difficult to establish an appropriate service-level constraint. Another classic problem, the K-median clustering, deals with minimizing the delay parameter between the facility and clients. In real-world planning, however, without an accurate estimation of K (number of fog nodes), the value of this model is undermined.

On the other hand, from the operator point of view, the fundamental goal of the fog network planning is to achieve the maximum network performance with minimum investment. This can be targeted into two objectives:

- **Minimizing the Capital Expenditure:** Budget and monetary factors have always been the major concerns for network designers. When implementing a fog network, it is important to identify and attempt to minimize the total capital expenditure such as the hardware purchasing, facility rental, link cost and installation cost, etc.

- **Minimizing the Delay:** Delay is arguably the most important performance parameter for most cloud services. A small increase in the user-perceived latency can cause substantial revenue loss of the service provider [7]. Also, since the fog is targeted for widely distributed IoT devices, an effective fog network should consider the latency requirements of the latency-critical IoT services.

As we can see, the two objectives are conflicting with each other. For instance, to minimize the network delay, a large number of fog nodes may need to be deployed, thus increasing the capital expense. On the other hand, to minimize the capital expense, only a small number of fog nodes may be selected. In this case, requests and demands will not be fully served by the fog nodes and will have to be routed to the distant cloud across the public network, thus degrading the network performance.

Optimizing two objectives that conflict with each other converts the problem into a multi-objective optimization problem. The multi-objective optimization aims to find the Pareto front which consists of a set of non-dominated trade-off solutions. The Pareto front gives network designers a chance to evaluate the pros and cons of each of the trade-off solutions, thus, leading to informed network planning decisions. However, it is difficult to choose an appropriate multi-objective approach due to the inability to derive an analytical knowledge of the problem.

Different multi-objective algorithms have been proposed by previous researchers. For example, the multi-objective fog planning problem can be solved by a linear combination of different objective functions to form a single objective function (weighted sum method) using a commercial solver such as cplex. However, it cannot be solved within a reasonable amount of time due to the NP-hardness.

The multi-objective fog planning problem can be solved by the Evolutionary Multi-objective Optimizations (EMOs) [8, 9]. Since EMOs are heuristic-based algorithms, there is no guarantee in finding Pareto-optimal points [10]. However, EMOs have effective operators to evolve the solution points towards the optimal frontier. For example, in [11], the EMO algorithm was proposed to solve the demand scheduling problem, while in [12] the EMO was applied to solve the facility location problem. Therefore, it is interesting to investigate how they perform on the fog network planning problem.

In conclusion, designing and planning fog networks is characterized by the complexity of the fog placement, fog dimensioning, and demand routing whilst minimizing the capital expenditure and network delay. Tackling these complexities, this thesis addresses the above mentioned problems and proposes exact and approximation multi-objective algorithms for the fog network planning problem.

1.2 Research Objectives

The main objective of this thesis is to develop efficient planning tools to solve the fog network planning problem. More precisely, we will achieve the following objectives:

- Propose a mathematical model for the fog network planning problem, which formulates the joint problem of fog placement, fog dimensioning, and demand routing. The objectives of this model are to minimize the capital expenditure and minimize the network delay.
- Solve the previous model using the weighted sum method. The optimal solution points obtained will be used as a benchmark to compare the quality of various approximation algorithms.
- Implement and evaluate two existing multi-objective evolutionary algorithms, namely NSGA-II and SMPSO.
- Propose a new multi-objective evolutionary algorithm.
- Compare the performance of the previous four different approaches in terms of multi-objective quality indicators, such as Hypervolume (HV) and Inverted Generational Distance (IGD).
- Compare the performance of the three approximation algorithms with the optimal solution points obtained from the weighted sum method.

1.3 Methodology

This section explains the methodology that will be used to meet our research objectives.

- **Develop the mathematical model:** To address the first objective (propose a mathematical model for fog planning problem), we will first study the reasons why the concept of fog computing was introduced and we will also look at the fog infrastructure. Based on this information, we will define the objective functions and derive a set of constraints that adequately represents the reality of fog networks.

- **Solve the model using the exact algorithm:** To address the second objective, the model from previous step will be used to solve the fog planning problem. The weighted sum method will be applied to combine multi-objective functions into single-objective function, then the single-objective function will be solved by the commercial solver CPLEX.

- **Solve the model using the existing EMOs:** To address the third objective, the fog planning problems will be solved by two prominent MOEAs (NSGA-II and SMPSO). An encoding scheme will be designed to translate the fog planning problem into evolutionary algorithms.

- **Evaluate the MOEAs and propose a new MOEA:** To address the fourth objective, the NSGA-II and SMPSO will be evaluated in terms of convergence and diversity. Motivated by the shortcomings in these two algorithms, we will propose a new multi-objective algorithm (PSONSGA). The proposed MOEA exploits the convergence and diversity quality in the genetic algorithm and the particle swarm algorithm.

- **Solution comparison in terms of the quality indicator:** To address the fifth objective, the Pareto front obtained from the weighted sum method, NSGA-II, SMPSO and PSONSGA will be evaluate in terms of two multi-objective quality indicators: HV and IGD. This will help us in determining which method generates the best approximate Pareto front.

- **Solution comparison in terms of the delay:** To address the sixth objective, the solution points obtained from CPLEX will be used to evaluate the quality of the solutions obtained from the NSGA-II, SMPSO and PSONSGA. This will help us in determining which approximation algorithm generates the best solutions regarding to the delay parameter.

1.4 Thesis Outline

This thesis is structured as follows.

- Chapter 2 - Background & Related Work. We introduce the basic concepts of the fog computing, network design process and multi-objective optimization.

Then, we overview some related works to the fog network, distributed-cloud planning, multi-cloud planning, and multi-objective optimization.

- Chapter 3 - Formulation of the Fog Planning Problem. This chapter defines and formulates the fog planning problem using a multi-objective mathematical model.

- Chapter 4 - Algorithms for Solving the Fog Planning Problem. This chapter presents a comprehensive explanation of the multi-objective algorithms and optimization procedures. The multi-objective algorithms include the weighted sum method, NSGA-II algorithm, SMPSO algorithm and proposed PSONSGA algorithm.

- Chapter 5 - Results and Analysis. This chapter presents the simulation results for the four multi-objective algorithms, including the evaluation and comparison in terms of the solution quality and computation time.

- Chapter 6 - Conclusion and Future Work. This chapter summarizes the research results and proposes future research directions.

Chapter 2

Background & Related Work

Fog computing has recently been a subject of great interest. As an infrastructure extending the cloud service to the edge of the network, fog networks can effectively improve network performance. However, due to the decentralization and geo-distribution of fog nodes, planning and designing a fog network is a challenging task. Therefore, an efficient network planning tool is necessary to design optimal fog networks.

In this chapter, we present several important concepts related to fog computing, network design process and multi-objective optimization. Then, we overview some related works in the area of fog computing, distributed cloud and multi-cloud. Since the fog planning problem is casted as an optimization problem, Section 2.2.4 and Section 2.2.5 present respectively, the single-objective and multi-objective optimization studies related to the fog planning problem.

2.1 Background

In this section, we first introduce some background of fog computing in Section 2.1.1. Then, we give an overview of the network design process in Section 2.1.2. In addition, Section 2.1.3 presents the fundamental concepts of Multi-Objective Optimization (MOO). Finally, Section 2.1.4 explains the basic procedures and optimization processes in the Evolutionary Multi-objective Optimization (EMO).

2.1.1 Fog Computing

A Fog Computing (FC) network is a geographically distributed platform, where heterogeneous, ubiquitous and decentralized devices perform storage and processing tasks at the edge of the network. A three-tier hierarchy, as shown in Figure 2.1, demonstrates how fog nodes provide services for edge devices. In this platform, applications and workloads are run over a local fog network infrastructure instead of connecting to the remote cloud. This closeness to end-users creates an automated response to the challenges observed in the centralized cloud computing. For instance:

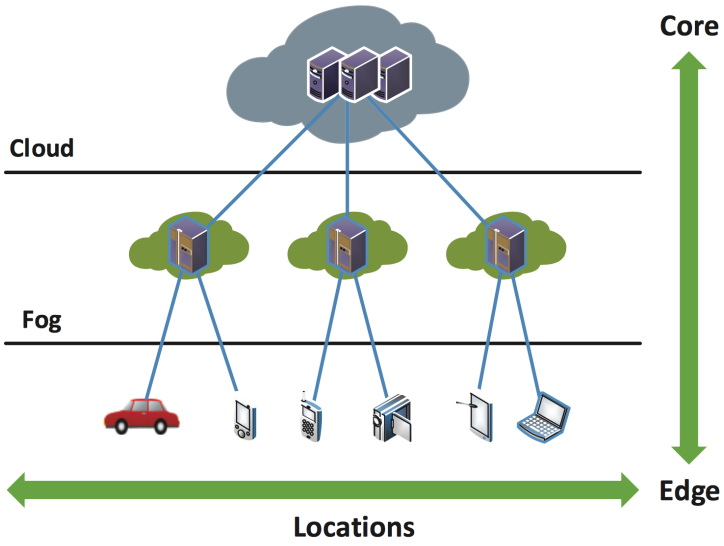

Figure 2.1: The fog multi-tier architecture [13]

Device Ubiquity Challenge: Fog networks can be deployed closer to edge devices at or above the access network. Therefore, it makes latency-sensitive applications possible. It is also possible to alleviate the traffic burden through deploying caching services in localized fogs [14], so that bulky data transfer can be sped up.

Service/Network Management Challenge: Fog technologies have been tested to ease this complexity [15, 16] . For instance, hosting services in fog nodes has been proved to be promising for auto-coordination of applications [15]. The asymptotic/declarative approach [16] can be applied at the fog level to achieve the service level management.

Connectivity Challenge: Fog networks can be beneficial to network and communication protocols, and offers low resource consumption and failure resilience [17]. The data locality makes it possible to aggregate and preprocess the data before sending; therefore, it avoids the potential congestion at the edge and boosts transmission efficiency.

Security Challenge: In fog networks, instead of sending data to centralized cloud-centres, fog nodes keep the privacy data within the local network. This localized manner keeps sensitive information secure from theft and vulnerability in today's cloud systems.

Although, FC has been deemed as an evolution of the current cloud model, it is not meant to "replace" Cloud Computing (CC). FC fills the technology gaps by supporting new services with its distributed computing, management, storage and networking capability [4]. In order to achieve a better performance, it could be installed between the cloud and the network edge, and between computing power and ubiquitous Internet of Things (IoT) devices [18]. Fog and cloud complement each other to form a service continuum between application and users.

From an Information and Communication Technologies (ICT) perspective, fog is not constrained to a particular technological area. To be specific, fog is the integration of a set of developed and mature technologies: cloud, sensor networks, peer-to-peer networks, network virtualization functions and the configuration management [15]. Vaquero et al., in [15], presented a thoroughly study of linking all these technologies together in the fog computing paradigm.

2.1.2 Network Design Process

Network design is a complex and iterative process. It includes the following steps [19]:

Declare the network requirements: This process requires a complete information of traffic loads, traffic types, and traffic paths. Then, this information is used to estimate the network capacity which in turn is used as the input to the next step.

Network design and planning: This process applies different design techniques and algorithms to produce a network topology. The decision variables include link and node placements, traffic routing decisions, facility dimensioning, etc.

Performance analysis: Once the previous steps are done, a candidate network solution is developed. In the performance analysis process, the candidate solution is

analyzed, so that different performance metric such as cost, reliability, and delay can be evaluated [19].

When these three steps are completed, the first design iteration is finished. Then the entire process is repeated, either with revised input data or by using a new design approach. The flowchart of this process is presented in Figure 2.2.

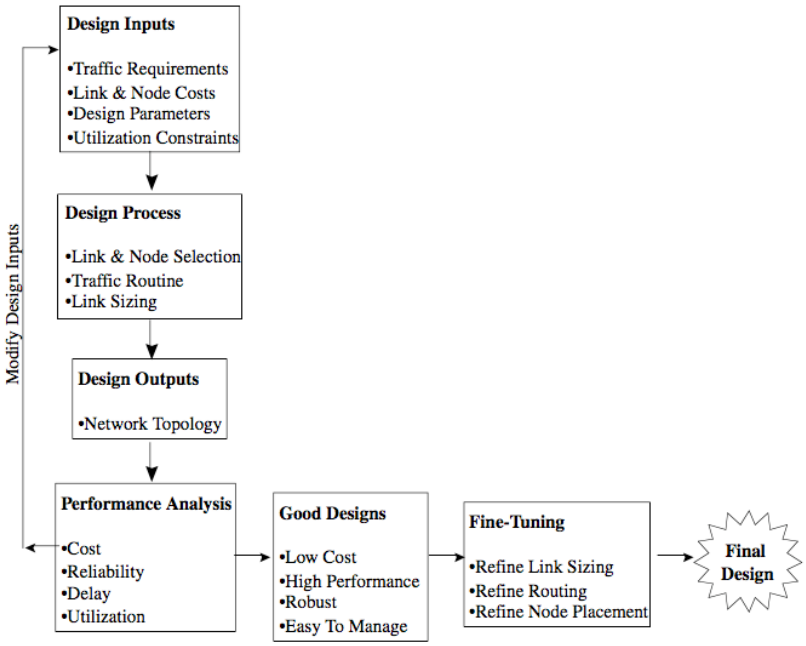

Figure 2.2: Overview of the network design process [19]

The essence of this iterative process is to provide optimal network planning schemes [19]. Unfortunately, most real world design problems can not have a mathematical perspective to obtain the optimal network solution. The network designer must use a form of trial and error to determine the best option. After investigating a variety of designs, the designer can select the one that provides the best performance at the lowest cost. The inability to derive an analytically perfect solution requires the network designer to explore as many alternatives as possible [19]. The multi-objective optimization we applied in this work is perfect for this job. The multi-objective optimization generates the trade-off solution set in which all the solutions are optimal. The richness of trade-off solutions, also known as Pareto frontiers, gives a chance to the decision maker to evaluate the pros and cons of each of these solutions based on all technical/non-technical considerations. In other words, it provides a complete

image of the whole objective space through the trade-off solution results. Therefore, in a multi-objective optimization, we put effort on finding the optimal set of trade-off solutions and take all the objectives into considerations.

2.1.3 Multi-Objective Optimization

As the name suggests, multi-objective optimization involves optimizing dual or multiple objectives simultaneously. The problem becomes challenging when the objectives conflict with each other, that is, the optimal solution of an objective function is different from that of the other. In other words, the solution has to compromise at least one of the other objectives to improve in this objective. Therefore, there does not typically exist a feasible solution that minimizes all objective functions simultaneously.

Pareto dominance: In a minimization problem, a vector $\vec{x^*} = (x_1^*, ..., x_n^*)$ Pareto dominates $\vec{x}$ ($\vec{x^*} \succeq \vec{x}$), if and only if x^* is not worse than x in all k objectives ($f_i(x^*) \leq f_i(x)\ \forall i = 1, ..., n$) and x^* is strictly better than x in at least one objective ($f_i(x^*) < f_i(x),\ i = 1, ..., n$) [20]. For a given set of solutions, all points which are not dominated by any other member of the set are called the non-dominated points. For example, in Figure 2.3, points 3, 5 and 6 comprise the non-dominated front.

Pareto-optimal front: The set X consisting of all non-dominated solutions x^* in the search space is called Pareto-optimal front. We call a set of non-dominated solutions that approximate the Pareto optimal front as Pareto front or known Pareto front [20].

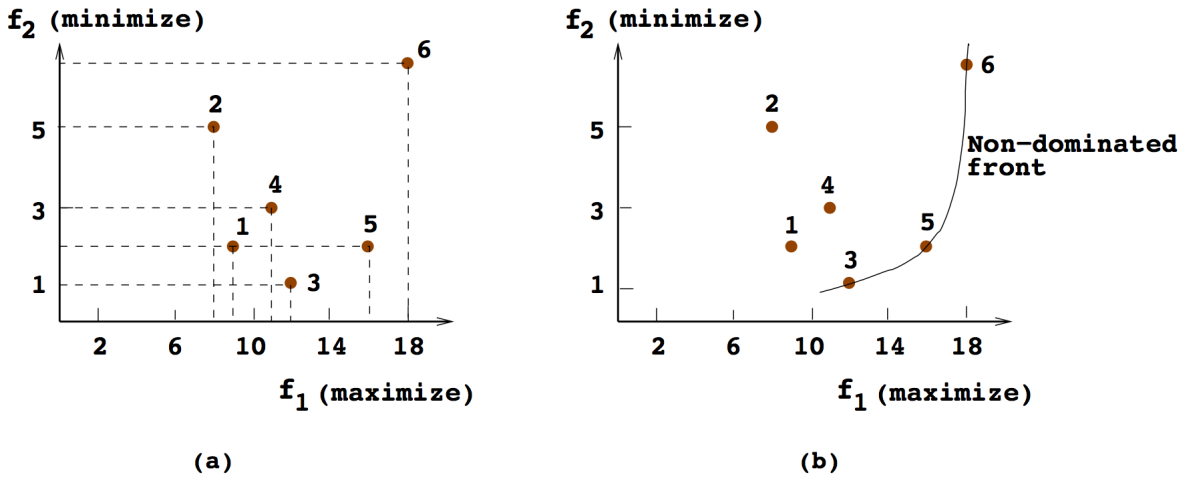

Figure 2.3: Points and non-domination front [20]

Multi-Objective Combinatorial Optimization (MOCO) is a special type of MOO problem that aims at finding a grouping, ordering, or assignment of a discrete, finite set of objects that satisfies given conditions. By its nature, multi-objective combinatorial optimization deals with discrete, non-continuous problems [21]. The fog planning problem addressed in this thesis belongs to the MOCO category. We will give detailed formulations of the fog planning problem in Chapter 3. An essential consequence of the discrete structure when trying to optimize the model is that it is not sufficient to aggregate the objectives through the weighted sum method. The study in [21] has proved that the discrete structure of the MOCO revokes the validity of LP completeness. Thus, there usually exist solutions that are not optimal for any weighted sum of the objectives. These solutions are called Non-supported Efficient (NE) solutions, whereas the remaining are called Supported Efficient (SE) solutions. Usually, there are more NE solutions than SE solutions [22]. We will show in Chapter 5 that our results directly proves this statement. All these NE solutions which conform to the Pareto-optimal front, are vital for trade-off analysis and decision making. Therefore, instead of the weighted sum method, we choose Evolutionary Multi-objective Optimization (EMO) to solve the fog planning problem.

2.1.4 Evolutionary Multi-Objective Optimization

Evolutionary Multi-objective Optimization (EMO) is based on heuristic multi-objective optimization techniques that imitate the principles of natural selection

and survival of the fittest to find near-optimal solutions [23]. In EMO, multiple Pareto-optimal solutions are found in a single simulation by emphasizing multiple non-dominated and isolated solutions. Since EMOs are heuristic based algorithm, they may not guarantee in finding Pareto-optimal points. Nonetheless, EMO procedures have essential operators to continually evolve solution points similar to the way most natural evolving systems strengthen their solutions [10].

Genetic Algorithm for MOCO Problems

The first class of evolutionary algorithm is Genetic Algorithm (GA). GA is a population-based meta-heuristic techniques. GA iteratively evolves a set of encoded solutions to find the best optimal solution. A set of randomly initialized solutions are first encoded as binary or integer vectors (known as encoding scheme). Then, the GA iteratively runs the selection, crossover, and mutation on these solutions to obtain (hopefully) better solutions. The fitness function is defined to quantify the optimality of solutions. The parent selection is based on the individual fitness. The higher the fitness level is, the higher the probability that the individual will be selected to reproduce. The whole process stops when the predefined conditions are met. The pseudocode of the genetic algorithm is shown in Algorithm 1.

Algorithm 1 Generalized Genetic Algorithm

```
1: N ← population size
2: P ← initialized parent population by randomly creating N individuals
3: while stop condition is not met do
4:     C ← empty child population
5:     while not enough individuals in C do
6:         parent1 ← select parent (by tournament selection)
7:         parent2 ← select parent (by tournament selection)
8:         child1, child2 ←crossover(parent1, parent2)
9:         mutate ( child1, child2)
10:        evaluate the fitness of child1, child2
11:        insert child1, child2 into C
12:    end while
13:    p ← combine P and C to get N individuals for next generation
14: end while
```

Particle Swarm Algorithm for MOCO problems

The second class of evolutionary algorithm is Particle Swarm Optimization (PSO). PSO is a stochastic population-based algorithm. PSO was introduced by J. Kennedy and R. Eberhart in [24]. It is inspired by the social behaviour of bird flocking or fish schooling.

The basic PSO imitates a swarm of S particles (i.e. potential solutions), which fly through a D-dimensional problem search space in search of the global optimum position that produces the best fitness value.

First, we assign each particle with a random position x and a velocity v_i ($i = 1, 2, ..., S$). In every iteration, each particle adjusts its velocity according to the historical best and global best solution. The PSO algorithm updates the particle's position by:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{2.1}$$

and the velocity function $\vec{v}_i(t)$ is given by:

$$\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + C_1 \cdot r_1 \cdot (\vec{x}_{p_i} - \vec{x}_i) + C_2 \cdot r_2 \cdot (\vec{x}_{g_i} - \vec{x}_i) \tag{2.2}$$

The parameters, C_1 and C_2 are two positive learning factors.

For MOCO problems, the fitness function needs to fit the multi-objective context. The redefined constrained-based fitness function for two solutions x^i and x^j is as follows.

Definition A solution x^i is said to "constrain-dominate" a solution x^j (or $x^i \preceq x^j$) if any of the following conditions are true:

1. Solution x^i is feasible and solution x^j is not.

2. Solution x^i and x^j are both infeasible, but solution x^i has a smaller constraint violation, which can be computed by adding the normalized violation of all constraints.

3. Both solution x^i and x^j are feasible and solution x^i dominates solution x^j in the sense of Pareto dominance.

The detailed demonstration of solving the fog planning problem by using MOCO is presented in Chapter 4.

2.1.5 Multi-Objective Solution Quality Indicator

Assessing the solution quality in multi-objective optimization is complex compared to single objective optimization. For the single-objective problem, one can assume that any solution giving the minimal (or maximal) solution is preferable for minimization problems (or maximization problems). However, in multi-objective optimization, the output is always an approximation set. Without the help of an efficient quality indicator, we cannot evaluate the output between various heuristic or meta-heuristic methods.

In previous studies, several quality indicators have been proposed: Hypervolume [25], Epsilon [26], IGD [27] and R2 [28], to name a few. In the context of network design, an efficient multi-objective planning algorithm should explore as many alternatives as possible, thus giving service providers a rich set of planning options. In addition, each candidate solution (Pareto frontier) inside the solution set (Pareto front) is expected to be as close to the optimal point as possible thus guaranteeing the best network performance to the end users. Considering these criteria, in this work, we use the hypervolume indicator to assess the diversity (richness of solution sets) of the solution set, and the IGD indicator to assess the convergence (closeness to the optimal front) of the solution set.

Hypervolume Indicator

The hypervolume indicator [29] calculates the volume covered by members of a non-dominated set of solutions in the objective space. As shown in Figure 2.4, the union space of all hypercubes constructed by reference point W and each solution is defined as hypervolume. Since hypervolume is an indicator that covers both convergence and diversity, this indicator is of high importance for practical evaluation. In Chapter 5, we employ the hypervolume indicator to evaluate and compare the solution qualities of three evolutionary algorithms. An algorithm that produces a higher hypervolume value is preferred [29].

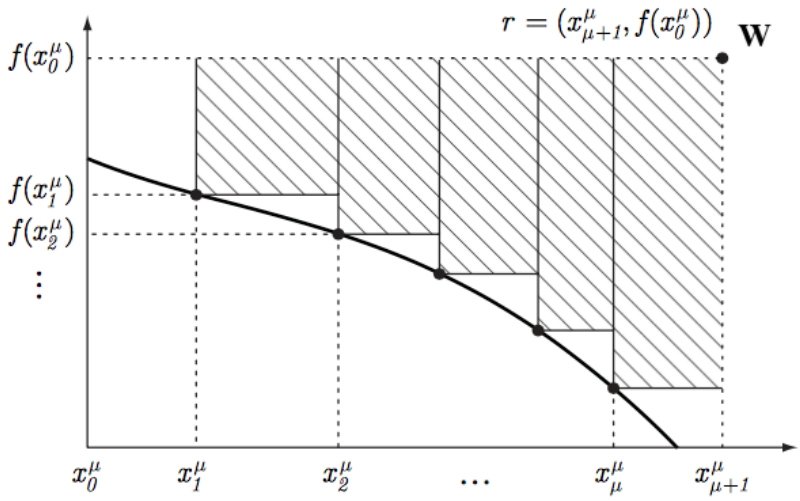

Figure 2.4: Hypervolume indicator [29]

IGD Indicator

The IGD indicator evaluates the quality of an objective vector set using a reference point set. The IGD indicator can be written for a non-dominated objective vector set $A = \{a_1, a_2, ...a_{|A|}\}$ and the reference point set $Z = \{z_1, z_2, .., z_{|Z|}\}$ [30].

$$IGD(A) = \frac{1}{|Z|}\left\{\sum_{i=1}^{|Z|} d_j^p\right\}^{1/p} \tag{2.3}$$

where d_j is the Euclidean distance from z_j to its nearest points in A. Since the IGD indicator represents the mean distance to the reference front, an lower IGD value means that the solution set is of high quality.

2.2 Related Work

Fog computing research has attracted lots of attention. Researchers have been working on studies of fog architecture [31], fog communication protocols [32] and application logic in fog networks [33]. However, to the best of our knowledge, there is little work dealing with the planning and design aspects of fog networks. Among the studies pertaining to the fog, we review some notable ones (Subsection 2.2.1) and hope this will give our readers some insight on fog studies. Two closely related topics

under the network planning context are the distributed-cloud and multi-cloud network planning. We give an overview of these two related works in Subsections 2.2.2 and 2.2.3. Moreover, since this thesis focuses on the fog/facility planning optimization, we report some state-of-the-art operations research in the single-objective (Subsection 2.2.4) and multi-objective optimization (Subsection 2.2.5) that relates to the Facility Location Problem (FLP).

2.2.1 Fog Computing

Ever since Cisco proposed the concept of fog computing, it has continually attracted a large number of publications. Researchers started to explore the potential technologies, services and applications that can benefit from the fog.

Bonomi et al. [34] discussed seven unique characteristics of the fog. All these characteristics make the fog an appropriate platform for IoT applications. The authors also stated that fog networks are beneficial to Wireless Sensors and Actuators Networks (WSANs). In another study, Stojmeovic et al. [13] discussed the advantage of fog computing over cloud computing in various IoT and SDN scenarios. They further emphasized that FC is a system-level horizontal architecture. This architecture is able to distribute resources and services for computing, storage, control and networking anywhere along the continuum from Cloud to Things. Klas, in [35], presented a summary of the essential features in the FC network: wide-spread and geographically available in large numbers, supporting mobile devices and IoT context specialized services. Vaquero et al. [15] gave a comparison between cloud computing and fog computing (FC) in terms of network infrastructures. Chiang in [36], discussed seven use cases in fog network: from network content provisioning to client-based HetNets, from shared bandwidth to smart traffic control. The possibilities, to leverage the massively distributed resources in the fog network to support data analytics system, have been discussed by authors in [34]. In another study, the scholars in [31], envisioned the FC in future smart cities and presented a hierarchical FC architecture to support the massive number of infrastructures and monitoring services.

Checko et al., in [37], illustrated an analogy technique to combine the fog computing into 5th Generation mobile networks (5G), which were named as the Cloud RAN technology (CRAN). Another study extended this concept further to Heterogeneous Cloud Radio Access Networks (H-CRAN) [38]. See the work by Peng et al. [32], for a complete overview of the concept of Fog-computing-based RAN.

Regarding workload allocation and resource management in fog computing networks, Dsouza et al. [39] proposed a policy-based resource management in fog computing, which expanded the current fog computing platform to support collaboration and interoperability between different resources. Deng et al., in [40], focused on investigating the tradeoff between power consumption and transmission delay in the fog-cloud computing system, and they formulated a work-load allocation problem that aims at minimal power consumption considering the constrained service delay. Sun and Zhang [33] proposed a fog computing structure to integrate spare resources in the network applying a crowd-funding algorithm.

Among all the work describe above that are directly focusing on the fog computing environment, none of them addresses the planning and design of fog networks as we are presenting it in this thesis.

2.2.2 Distributed-Cloud Network Planning

The distributed clouds can be seen as a system in which geographically distributed resources are available to application developers. In this context, dynamic resource allocation and reallocation are important for accommodating unpredictable demands. Endo et al. [41] highlighted and categorized the main challenges inherent to the implementation of distributed-clouds. This research offered a stepwise view from the modelling phases to the optimization phases of the resource allocation in distributed system. However, their research didnot consider the facility placement and facility provisioning optimization.

Hwang et al [42] illustrated the planning processes of creating high-performance, scalable, reliable distributed computing systems including the design principles, architectures, and innovative applications.

Khosravi and Buyya [43] presented a taxonomy and classification of the existing techniques for resource management in achieving a green cloud computing environment.

Alhazmi et al. [44] introduced a comprehensive system to solve the problem of network mapping for a set of requests sent by cloud clients. The node and link provisioning were performed periodically. Their simulation results showed that their algorithm achieved the objectives in terms of generated revenue, served-connection ratio, resource utilization and computational overhead.

From the perspective of virtual machine placement, Zhang et al. [45] proposed a

clustering-based virtual machine placement algorithm. Their algorithm took full use of the topology and the density property of cloud network. The simulation showed that their algorithm was especially appropriate for large scale problems. In another study, Khosravi et al. [46] proposed a dynamic virtual machine placement approach considering energy and carbon cost in geographically distributed clouds. Their approach formulated the total energy cost using power usage effectiveness metric. The experimental results showed that their approach minimized the energy and carbon cost while still meeting cloud users' service level agreements.

Iturriaga et al. [11] studied the application of multi-objective evolutionary algorithms for solving the energy-aware scheduling problem. The scheduling problem took account of the scheduling of large workflows in a federation of datacenters. Their model maximized the service quality metrics and reduces the energy requirement for the computation. The experimental results demonstrated that the multi-objective evolutionary approaches were capable to compute the accurate scheduling results.

In another study, Xu and Li, in [47], presented a joint request mapping and response routing policy for geo-distributed cloud services. The utility functions were used to capture the performance goals. However, their model focused on the resource mapping and traffic routing in distributed cloud networks without considering the placement and sizing of the geo-distributed cloud. Also, Agarwal et al. in [48] designed a data placement system for the geo-distributed cloud, and Garg et al. [49] proposed an environment conscious scheduling scheme for the distributed network. Both of their studies [48, 49] focused on resource mapping and demands allocation in distributed-cloud networks, but the facility planning and provisioning were missing.

Among all the work describe above that are directly related to the distributed-cloud computing environment, none of them addresses the facility planning and provisioning in distributed computing networks as we are presenting it in this thesis.

2.2.3 Multi-Cloud Network Planning

Similar to fog network planning, the geographical complexity of the distributed network makes multi-cloud provisioning and dimensioning a complex task. Petcu [50] identified the state-of-the-art research in building multi-cloud, and discussed the enhancements that need to be done to the current solutions.

Stantchev and Schropfer [51] presented a planning policy to route business requests to multi-cloud infrastructures. Their approach imposed an on-demand service level

objective to specify the response time, traffic and service rate. However, this model only considered a single resource in the planning process.

Wang et al. [52] proposed an online resource allocation algorithm (NET) to improve the multi-cloud performance. However, their algorithm did not consider the financial cost, and they only solved the problem with a linear programming toolbox. In another study, Pascual et al. [53] proposed a multi-objective cloud infrastructure placement policy. This online policy supported traffic-awareness optimization. Nevertheless, the bi-objective function (communication cost and number of servers (K)) over-simplified the real-world deployment.

From the workload scheduling perspective, Chen et al. [54] proposed two heuristic algorithms to allocate streaming workflows to geo-distributed DCs with the goal of minimizing the communication and computation costs. Their model imposed a strict latency constraint. In another study, Bossche et al. [55] analyzed and proposed a binary integer program formulation of the scheduling problem. Their work proved that the linear programming solver is capable of producing tractable solutions for scheduling applications in the public cloud, but the high solving time made this approach less feasible in a hybrid cloud scenario.

Simarro et al. [56] proposed a scheduling model for optimizing virtual cluster placements across multiply cloud offers. Their evaluation was based on a real-world cloud environment, and the results showed that users' investment decreased when part of the virtual infrastructure was dynamically distributed among clouds instead of maintaining it in a fixed one.

From the load management perspective, Grozev and Buyya [57] proposed an adaptive provisioning and load distribution algorithm that optimized overall cost and response delays concerning regulatory constraints. Ismail et al. [58] proposed a self-adaptation architecture that operated in the multi-cloud environment. This model addressed the decentralized planning problem in load adaptation.

Wang et al., in [59], created a dynamic cloud service selection strategy named DCS. Their selection strategy adopted the methodology of cloud service brokers. Each cloud service broker managed some clustered cloud services, and performed the DCS strategy applying the adaptive learning mechanism. The objective was to dynamically optimize the cloud service selection and to return the best service result to the user.

Celesti et al. in [60] proposed the Cross-cloud Federation Model (CCFM) to model

the multi-cloud federation. The CCFM could be divided into four stages: discovery, matchmaking, authentication, and establishment. However, their work focused on optimizing the virtual machine management without taking account of the physical machine provisioning.

Among all the work describe above that are directly related to the multi-cloud computing environment, most of them focused on workload scheduling and resource allocation, while none of them addresses the planning and design of distributed computing networks as we are presenting it in this thesis.

2.2.4 Single-Objective Optimization Related to the Fog Planning Problem

The fog planning problem can be solved by using single-objective optimizing methods. Intuitively, there are two approaches to model the problem in a single-objective setting:

- First approach: Minimize the network latency and impose a monetary budget constraint.
- Second approach: Minimize the building cost of the fog infrastructure and impose a network performance constraint.

In combinatorial optimization research, the first approach can be described as a K-median clustering problem, while the second one can be described as a Facility Location Problem (FLP).

The K-median models the problem of finding a minimum cost clustering through applying an upper bound on the number of fogs (K) that can be deployed. The objective of the K-median is to minimize the total distance between each facility and clients. The distance between facility i and client j can reflect various network performances such as latency, hop counts, etc. There have been a number of approximation algorithms developed for the K-median clustering problem. The first constant factor approximation algorithm was created by Charikar et al. [61] based on LP-rounding. The current best (3+2/p)-approximation approach was due to Arya et al. [62] based on a local search approach.

FLP considers facility cost and connect cost into the objective function. Therefore, it reflects the total required budget. Minimizing the objective function is equivalent to

minimizing the capital expenditure required to build the network. The first constant factor approximation, 3.16-approximation guarantee, was due to Shmoys and Tardos [63], which is based on LP-rounding. The current best algorithm, achieving 1.61 guarantee, was due to Jain, Mahdian, and Saberi [64], which is based on greedy Primal-Dual. Additionally, Guha and Khuller [5] proved that it is impossible to get an approximation guarantee of 1.463 for the metric facility location problem, unless $NP \subseteq DTIME[n^{O(loglogn)}]$.

2.2.5 Multi-Objective Combinatorial Optimization

The research related to multi-objective optimization can be classified into two classes:

1. Classical preference-based method
2. Multi-objective evolutionary method

The former method includes the weighted sum approach, the compromised approach and the rotated weighted metric method [65]. Additionally, the ϵ-Constraint method and analytic hierarchy solve the multi-objective function by reformulating an objective as constraints [66]. Also, there is the Benson's method [67] that drives the solution point to the Pareto front through maximizing the difference between the solution points and the reference points. In addition, the value function method [68] employs the unconstrained auxiliary function to reach the unique root of the global optimization.

In the second method, evolutionary algorithms are applied to perform Pareto search [69–71]. The researchers in [10], presented a detail description and evaluations of various MOEAs. Their experimental results have showed that NSGA-II and its derivative Evolutionary Algorithms (EA) provide better performance compared to previous EA: such as (HLGA, NPGA and VEGA). The NSGA-II algorithm, proposed in [8], is the most prominent evolutionary multi-objective heuristic today. For example, Deb et al. [8] unveiled that, on eight of the nine benchmark problems, NSGA-II outperforms HLGA, NPGA, and VEGA in 75% of the runs. Moreover, HLGA, NPGA, and VEGA cover less than 10% of the NSGA outcomes in 75% of all runs and less than 25% in 99% of the runs.

What differentiated NSGA-II from previous genetic algorithms is an intuitive invention of fast elitist ranking procedure. Using this fast ranking procedure, NSGA-II

always preserves the best (higher rank in nondominated rank and larger crowding distance) solution inside the latest population. We will apply the same non-dominated sorting and elitist ranking in our experiment and proposed algorithm.

2.3 Summary

The research related to fog planning is still insufficient. Although previous researchers have proposed resource allocation schemes and facility planning algorithms in geo-distributed cloud (distributed-cloud and multi-cloud), a complete model, that considers placement and provisioning of fog nodes and tactical customers allocation in fog networks, is yet to be proposed. Moreover, a robust solving procedure considering budget and service level is a necessity to this problem. Aiming at these requirements, in this thesis, we propose a multi-objective method to solve the fog planning problem. This model optimizes the capital expenditure and the network delay as a bi-objective problem and provides a complete Pareto front of the cost and the network performance.

Chapter 3

Formulation of the Fog Planning Problem

In this chapter, a mathematical model is formulated for the fog planning problem with a comprehensive explanation. The rest of this chapter is organized as follows. In Section 3.1, we first present the model formulation including the basic concepts, assumptions, objective functions and constraints. Then, Section 3.2 gives an analysis of the NP-hardness of the fog planning problem. In addition, Section 3.3 describes two tuneable parameters in our formulation: network usage function and renting cost. Finally, Section 3.4 summarizes the chapter.

3.1 Fog Planning Problem Description and Modelling

Essentially, the fog planning problem can be decomposed into three subproblems: the fog placement subproblem, fog dimensioning subproblem and demand routing subproblem. In order to find a solution for the whole fog network, these three subproblems can be solved sequentially. Unfortunately, such an approach does not consider the interconnections between the subproblems and the solutions may suffer from local optimum. A different way of solving the fog planning problem is to use a global approach, where the three subproblems are solved simultaneously. Since all the combinations between the subproblems are taken into consideration, the global approach can be expected to reach the global optimum. In this thesis, we use a global approach to model the fog planning problem.

3.1.1 Basic Concepts

Resource demand: In this thesis, two types of resources are considered at the fog nodes: vCPU cores and memory. However, other resources can be included such as the storage and GPU units. This can be achieved by increasing the dimension of the fog profile. For any single period of time, each user-cluster generates a request consisting of vCPU, memory, and bandwidth demands. If a request can be routed to a fog node in such a way that the required vCPU, memory and link bandwidth can be satisfied, then the request is served; otherwise, the request is dropped and sent to the cloud.

Fog type: The fog type we talk about here is an abstraction of the real-world machine server. Different fog types are associated with different computation resources. In real-world planning, the number of fog types and fog profiles can be changed to adequate types or amounts.

Link type: Similarly, different link types between the fog nodes and the cloud are considered. Each link is associated with a bandwidth capacity. These links carry the traffic flow between the fog nodes and the cloud for back-end services such as data synchronizations and application management.

Edge-cluster: Edge-cluster is the notion we used to represent an agglomeration of user requests. Typically, several users are using the cloud at the same time from a common geographical region sharing a unique IP prefix. Instead of modelling them individually, we aggregate them together as a "edge-cluster". However, in optimization research, the request unit is named as "user". In this thesis, the terms "user" and "edge-cluster" have the same meaning.

3.1.2 Assumption

This section describes the assumptions used in this thesis. To formulate the fog planning model, we assume the following information is known:

- The locations of all the edge devices and the possible locations of all the fog nodes (i.e., x and y coordinates). For each edge device, the generated traffic is known.
- The characteristics, i.e., memory, virtual Central Processing Unit (vCPU) of different types of fogs that may be installed in the network.

- The bandwidth availability and the cost of each link type. The link is installed between the fog nodes and the cloud.
- The cloud has unlimited memory and vCPU. We also assume that the cloud is located in a remote location and if a user cannot be served by the fog, it will be routed to the cloud.
- The fog nodes send a fix ratio τ of their total traffic to the cloud. The τ here is a tunable parameter. Different applications will have a different value of τ which captures the amount of traffic that needs to be sent from the fog nodes to the cloud. The possible components of this traffic include database synchronization, data uploading, service management, etc.

3.1.3 Notation

The following notation is defined based upon the information mentioned above.

1. Sets
   - $I = \{1, ..., i, ..., m\}$ set of potential location sites. The location m represents the remote cloud-centre.
   - J, set of edge device clusters that must be served by the fog nodes or the cloud. Each edge-cluster has an aggregated memory, vCPU and traffic demands.
     - η_j, the total number of vCPU required by an edge-cluster $j \in J$.
     - ζ_j, the total amount of memory required by an edge-cluster $j \in J$.
     - T_j, the total traffic generated by an edge-cluster $j \in J$.
     - κ_j, the link speed of an edge-cluster $j \in J$.
   - K, set of fog types (or capacity level) that can be installed at different locations. Different fog types $k \in K$ have different amount of vCPU, memory.
     - α^k the total number of vCPU available for the fog of type $k \in K$.
     - λ^k, the total amount of memory available for the fog of type $k \in K$.
     - c_k^{Fog}, the cost for a fog of type $k \in K$.

- L, set of link types that can be installed at different locations to maintain connections to cloud datacenters. Different link types $l \in L$ have different bandwidth capacities.
  - β^l, the egress bandwidth upper limit for link type $l \in L$.
  - c_l^{Link}, the cost (\$/meter) for a link of type $l \in L$.
- c_i^{Rent}, the renting cost for each potential location $i \in I$.

2. Functions

- $d_{ab} = Distance(a, b)$. The euclidean distance between points a and b. The values of points a and b are the x, y coordinates.
- γ, processing delay. Each router or switch in the data path adds a finite amount of delay as the packet is received, processed, and then forwarded. This includes the time taken at each layer of the Transmission Control Protocol/Internet Protocol (TCP/IP) down to the bit level layer. The processing delay depends on the hop count between user's connection to fog or cloud. The processing delay is calculated as:

$$\text{Processing Delay}(\gamma) = r \cdot h \tag{3.1}$$

  where r is the mean processing delay for each hop (switch or router) and h represents the hop count.

- ψ, transmission delay. The time taken for a process to send the information to the transmission medium (finer or wire). The transmission delay depends on the link speed that is used and the packet size that is to be sent. The transmission delay is calculated as:

$$\text{Transmission Delay}(\psi) = \sigma/\kappa \tag{3.2}$$

  where σ is the packet size (bytes) and κ represents the link speed (bytes/sec).

- μ, propagation delay. It equals to the time taken to transmit a signal from the source to the destination. The propagation delay depends on the medium used. For copper wires, the speed can be approximated to 0.59

speed of light. In this thesis, we use 0.59 speed of light for the speed of copper wire. The propagation delay is calculated as:

$$\text{Propagation Delay}(\mu) = d_{ab}/(0.59 \cdot \textit{Light Speed}). \tag{3.3}$$

where d_{ab} is the euclidean distance between Edge-cluster a and Fog b (km).

- $D(d_{ab})$. The function $D(x)$ is the network usage descriptor. The network usage function $D(x)$ is an abstraction of the network usage between each user and fog facilities, which could represent the network latency experienced by users or the traffic sending to the cloud. This function is transparent to the optimization algorithm. In real life planning, latency is arguably the most important performance metric. A small increase in the latency can cause substantial service level degradation [72, 73]. Therefore, in our experiment, the network usage function $D(x)$ is modelled as the point to point delay.

3. Decision Variables

   - x_{ij}, a 0-1 variable such that $x_{ij} = 1$ if and only if the edge device cluster $j \in J$ is connected to location $i \in I$;
   - y_{ik}, a 0-1 variable such that $y_{ik} = 1$ if and only if the fog type $k \in K_i$ is installed at location $i \in I$;
   - z_{il}, a 0-1 variable such that $z_l = 1$ if and only if the link type $l \in L$ is installed at location $i \in I$.

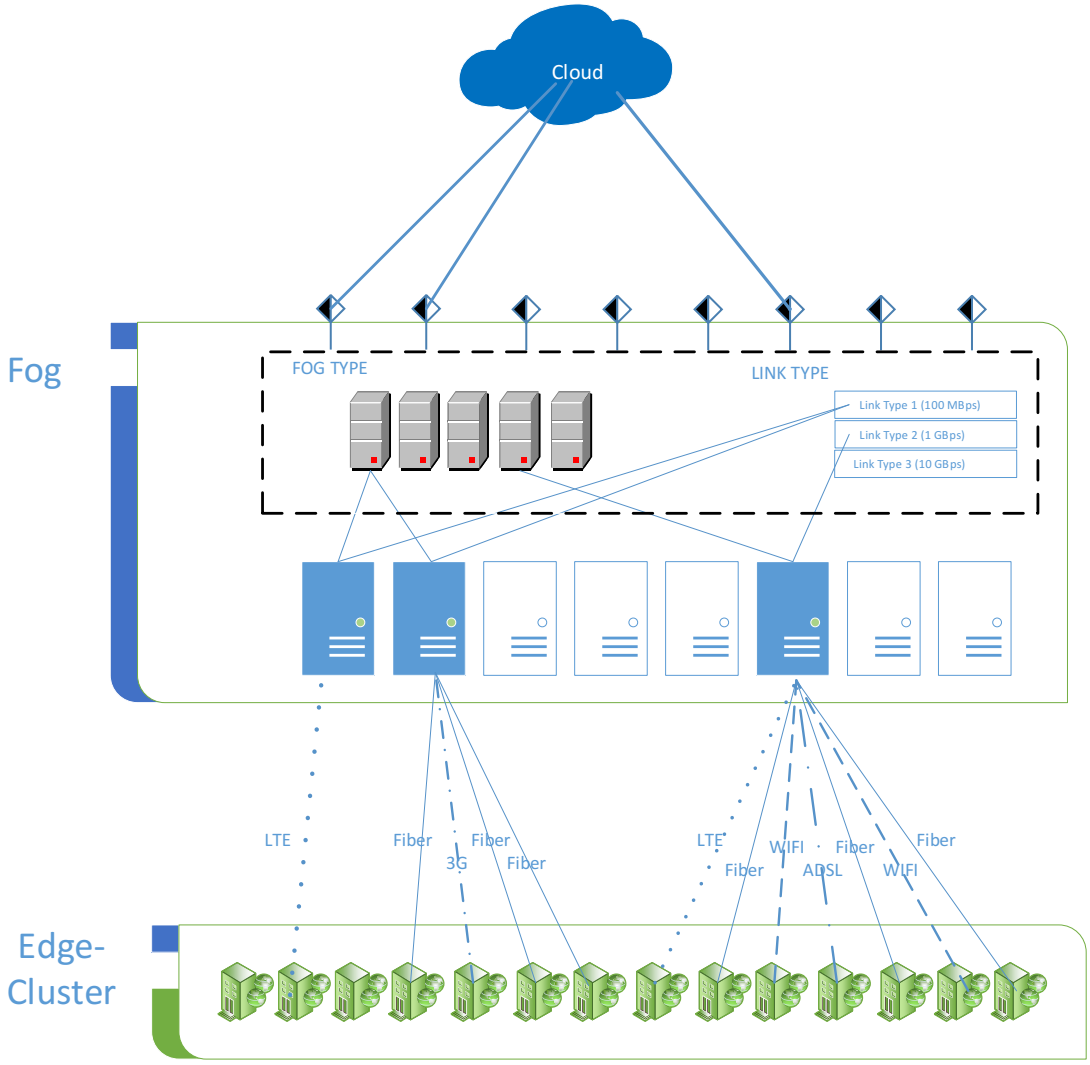

Figure 3.1: Fog placement and fog dimensioning

3.1.4 Mathematical Model

Based on the notation presented in the previous section, we can now formulate the Fog Planning Problem, denoted FPP, as follows:

Minimize Cost

$$Minimize\Big[\sum_{i=1}^{m}\sum_{k\in K} y_{ik}c_i^{Rent} + \sum_{i=1}^{m}\sum_{k\in K} y_{ik}c_k^{Fog} + \sum_{i=1}^{m}\sum_{l\in L} z_{il}c_l^{Link}d_{i,cloud}\Big] \tag{3.4}$$

Minimize Delay

$$Minimize\Big[\sum_{i=1}^{m}\sum_{j\in J} D(d_{ij})x_{ij}\Big] \tag{3.5}$$

$$D(d_{ab}) = \begin{cases} \psi^{ToFog} + \mu^{ToFog} + \gamma^{ToFog} & \sum_{i\in I\setminus m} x_{ij} = 1 \quad \textbf{(connect to the fog)} \\ \psi^{ToCloud} + \mu^{ToCloud} + \gamma^{ToCloud} & x_{mj} = 1 \quad \textbf{(connect to the cloud)} \end{cases}$$

As shown above, to achieve the maximum network performance and cost efficiency, the model simultaneously minimizes the total network delay and the total capital expenditure required to deploy the fog network.

Both Equation (3.4) and Equation (3.5) are subject to following constraints:

$$\sum_{i=1}^{m} x_{ij} = 1 \quad (\forall j \in J) \tag{3.6}$$

Constraints (3.6) are the single source constraints. They ensure that each user connects to exactly one fog or cloud.

$$\sum_{k\in K} y_{ik} \leq 1 \quad (\forall i \in I) \tag{3.7}$$

Constraints (3.7) are the uniqueness constraints. They enforce that at most one fog node is installed at a given location. In practice, we can install multiple servers in each location, and each server can have different hardware configurations (memory sticks, CPU, hard disk drive, GPU, etc.) To reduce the complexity, we generalize different server types and hardware combinations to a fix number of fog types. Under this assumption, each potential location can select an appropriate fog type to accommodate the workload demands. In other words, we cannot install two or more fog nodes at the same location. If the left side of Equation (3.7) equals to zero, it means that the corresponding location is not selected; no fog facility will be installed at this

location.

$$\sum_{l \in L} z_{il} \leq 1 \quad (\forall i \in I) \tag{3.8}$$

Similar to Constraints (3.7), Constraints (3.8) are the link uniqueness constraints. They enforce that at most one link type can be installed at each location. If Equation (3.7) equals to zero, the corresponding location is not open and therefore, no link will be installed at this location.

$$x_{ij} - \sum_{k \in K} y_{ik} \leq 0 \quad (\forall i \in I, \forall j \in J) \tag{3.9}$$

Constraints (3.9) : Openness constraints. These constraints ensure that users can only connect to a fog that is opened.

$$\sum_{l \in L} z_{il} \leq \sum_{k \in K} y_{ik} \quad (\forall i \in I) \tag{3.10}$$

Constraints (3.10) make sure that each installed fog node at location i will be connected to the cloud.

$$\sum_{j \in J} \eta_j x_{ij} \leq \sum_{k \in K} y_{ik} \alpha^k \quad (\forall i \in I) \tag{3.11}$$

$$\sum_{j \in J} \zeta_j x_{ij} \leq \sum_{k \in K} y_{ik} \lambda^k \quad (\forall i \in I) \tag{3.12}$$

Constraints (3.11) and (3.12) are the capacity constraints. They are the capacity constraints for vCPU, memory at the node level. They ensure that the total resource demand does not exceed each fog node's hardware capacity.

$$\sum_{j \in J} x_{ij} T_j \cdot \tau \leq \sum_{l \in L} z_{il} \beta^l \quad (\forall i \in I) \tag{3.13}$$

Constraints (3.13) are the link capacity constraints. They state that the total bandwidth from fog site to cloud cannot exceed the egress link bandwidth upper bound.

$$x_{ij} \in \{0,1\} \quad (\forall i \in I, \forall j \in J) \tag{3.14}$$

$$y_{ik} \in \{0,1\} \quad (\forall i \in I, \forall k \in K) \tag{3.15}$$

$$z_{ik} \in \{0,1\} \quad (\forall i \in I, \forall k \in K) \tag{3.16}$$

Finally, Constraints (3.14), (3.15) and (3.16) define the decision variables as binary.

3.2 NP-Hardness

This section establishes the NP-hardness of the FPP. The FPP has two objective functions that need to be minimized: building cost and network delay. The decision variables consist of the placement and assignment decisions.

Essentially, this problem is a multi-objective combinatorial problem. If we relax certain conditions, there are related single-objective optimization problems that we can gain insight from. Suppose that we know that certain number of fog facilities will be opened and that we relax the capacity constraint at each location. Then the problem of assigning edge-clusters to open facilities while minimizing the total delay can be reduced to the K-median clustering problem.

K-median clustering problem: Suppose there exists a bipartite graph with a bi-partition (F, C), where F is a set of facilities and C is a set of clients, and let k be a positive integer specifying the number of facilities allowed to be opened. Let c_{ij} be the cost of connecting client j to facility i. The objectives are to find a subset $I \subseteq F, |I| \leq k$ of facilities that should be opened and a function $\phi : C \to I$ assigning clients to open facilities that minimize the total connecting costs [74].

The NP-hardness of the K-median clustering problem was proved in 1984 by Megiddo and Supowit [75]. However, even if we can solve this problem using an approximation algorithm, we still have to decide a reasonable k regarding to the total budget.

From another perspective, suppose we combine the building cost and network

delay into a single objective function. The objective function becomes:

$$Minimize: \ \alpha \cdot \sum^{opened\ fog} fog\ building\ cost + \beta \cdot \sum^{all\ users} network\ delay \tag{3.17}$$

where α and β are the normalizing constants. Problems of this form are referred to as the capacitated facility location problem.

Capacitated Facility Location Problem (CFLP): Suppose there exists a bipartite graph with a bi-partition (F, C), where F is a set of facilities and C is a set of clients. A fixed cost $f_i \geq 0$ for opening each facility $i \in F$; a capacity $u_i \geq 0$ for each facility $i \in F$; a demand $d_j \geq 0$ for each client $j \in C$. The problem is to find a subset $I \subseteq F$, of facilities to opened as well as a assignment function $\phi : C \to I$ that assigns clients to open facilities. The objective is to minimize the total connecting cost without violating each facility's capacity constraint: $\sum_{j \in C} x_{ij} d_j \leq u_i, \forall i \in F$ [74].

Megiddo and Supowit [75] proved that exact solution of CFLP is NP-hard. Fowler el al. [76] proved that when the error is small, even a approximation to this problem is NP-hard.

Specifically, our problem needs to add a single source constraint. The reason is each users can only go to one single fog; in this regard, the problem becomes the Single-Source Capacitated Facility Location Problem (SSCFLP). In SSCFLP, deciding whether a feasible solution exists at all is NP-complete [77]. Moreover, our problem has a modular facility cost model, which provides several capacity levels of the fog facility. This transfers the model to a single-source modular capacitated facility location problem, which is a more complicated version of CFLP [77].

As discussed above, the single objective model of this problem is extremely complicated and computational complex. Therefore, in the following chapter, we will introduce methods to solve this planning problem from the multi-objective perspective.

Noticeably, if we take $\alpha = 1$ and $\beta = \lambda - 1$ and divide the above equation with λ, the equation turns to:

$$\textbf{Minimize}: \ \frac{1}{\lambda} \cdot \sum^{opened\ fog} fog\ building\ cost + (1 - \frac{1}{\lambda}) \sum^{all\ users} network\ delay \tag{3.18}$$

This method coincides with the Weighted Scalarization method, which is a general form of the weighted sum method. Similar to the weighted sum method, weighted

scalarization has the same drawbacks of the discrete incompleteness, which has been discussed in Chapter 2.

3.3 Extended Discussion of the FPP Model

Network Usage Descriptor $D(x)$: As mentioned above, the network usage function $D(x)$ is an abstraction of the network usage between each user and fog facilities. Therefore, it is transparent to the mathematical model and solving procedures described above. The $D(x)$ can be the end to end delay, bandwidth usage or hop count (see Table 3.1). More importantly, it means we can manipulate this $D(x)$ matrix beforehand to fit specific planning requirements. For instance, some users may not want their data to be uploaded to a specific fog entity due to security concerns or policy regulation issues. We can label the value $D(x)$ of this user-fog link to infinite; therefore, the traffic of these users will never be sent to the unfavourable fog node.

Table 3.1: Network usage descriptor $D(x)$

#USER/#FOG	FOG1	FOG2	FOG3	CLOUD
D(x) equals to point to point delay (Microsecond)				
User-1	32.66	95.75	72.12	62.28
User-2	77.89	30.4	66.16	51
User-3	63.12	93.16	49.87	93.01
User-4	95.95	56.19	54.23	31.89
User-5	62.97	59.38	46.35	79.03
...				
D(x) equals to traffic (Megabyte)				
User-1	1.0889	2.2891	9.4236	1.6984
User-2	9.9679	7.0204	9.9916	7.0415
User-3	6.5581	6.8513	1.9966	3.3555
User-4	4.4329	4.7594	3.8985	5.5533
User-5	4.9868	5.9794	4.8781	4.6624
...				
D(x) equals to hop count				
User-1	5	11	15	14
User-2	13	8	11	3
User-3	7	11	8	4
User-4	10	5	13	6
User-5	11	6	8	9
...				

Renting Cost: The strategy of using a massive number of fogs in various locations may be preferred for a better network performance. Nonetheless, building a larger number of fog sites within a local area incurs higher operational expenses. Also, the energy efficiency would suffer from a large number of operating servers. Therefore, the rent parameter gives us the power to adjust the number of fog to be built. A large renting cost will force the model to operate a smaller number of fogs and produce an efficient fog placement plan with low maintenance costs.

3.4 Summary

In this chapter, we formulated the mathematical model for the fog planning problem. The fog planning problem is an optimization problem with two objectives: 1) minimize the capital expenditure and 2) minimize the delay experienced by the users. To better understand the problem, the NP-hardness of the FPP is analyzed in Section 3.2. In Section 3.3, we further explained two parameters in our model: network usage and renting cost. Varying the values of these two parameters could result in a flexible planning model.

Chapter 4

Algorithms for Solving the Fog Planning Problem

This chapter presents the exact and approximation algorithms for the fog network planning. The rest of this chapter is organized as follows.

First, an exact algorithm (the weighted sum method) is presented in Section 4.1, including modelling procedure and solving processes by using Cplex. The results obtained from the weighted sum method are used as a benchmark for the delay gap comparison. The details of the delay gap comparison will be presented in Chapter 5. In Section 4.2, we present the methodology and procedure of two existing Evolutionary Multi-objective Optimization (EMO) algorithms (NSGA-II and SMPSO). These two algorithms are based on the well known Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) respectively. The optimization process and the pseudocode of both algorithms are provided. Then, motivated by various drawbacks of existing heuristic algorithms, a novel evolutionary algorithm is proposed in Section 4.3. Finally, Section 4.4 summarizes the chapter.

4.1 Exact Algorithm for the FPP

To solve the multi-objective problem, the basic ideology is to combine multiple objectives into a single objective. The most popular approach applied to this ideology is the weighted sum method.

4.1.1 Weighted Sum Method

Weighted sum method is a popular and widely used approach for multi-objective optimization problems [78]. The weighted sum approach combines multiple objective functions into a single objective function. In this combination, different objectives are given a certain weight between 0 and 1. The multi-objective problem can be combined by using the Formula (4.1).

$$min\ or\ max\left\{\sum_{j=1}^{Q}\lambda_j z^j(x) : x \in X\right\} \tag{4.1}$$

$$0 \leq \lambda_j \leq 1 \tag{4.2}$$

$$\sum_{j=1}^{Q}\lambda_j = 1 \tag{4.3}$$

where λ_j is the weight parameter assigned to each objective to capture the relative importance on a given objective. Assigning a higher (lower) weight places more (less) emphasis on this objective. Formula (4.1) is used to minimize or maximize the parametric summation of all the objective functions. By varying the weight vector, all Supported Efficient (SE) solutions can be found [21]. The advantage of the weighted sum method is that for each $\lambda \in \mathbb{R}^Q$, the problem is as difficult as the single objective problem [21]. The optimal solution can be obtained by using the Linear Programming (LP) solver if the time complexity is not a constraint.

4.1.2 Solving the Weighted Sum Model with Cplex

After combining multiple objective functions into one single objective function, the next step is to solve this single objective function. In this work, a commercial LP solver called Cplex, is employed to obtain optimal solutions and solve the problems.

4.2 Approximation Algorithms for the FPP

In this thesis, the Evolutionary Multi-objective Optimization (EMO) is applied to solve the fog planning problem. EMO is based on heuristic multi-objective optimization techniques that imitate the principles of natural selection and survival of the fittest to find near-optimal solutions [23]. In EMO, multiple Pareto-optimal solutions

are found in a single simulation by emphasizing multiple non-dominated and isolated solutions [23]. To better understand EMO, we will introduce several basic concepts of the EMO in next section.

4.2.1 Basic Concepts

Chromosome or particle: In evolutionary algorithms, the solution of a problem is first encoded in the form of strings of binary numbers. Then, the evolutionary operators, such as crossover and mutation, can be applied on the encoded strings to obtain the optimal solution of the problem. The terms "chromosome" and "particle" are used to represent this "string" in GA and PSO respectively. For the fog network planning problem, the string is encoded to reflect the fog placement scheme, the fog dimensioning scheme and the edge-cluster routing policy.

Non-dominated sorting: The non-dominated sorting process sorts a population into a hierarchy of subpopulations based on the ordering of the Pareto dominance. NSGA-II employs this process to keep the high rank of non-dominated solutions inside the latest population.

Position and velocity updating: SMPSO employs the position and velocity updating process to evolve each "particle". In every iteration, each particle updates its position and velocity in the search space by following the global and historical best solutions (fitness).

4.2.2 Evolutionary Multi-Objective Optimization Methodology

The EMO algorithm starts the reproduction process by generating an initial population. Each member in the initial population represents a candidate solution to the goal problem. Each member is then evaluated by the given objective and constraint functions to obtain a fitness value. Then, different EMO algorithms will apply different evolving processes to improve the solution quality (fitness value) of the members in the population. The whole process terminates when the predefined conditions are met. The solutions will be selected as output only if they are non-dominated by any members in the population, and they are not violating any constrains.

As mentioned above, the initial population is a group of candidate solutions. The first step of generating the initial population, is to design a representative scheme to

encode the FPP decision variables. The encoding scheme transfers multiple variables from a goal problem into a string of binary or integer values. The encoding technique has significant influence on EMO's performance. A detailed overview of EMO's encoding techniques is presented in [79]. In order to solve different engineering problems, EMO's encoding must be modified to fit the specific problem.

4.2.3 Proposed FPP Decision Variables Encoding

The method to encode the fog placement, fog dimensioning, and routing tree into chromosomes or particles is critical for developing the solution procedure. The classic random encoding is not appropriate for the fog planning problem due to the following reasons:

- The nature of the single-source combinatorial problem indicates that each edge-cluster can only connect to one single fog, and each fog can only select one fog type. The crossover and mutation process in evolutionary algorithms will break this unmodality, and will return infeasible results which do not correspond to a valid network plan.
- Random encoding leads to a longer computing time due to the fact that a large portion of time is consumed in evolving infeasible solutions.

In this thesis, a modified Count-Preserving Encoding (CPE) is proposed to solve the FPP. The CPE, proposed in [80], solves the problem in random encoding through a bit-count tracking scheme. In our encoding scheme, we employ the same bit-count tracking scheme, and combine a novel facility capacity guarantee. The facility capacity guarantee is enforced by applying the following constrains on each chromosome.

We assume the number of edge-clusters as $|J|$, and the number of potential locations as $|I|$. The dimension of the chromosome (N) equals to $2|I| + |J|$. Let, $C_m = \left[X_{m,1}, X_{m,2}, X_{m,3}, ..., X_{m,N}\right]$ be the m_{th} particle of the population where each component: $X_{m,n}, 1 \leq n \leq |J|$ denotes where edge-cluster n is routed to. $X_{m,n}, |J| + 1 \leq n \leq |I| + |J|$ denotes the fog type decisions for different potential location i. $X_{m,n}, |I| + |J| + 1 \leq n \leq 2|I| + |J|$ denotes the link decisions for location i. Figure 4.1 illustrates a variable string example for a FPP with five edge-clusters and three potential fog sites.

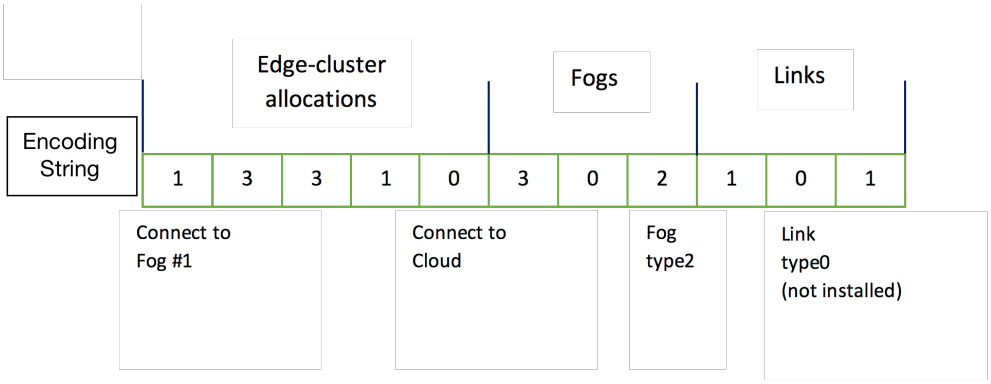

Figure 4.1: An EMO encoding example for FPP

The capacity enforcement for the m_{th} particle is shown below:

$$\text{For each fog type field:} \quad X_{m,n}, \quad (|J|+1 \leq n \leq |I|+|J|) \tag{4.4}$$

$$D^{vCPU}_{X_{m,n}} = Max(LUB(k \in K | \alpha^k > g(i), i \in I), X_{m,n}) \tag{4.5}$$

$$D^{memory}_{X_{m,n}} = Max(LUB(k \in K | \lambda^k > h(i), i \in I), X_{m,n}) \tag{4.6}$$

$$X_{m,n} = Max(D^{vCPU}_{X_{m,n}}, D^{memory}_{X_n}) \tag{4.7}$$

$$\text{For each link type field:} \quad X_{m,n}, \quad (|I|+|J|+1 \leq n \leq 2|I|+|J|) \tag{4.8}$$

$$X_{m,n} = Max(LUB(l \in L | \beta^l > \tau \cdot o(i), i \in I), X_{m,n}) \tag{4.9}$$

Function LUB$(\cdot)$ calculates the Least Upper Bound (LUB) for the fog type or the link type. In other words, for each location, it outputs the facility type with computation resources no less than the total demands routing to this location. Function *g(i), h(i), o(i)* represents the total vCPU requests, memory requests, and traffic requests routing to the location $i, (i \in I)$. Equation (4.7) enforces that each fog site's capacity is no less than either the total vCPU or the total memory demands. Equation (4.9) ensures that each fog site's link bandwidth is no less than the traffic demands between this fog node and the cloud.

4.2.4 Description of NSGA-II

NSGA-II follows the same steps as the classical GA. The classical GA first initializes a population of N individuals, then it generates offsprings by applying the

crossover and mutation operations, and finally evaluates and selects the fittest solutions as output solutions. What differentiates NSGA-II from previous algorithms (such as VEGA and HLGA) is an intuitive invention of the fast elitist ranking procedure (Non-dominated Sorting). Using this fast ranking procedure, NSGA-II always preserves the best (higher rank in non-dominated rank and larger crowding distance) solutions inside the latest population. The pseudocode of the NSGA-II algorithm is presented in Algorithm 2.

Algorithm 2 NSGA-II Algorithm

```
Initialize a population of N individuals as the "parent population" P
while Iteration < MaxIteration do
    C ← Empty child population
    while the number of individuals C in < N do
        Select parent1 (by tournament selection)
        Select parent2 (by tournament selection)
        Get child1, child2 through the Binary Crossover (parent1, parent2)
        Polynomial Mutation (child1, child2)
        Evaluate child1, child2 for their fitness values
        Insert child1, child2 into C
    end while
    U ← Combine P and C to get 2N individuals
    Rank the union set U using the nondominated sorting.
    P ← N front individuals in U by the crowded comparison selector.
end while
Return the set of feasible non-dominated solutions in the latest population.
```

4.2.5 Description of the SMPSO

Particle Swarm Optimization (proposed by J. Kennedy and R. Eberhart in [24]) models the social behaviour of biological creatures through the mathematical approach. The pseudocode of the SMPSO algorithm is presented in Algorithm 3. Similar to the classical PSO algorithm, SMPSO first randomly generates a set of N initial solutions, then iteratively updates the "solution positions" in the searching space [9]. In each iteration, every particle adjusts its velocity to follow the local and global best solutions. We assume that each particle i is randomly assigned a position

$\vec{x}_i$ $(i = 1, 2, ..., N)$ and a velocity $\vec{v}_i$ $(i = 1, 2, ..., N)$. The algorithm updates the particle's position by:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{4.10}$$

The velocity function $\vec{v}_i(t)$ is given by:

$$\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + C_1 \cdot r_1 \cdot (\vec{x}_{p_i} - \vec{x}_i) + C_2 \cdot r_2 \cdot (\vec{x}_{g_i} - \vec{x}_i) \tag{4.11}$$

where $\vec{x}_i(t)$ and $\vec{v}_i(t)$ are the location and velocity of particle i at time t. The $\vec{x}_{p_i}$ and $\vec{x}_{g_i}$ are respectively, the historical best and the global best points. w is the inertia weight of the particle, which controls the trade-off between the global and local experience. C_1 and C_2 are learning factors which control the effect of the local and global best particle. r_1 and r_2 add randomness between the global and local searching direction.

Algorithm 3 SMPSO Algorithm

1: Initialize a population of N individuals as "swarm population" S
2: Evaluate the solutions in the "swarm population"
3: Put non-dominated solutions into an empty "elite archive" A
4: **while** `Iteration` $<$ `MaxIteration` **do**
5: **for each** $s \in S$ **do**
6: Use constrained binary tournament to select a solution from the elite archive
7: Use the solution from last step as the global best particle
8: Compute the speed of s according to the speed formula (4.11)
9: Update the position of s according to the speed calculated in the last step
10: **end for**
11: Apply the polynomial mutation to $\tau\%$ of the population
12: Evaluate the solutions in the swarm population
13: Insert the non-dominated solutions from S to A.
14: **end while**
15: Return the set of feasible non-dominated solutions in the elite archive (A).

A major difference between SMPSO and previous PSO-based algorithms is that SMPSO adopts a constriction coefficient as shown in Equation (4.12) on the resulting velocity [81].

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{4.12}$$

where

$$\varphi = \begin{cases} C_1 + C_2 & \text{if } C_1 + C_2 > 4 \\ 1 & \text{if } C_1 + C_2 \leq 4 \end{cases} \tag{4.13}$$

Another difference, in SMPSO, is the polynomial mutation is applied to the τ percent of the particles. In our experiment, we use the same value for τ (set to 15%), as in the original implementation of SMPSO [9].

4.3 Proposed Approximation Algorithm

Previous studies in NSGA-II and SMPSO mainly focus on applying these algorithms on continuous problems. However, for discrete problems, such as the combinatorial problem we are dealing with, the preliminary results of our experiment reveal that these two heuristic algorithms hold different characteristics.

As shown in Figure 4.2, the NSGA-II performs efficiently regarding the convergence to the Pareto front, however, its solution points are unevenly distributed in the search space. Even if we change the mutation index or the crossover index settings, no obvious improvement can be perceived. The researchers in [82], have observed the same disadvantage in continuous NSGA-II applications. This disadvantage stems from the NSGA-II's sorting process. The concentrated effect in the non-dominated sorting harms the diversity in NSGA-II solutions. The researchers in [82] proposed an elitism strategy to overcome this shortcoming. However, the time-complexity involved in evaluating the elitism sets is high, and can be a huge issue for large scale problems.

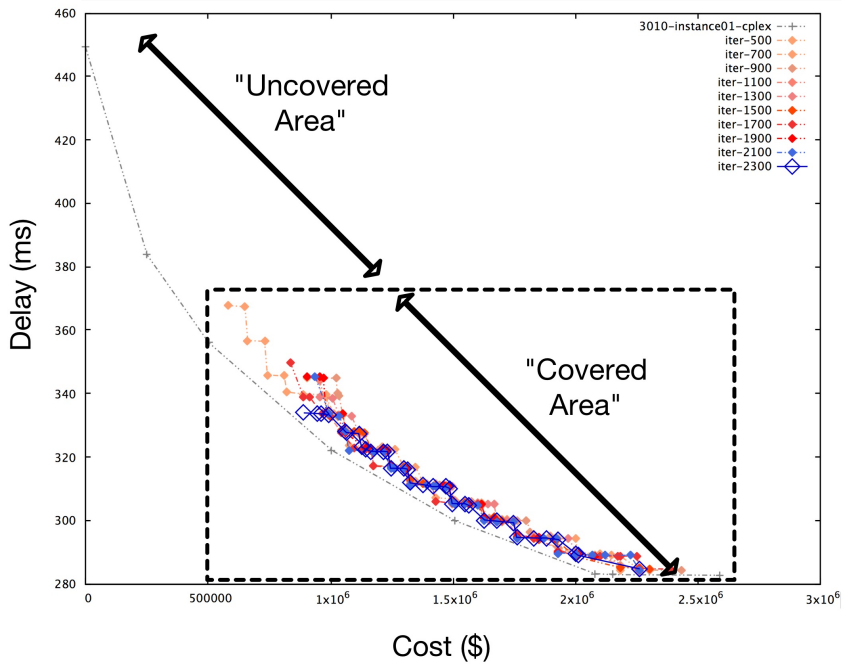

Figure 4.2: NSGA-II algorithm struggles to cover the whole Pareto front

On the other hand, SMPSO shows different characteristics. The preliminary experiment demonstrates that SMPSO performs efficiently on exploring the whole front. It preserves the diversity in the solutions set and always produces evenly distributed Pareto frontiers. However, after a given number of iterations, SMPSO struggles to push the solution set to converge to the true optimal front. The preliminary results are shown in Figure 4.3.

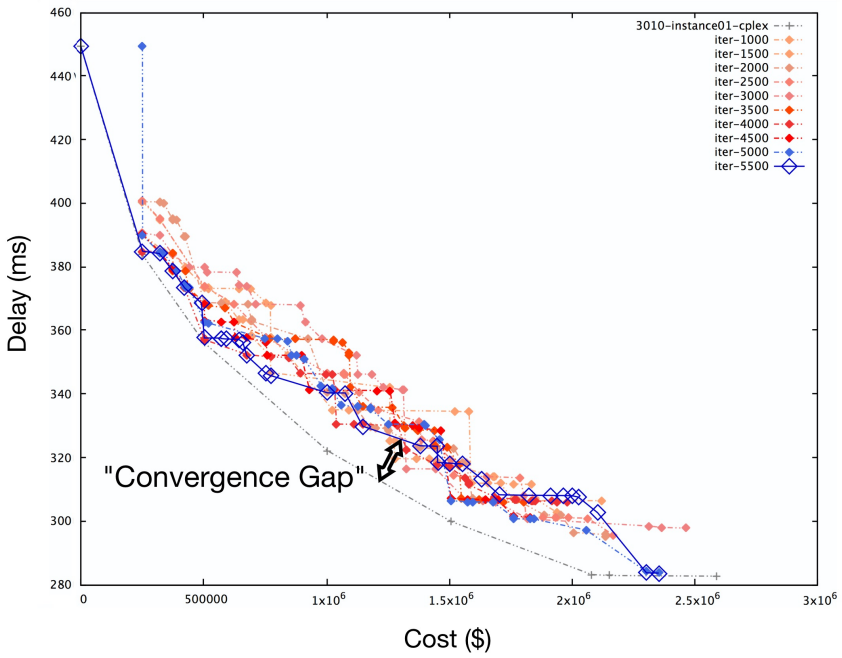

Figure 4.3: SMPSO algorithm struggles to converge to true pareto front

Motivated by the disadvantages in NSGA-II and SMPSO algorithms, we propose a new version of NSGA-II. The algorithm developed is named Particle Swarm Optimized Non-dominated Sorting Genetic Algorithm (PSONSGA). The PSONSGA is designed specifically to exploit the searching efficiency of PSO-based algorithms.

4.3.1 Description of the PSONSGA

Particle Swarm Optimized Non-dominated Sorting Genetic Algorithm (PSONSGA) is a variation of NSGA-II based on the idea of employing particle swarm optimization before proceeding to the NSGA-II's procedure. Following the similar two-phase methodologies in [83–85], PSONSGA consists two optimizing phases: the PSO phase and the NSGA-II phase. The purpose of the PSO phase is to explore the decision space and preserve the diversity in the solution population. In the NSGA-II phase, the non-dominated sorting emphasizes the effort to converge the solutions towards the optimal front. Furthermore, in the NSGA-II phase, an enforced selection process is introduced to increase the fitness pressure on the population. The

pseudocode of PSONSGA is presented in Algorithm 4. PSONSGA uses the same behaviour as in the SMPSO for the first 50% of the run. In the last 50% of the run, a variant of NSGA-II with an aggressive selection is executed. The aggressive selection enforces the convergence of solutions and the extension of the solution front. The population is expected to get closer to the optimal solutions.

4.3.2 Procedures of PSONSGA

PSO procedure

In the first 50% of the PSONSGA iterations, the PSO procedure is employed to explore the decision space, and to test different search directions (see rows 3-12). The position and velocity updating mechanism in PSO, speeds up this searching process. The diversity in population is preserved before proceeding to the next phases.

NSGA-II procedure

In the last 50% of the run of PSONSGA, the population is expected to be evenly diversified and relatively close to the Pareto front. The main goal is not to explore the searching space anymore but to increase the closeness to the optimal solutions (see rows 17-27) [86]. The NSGA-II procedure, including non-dominated sorting, is executed to concentrate the solutions towards the true optimal front. Also, we employ an aggressive selection process as proposed in [86] (see row 24).

Algorithm 4 PSONSGA Algorithm

Initialize a population of N individuals as "swarm population" S
Evaluate the solutions in the S
Put non-dominated solutions into an empty "elite archive" A
while `iteration` $\leq$ `50% Maximum iteration` **do**
    PSO procedure:
    for each $s \in S$ **do**
        Use constrained binary tournament to select a solution from elite archive
        Use the solution from last step as the global best particle
        Compute the speed of s according to the speed equation shown in (4.11)
        Update the position of s by the speed calculated in the previous step
    end for
    Apply the polynomial mutation to 15% of the population
    Evaluate the solutions in the swarm population
    Update the elite archive: insert the non-dominated solution from swarm to archive
end while
$M \leftarrow$ Elite solutions in SMPSO's elite archive A
$C \leftarrow M$ (Use the solutions in M as initial population for NSGA-II)
while `Iteration` $\leq$ `MaxIteration` **do**
    NSGA-II procedure:
    $D \leftarrow$ Empty child population
    Use constrained binary tournament to select parents in C.
    while `not enough individuals in D` **do**
        Select parent1 (by tournament selection)
        Select parent2 (by tournament selection)
        Getting child1, child2 through Binary Crossover (parent1, parent2)
        Polynomial Mutation (child1, child2)
        Evaluate child1 and child2 for their fitness values
        PSONSGA's Aggressive Selection Process
        Insert the child(ren) into D
    end while
    Execute the non-dominated-sorting over "preprocessed population" C and offsprings population D.
    Select individuals for the next generation.
end while
Return the set of feasible non-dominated solutions in population C

Aggressive Selection Process:

An offspring is included in the population only if it is non-dominated or it improves one objective's minimum/maximum value so far. The aggressive selection process enforces the fitness pressure upon the population. Since the diversity is preserved in the PSO-procedure, a harsh fitness process will not hurt the solution's spread character [86].

In conclusion, PSONSGA is expected to show an improvement compared to the traditional NSGA-II by applying the PSO-based preprocessing and an aggressive selection process in the second phase. The Pareto front from the PSONSGA could be expected to have the better diversity and convergence.

4.4 Summary

In this chapter, a detailed description of the weighted sum method, NSGA-II algorithm and SMPSO algorithm was provided. Motivated by various drawbacks on existing multi-objective heuristic algorithms, a novel evolutionary algorithm was proposed. In the next chapter, we present the numerical results of applying these four multi-objective algorithms on the fog planning problem.

Chapter 5

Results and Analysis

In this chapter, we solve the fog planning problem with the weighted sum method, NSGA-II, SMPSO, and PSONSGA algorithms. Then, we do a complete analysis of the obtained results in terms of the solution quality and computation time. More precisely, the chapter begins with a summary of the steps taken to carry out the experiments. Then, in Section 5.2, we introduce the experiment setup which consists of input and the selection of the parameters. In Section 5.2.3, we analyze the planning results from an instance of the FPP to better understand the characteristics of the solution sets and delay differences. Then, we evaluate the results from four instances of 26 different problem sizes followed by a comparison in terms of quality indicators, delay gaps and CPU time. Finally, the last section summarizes the chapter.

5.1 Framework for the FPP Experiments

Figure 5.1 summarizes the steps that we used for the fog planning experiments. First, the FPP instances are generated. Each instance includes the edge-cluster demands, the edge-cluster locations, the potential fog locations as well as the fog and link types. Second, the FPP instance is solved using the weighted sum method. The optimal solution points are used as a benchmark in the delay gap comparison. In parallel, the proposed encoding scheme is employed to encode the FPP instance for the EMO. Third, the same FPP instance is solved using three EMOs (NSGA-II, SMPSO and PSONSGA). Fourth, the solutions returned by different EMOs are compared with the solution obtained from the weighted sum method. Fifth, all these solutions are evaluated by two multi-objective quality indicators: HV and IGD.

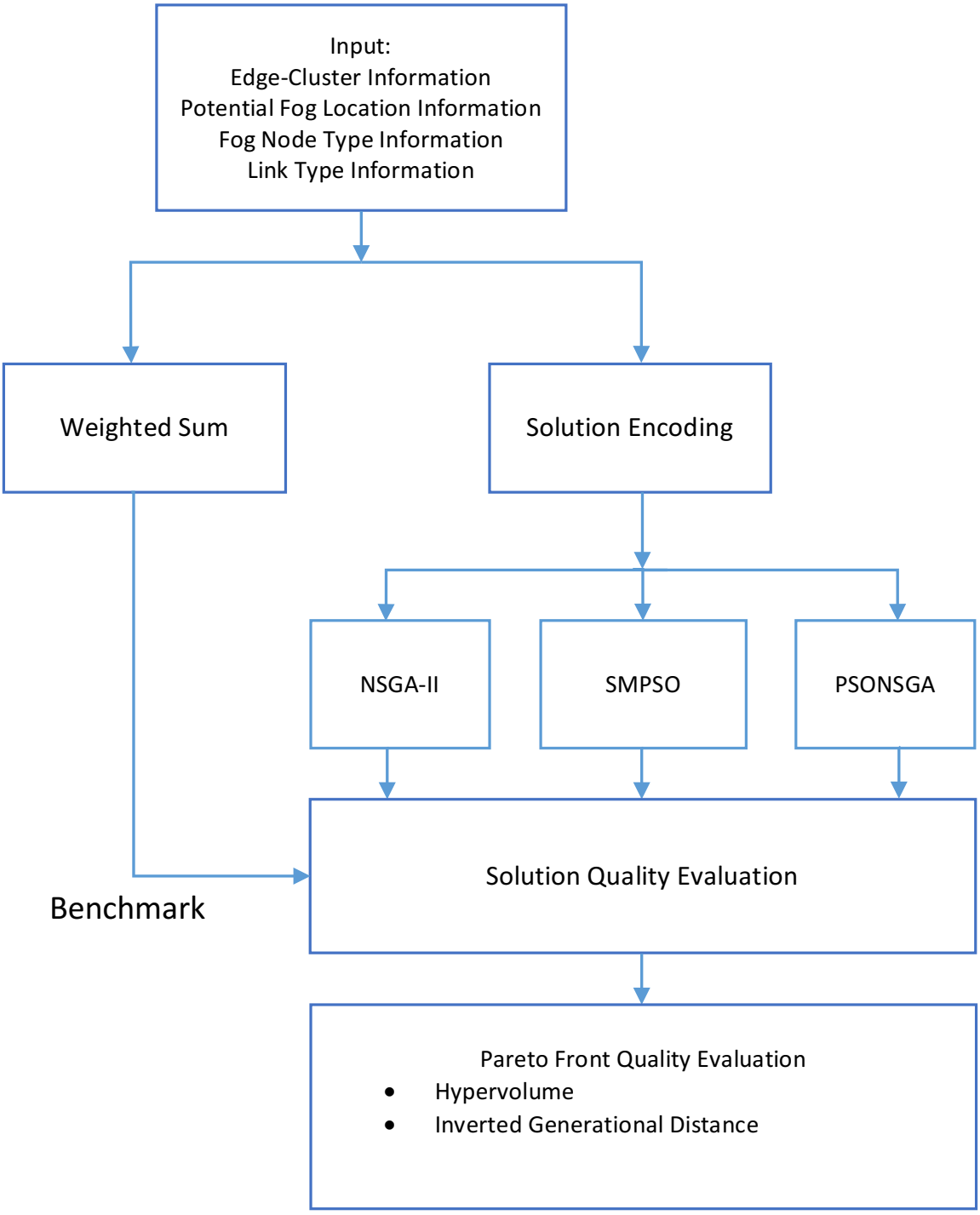

Figure 5.1: Steps for the FPP experiments

5.2 Experiment Setup

5.2.1 Experiment Input

In this thesis, the notion of edge-cluster is used to model the demands. Each edge-cluster represents a group of co-located clients who would be sending requests to the fog or cloud. In our experiments, each edge-cluster has its own number of users, and for each user, the demand will be generated according to the parameters presented in Table 5.1. This includes the vCPU, memory and bandwidth demands. All these demands are generated following the uniform distribution. Also, each edge-cluster has a coordinate (x, y) which is randomly generated in the area. The euclidean distance between the edge-cluster and the fog is used to calculate the propagation delay.

Table 5.1: Edge cluster demands

For each edge-cluster:	
Number of users within the cluster	U(10-150)
Coordinates of the edge-cluster	(x, y) (within 100 x 100 km^2)
For each user inside an edge-cluster:	
Number of vCPU core	U(1-4)
Memory	U(1-40) GB
Number of packets sent per second	U(1-64)
Network access bandwidth	U(20-70) Mbps

We define a problem with the following notation: FPP(number of edge-cluster) (number of candidate locations). For example, “FPP2005” represents a problem where 20 edge-clusters and 5 candidate locations are uniformly distributed in a given area. To build fog networks, we assume, without loss of generality, that four different fog types and three different link types are available. Table 5.2 and Table 5.3 present the different fog and link types respectively.

Table 5.2: Fog type characteristics

Fog Type	**# of CPU**	**Memory (GB)**	**NIC (Mbps)**	**Cost ($)**
1	90	480	360	67200
2	180	800	1024	120000
3	360	1600	1024	170000
4	720	3200	10240	250000

Table 5.3: Link types characteristics

Link Type	**Capacity (Mbps)**	**Link cost ($/meter)**
1	100	0.25
2	1000	2
3	10000	200

Four instances of 26 different problem sizes are generated within a $100km \times 100km$ area. Table 5.4 shows the 26 different problem sizes. The first column in the table represents the problem number. Column 2 shows the problem name using the notation described previously. Finally, the last two columns present the number of edge-clusters that need to be served and the number of potential fog locations respectively.

Table 5.4: Problem sizes

Problem Index	**FPP Names**	**Number of Edge-Clusters**	**Number of Potential Locations**
1	FPP0505	5	5
2	FPP1005	10	5
3	FPP1505	15	5
4	FPP2005	20	5
5	FPP2505	25	5
6	FPP3005	30	5
7	FPP3505	35	5
8	FPP4005	40	5
9	FPP4505	45	5
10	FPP5005	50	5
11	FPP5505	55	5
12	FPP6005	60	5
13	FPP3010	30	10
14	FPP3510	35	10
15	FPP4010	40	10
16	FPP4510	45	10
17	FPP5010	50	10
18	FPP5510	55	10
19	FPP6010	60	10
20	FPP6510	65	10
21	FPP7010	70	10
22	FPP7510	75	10
23	FPP8010	80	10
24	FPP8510	85	10
25	FPP9010	90	10
26	FPP9510	95	10

5.2.2 Experiment Environment

All the experiments were run on a HP workstation with a Quad core processor, 2.66GHz internal clock and 4GB of memory.

The problem inputs are generated by a program written in JAVA. The source code of NSGA-II and SMPSO were taken from the JMETAL framework [87]. PSONSGA was also implemented in JAVA.

5.2.3 Settings for the Weighted Sum Method

The commercial solver CPLEX 12.07 with default settings is used for the weighted sum method. We use 11 pairs of weights (with steps of 0.1) to solve each problem instance. For each weight combination, the time limit for CPLEX is set to one hour. This means that if CPLEX cannot find the optimal solution for each single weight setting within one hour, it will return the best solution found so far. Also, due to the NP-hardness of FPP, even the computer's memory may be insufficient. In this case, CPLEX will return the best solution found before it runs out of memory.

5.2.4 Settings for Evolutionary Algorithms

As mentioned previously, each heuristic algorithm has several tunable parameters, which can eventually have an impact on the quality of the final solution. Before running experiments on the generated data sets, NSGA-II and SMPSO's parameters were tuned so that best results can be obtained.

The first parameter that needs to be decided is the population size. Reeves [88] studied on the minimum population size for an efficient GA search. Their research showed that for a q-ary alphabets (q represents the possible values for a string position) encoding scheme, the minimum population size can be numerically calculated for specified confidence level. Figure 5.2 illustrates the string length and population size relationship for a 99.9% confidence level. According to Figure 5.2 and the size of our FPP problems (15-115 string length, 4-8.87 possible values), we can conclude that a population size of 100 is an efficient population size. The search space can be sufficiently covered within a reasonable computation time.

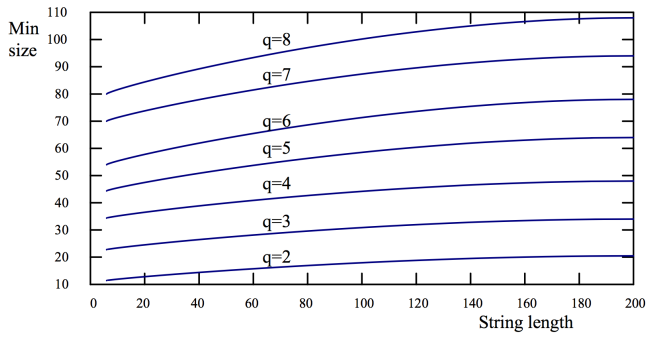

Figure 5.2: Efficient population size vs string length [88]

For the NSGA-II algorithm, we compare the volume of the dominated space (HV) for four different parameters: mutation distribution index, crossover distribution index, mutation probability and crossover probability. For tuning purposes, a population size of 100 and 3000 iterations were used. Table 5.5 summarizes the best parameter settings for the NSGA-II algorithm. Please refer to Appendix A1 for the complete results.

Table 5.5: NSGA-II parameter selection

NSGA-II Parameter Choice	**Experiment Group**	**Final Choice**
Mutation Distribution Index	2, 4, 6, ..., 20	18
Mutation Probability	0.1, 0.2, ... 1.0	0.1
Crossover Distribution Index	2, 4, 6, ..., 20	18
Crossover Probability	0.1, 0.2, ..., 1.0	0.9

Similarly, for the SMPSO algorithm, we compare the volume of the dominated space for two parameters: mutation distribution index and mutation probability. For tuning purpose, a population size of 100 and 3000 iterations were used to test the different settings.

The complete list of selected parameters for the SMPSO algorithm is presented in Table 5.6. Please refer to the Appendix A1 for the complete results of the SMPSO tuning tests.

Table 5.6: SMPSO parameter selection

SMPSO parameter choice	**Experiment group**	**Final choice**
Mutation Distribution Index	2, 4, 6, ..., 20	14
Mutation Probability	0.1, 0.2, ... 1.0	0.1

5.3 Detailed Example

In this section, a detailed example is presented to explain the FPP planning results. The first instance of FPP3010 is solved with the weighted sum method, NSGA-II,

SMPSO and PSONSGA algorithms. The FPP3010 assumes that one needs to plan and design a brand new fog network to accommodate 30 edge-clusters. To achieve this, we need to find the optimal number, location and capacity of fog nodes. Figure 5.3 shows the initial planning area with 30 edge-clusters and 10 potential locations that are uniformly distributed in a $100km \times 100km$ area. Each edge-cluster has resource demands which need to be accommodated.

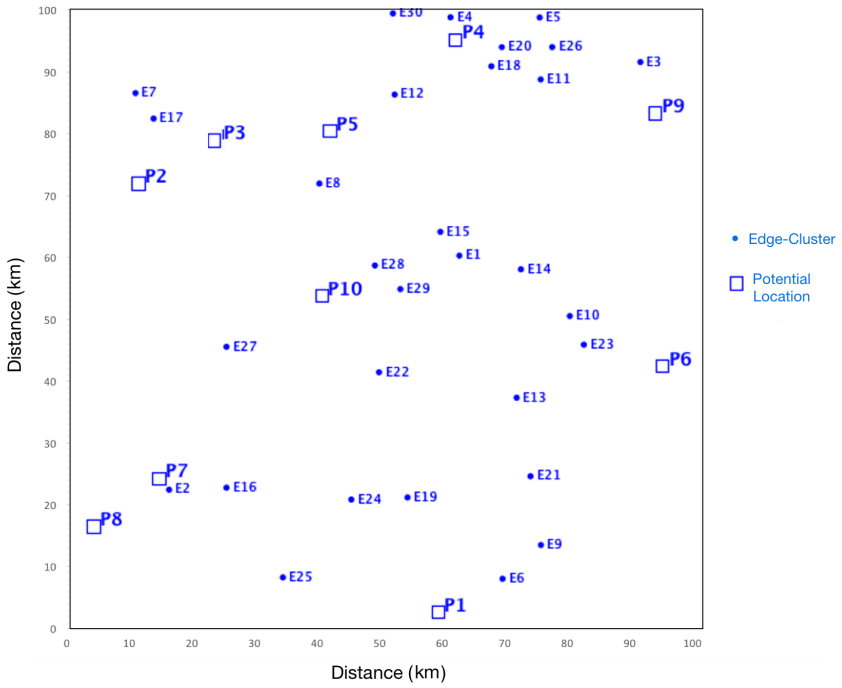

Figure 5.3: Edge-cluster locations and potential locations of fog for FPP3010 (instance-1)

Results from the Weighted Sum Method

The results for problem FPP3010 (instance-1) by using the weighted sum method are shown in Table 5.7. The first column shows the solution index which makes reference to the 11 different weight combinations. The following two columns contain the decision output for the link type and the fog type at each location. Columns 4 and 5 provide the results of the two objective functions (cost and delay respectively) obtained with the weighted sum method. Finally, column 6 shows the relative Mixed-Integer Programming (MIP) gap. The MIP gap represents the percentage gap

between the solution value found by CPLEX and the value of the optimal. For example, a 0.05 relative MIP gap means that CPLEX has found a solution that is five percent from the optimal. Using CPLEX to solve the FPP3010 (instance-1), we can find 7 solution frontiers (as shown by the 7 different solutions from Table 5.7, the duplicated results are grey colored) .

Table 5.7: CPLEX's result for problem FPP3010

Solution number	**Link output**	**Fog output**	**Cost ($)**	**Delay (ms)**	**Gap (%)**
1	1 1 1 1 1 1 1 1 1 1	4 1 4 3 4 4 4 4 4 4	2586500.0	283	0.0
2	1 1 1 1 1 1 1 0 1 1	4 1 4 4 4 4 4 4 0 4 4	2078225.0	283	9.94e-5
3	1 1 1 1 1 1 1 0 1 1	4 1 4 4 4 4 4 4 0 4 4	2078225.0	283	9.98e-5
4	1 1 1 1 1 1 1 0 1 1	4 1 4 4 4 4 4 4 0 4 4	2078225.0	283	9.99e-5
5	0 0 1 1 1 0 1 0 1 1	0 0 4 4 4 0 4 0 4 4	1507350.0	300	0.0047
6	0 0 1 1 1 0 0 0 0 1	0 0 4 4 4 0 0 0 0 4	1004900.0	322	9.80e-5
7	0 0 1 0 0 0 0 0 0 1	0 0 4 0 0 0 0 0 0 4	502450.0	356	7.69e-5
8	0 0 1 0 0 0 0 0 0 0	0 0 4 0 0 0 0 0 0 0	251225.0	384	0.0
9	0 0 1 0 0 0 0 0 0 0	0 0 4 0 0 0 0 0 0 0	251225.0	384	0.0
10	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0.0	449	0.0
11	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0.0	449	0.0

Noticeably, different weight settings may converge to the same objective function values (for example, see solutions 2, 3, 4 from Table 5.7). Figure 5.4 plots the solution frontier obtained from the weighted sum method.

Results from Evolutionary Algorithms

One fundamental difference between single and multiple objective optimization is the number of the solutions. Since several solutions can be optimal, each of these solutions represent a planning and routing scheme which considers a different cost/performance balance. An example of the 49 planning results (Pareto front) obtained with NSGA-II is presented in Table 5.8.

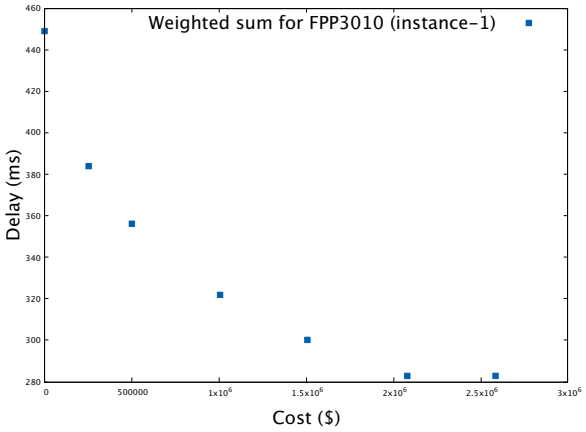

Figure 5.4: Solution frontier of weighted sum (CPLEX)

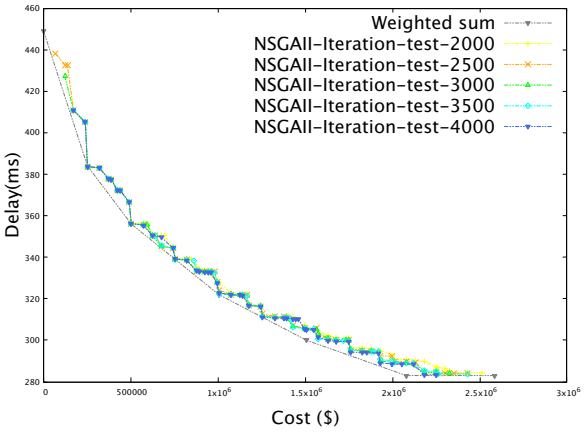

Figure 5.5: Solution frontier of NSGA-II

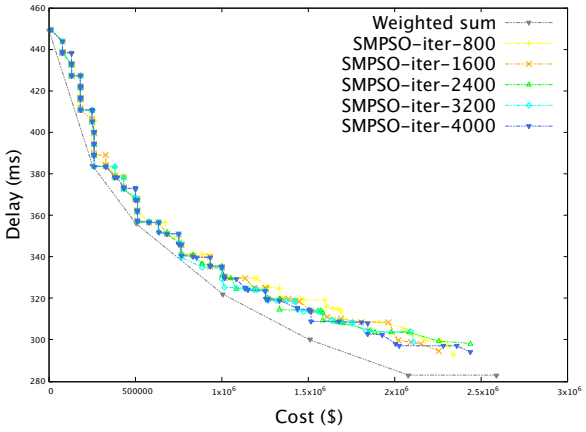

Figure 5.6: Solution frontier of SMPSO

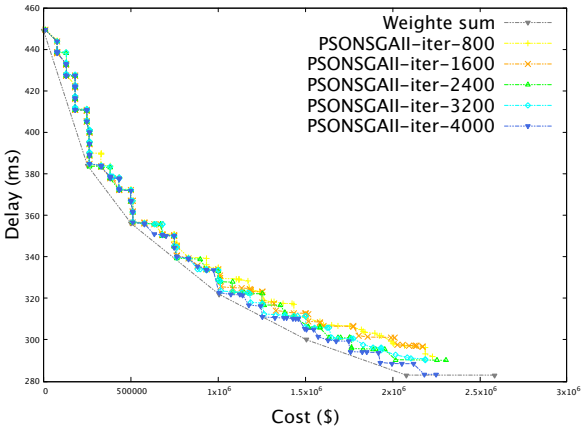

Figure 5.7: Solution frontier of PSONSGA

The first column shows the solution number. The following two columns contain the cost and delay results obtained by the NSGA-II algorithm. Columns 4 and 5 provide, respectively, the decision variables for the link and fog type at each location, where "0" indicates that no facility is installed, and subsequent numbers correspond to different facility types (fog or link). The following two columns show the CPLEX's results (cost and delay values). Since CPLEX only produces 7 solution frontiers, the non-existing solution rows are labelled as "NA". Finally, the cost gaps and delay gaps

between solutions from the NSGA-II and corresponding solution from the weighted sum is provided in the last two columns. The solution frontiers, for FPP3010 by applying NSGA-II, SMPSO, PSONSGA, are plotted in Figures 5.5, 5.6 and 5.7.

Table 5.8: NSGA-II's solution set for FPP3010 (instance-1)

	NSGA-II				Weighted Sum			
Solution num	Cost ($)	Delay (ms)	Variables-link	Variables-fog	Cost ($)	Delay (ms)	Cost diff (%)	Delay diff (%)
1	121225	427.2	0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 2	NA	NA		
2	171225	410.8	0 0 1 0 0 0 0 0 0 0 0	0 0 3 0 0 0 0 0 0 0 0	NA	NA		
3	239650	405.0	1 0 1 0 0 0 0 0 0 0 0	1 0 3 0 0 0 0 0 0 0 0	NA	NA		
4	251225	383.7	0 0 1 0 0 0 0 0 0 0 0	0 0 4 0 0 0 0 0 0 0 0	251225	383.7	0	0.00%
5	319650	383.1	0 0 1 0 0 0 0 0 1 0	0 0 4 0 0 0 0 0 1 0	NA	NA		
6	372450	377.8	0 0 1 0 0 0 0 0 1 0	0 0 4 0 0 0 0 0 2 0	NA	NA		
7	388075	377.6	0 1 0 0 0 1 1 0 0 0	0 1 0 0 0 1 4 0 0 0	NA	NA		
8	422450	372.3	0 0 1 0 0 0 0 0 0 1	0 0 3 0 0 0 0 0 0 4	NA	NA		
9	440875	372.2	0 0 1 0 1 0 0 0 0 1	0 0 1 0 2 0 0 0 0 4	NA	NA		
10	490875	366.6	1 0 1 0 0 0 0 0 0 1	1 0 4 0 0 0 0 0 0 3	NA	NA		
11	502450	356.2	0 0 1 0 0 0 0 0 0 1	0 0 4 0 0 0 0 0 0 4	502450	356.1	0	0.03%
12	570875	355.5	1 0 1 0 0 0 0 0 0 1	1 0 4 0 0 0 0 0 0 4	NA	NA		
13	623675	350.4	1 0 1 1 0 0 0 0 0 0	2 0 4 4 0 0 0 0 0 0	NA	NA		
14	673675	350.0	0 0 1 1 0 0 0 0 0 1	0 0 3 4 0 0 0 0 0 4	NA	NA		
15	692100	350.0	0 1 0 0 0 0 1 0 1 1	0 1 0 0 0 0 4 0 2 4	NA	NA		
16	742100	344.5	1 0 1 1 0 0 0 0 0 1	1 0 4 3 0 0 0 0 0 4	NA	NA		
17	753675	339.3	0 0 1 1 0 0 0 0 0 1	0 0 4 4 0 0 0 0 0 4	NA	NA		
18	822100	338.8	1 0 1 1 0 0 0 0 0 1	1 0 4 4 0 0 0 0 0 4	NA	NA		
19	874900	333.6	1 0 1 1 0 0 0 0 0 1	2 0 4 4 0 0 0 0 0 4	NA	NA		
20	890525	333.4	1 0 1 1 0 0 1 0 0 1	1 0 4 4 0 0 1 0 0 4	NA	NA		
21	924900	333.2	0 0 1 1 1 0 0 0 0 1	0 0 4 4 3 0 0 0 0 4	NA	NA		
22	993325	327.6	0 0 1 1 1 0 1 0 1 0	0 0 3 4 4 0 4 0 1 0	NA	NA		
23	1004900	322.6	0 0 1 1 1 0 0 0 0 1	0 0 4 4 4 0 0 0 0 4	1004900	322.1	0	0.17%
24	1073325	322.0	0 0 1 1 1 0 1 0 1 0	0 0 4 4 4 0 4 0 1 0	NA	NA		
25	1141750	321.8	1 1 0 1 0 1 1 0 0 1	1 1 0 4 0 4 4 0 0 4	NA	NA		
26	1176125	317.1	0 0 1 1 1 0 1 0 1 0	0 0 4 4 4 0 4 0 3 0	NA	NA		
27	1244550	316.4	1 0 1 1 1 0 1 0 1 0	3 0 1 4 4 0 4 0 4 0	NA	NA		
28	1256125	311.5	0 0 1 1 0 0 1 0 1 1	0 0 4 4 0 0 4 0 4 4	NA	NA		
29	1324550	311.3	1 1 0 1 0 1 1 0 0 1	1 4 0 4 0 4 4 0 0 4	NA	NA		
30	1377350	311.2	1 0 0 1 1 1 1 0 1 0	2 0 0 4 4 4 4 0 4 0	NA	NA		
31	1392975	311.0	1 1 0 1 1 1 1 0 0 1	1 4 0 4 1 4 4 0 0 4	NA	NA		
32	1415775	311.0	0 1 0 1 1 1 1 1 0 1	0 4 0 4 3 4 3 1 0 4	NA	NA		
33	1445775	310.9	1 0 1 1 1 1 1 0 1 0	2 0 1 4 4 4 4 0 4 0	NA	NA		
34	1495775	306.1	1 0 1 1 1 1 1 0 1 0	3 0 1 4 4 4 4 0 4 0	NA	NA		
35	1507350	305.6	0 0 1 1 1 0 1 0 1 1	0 0 4 4 4 0 4 0 4 4	1507350	299.9	0	1.87%
36	1575775	301.3	1 0 1 1 1 0 1 0 1 1	1 0 4 4 4 0 4 0 4 4	NA	NA		
37	1628575	300.6	1 0 1 1 0 1 1 0 1 1	4 0 4 4 0 2 4 0 4 4	NA	NA		
38	1678575	300.2	1 0 1 1 1 0 1 0 1 1	3 0 4 4 4 0 4 0 4 4	NA	NA		
39	1697000	300.1	1 1 0 1 1 1 1 0 1 1	4 4 0 4 4 4 1 0 4 2	NA	NA		
40	1747000	299.9	1 1 0 1 1 1 1 0 1 1	4 4 0 4 4 4 1 0 4 3	NA	NA		
41	1758575	295.1	1 0 1 1 1 0 1 0 1 1	4 0 4 4 4 0 4 0 4 4	NA	NA		
42	1827000	294.6	1 0 1 1 1 1 1 0 1 1	4 0 4 4 4 1 4 0 4 4	NA	NA		
43	1929800	292.0	1 0 1 1 1 1 1 0 1 1	4 0 4 4 3 4 4 0 4 4	NA	NA		
44	1998225	289.8	1 1 1 1 1 1 1 0 1 1	4 1 4 4 4 4 4 0 4 3	NA	NA		
45	2009800	289.2	1 0 1 1 1 1 1 0 1 1	4 0 4 4 4 4 4 0 4 4	NA	NA		
46	2078225	289.0	1 1 1 1 1 1 1 0 1 1	4 1 4 4 4 4 4 0 4 4	2078225	283.2	0	2.03%
47	2181025	284.5	1 0 1 1 1 1 1 1 1 1	4 0 4 3 4 4 4 4 4 4	2148225.151	282.8	-0.015038731	0.60%
48	2249450	284.4	1 1 1 1 1 1 1 1 1 1	4 1 4 3 4 4 4 4 4 4	2586500	282.7	0.149836627	0.61%
49	2261025	284.1	1 0 1 1 1 1 1 1 1 1	4 0 4 4 4 4 4 4 4 4	NA	NA		

To better understand the results, we select the solution number 23 from Table 5.8 (cost: $1,004,900, delay: 322.1ms) and plot the corresponding topology for this result

in Figure 5.9. The result for the same expense (i.e. $1,004,900) from Weighted sum, SMPSO and PSONSGA are also plotted respectively, in Figures 5.8, 5.10 and 5.11. The detailed examination shows that both NSGA and PSONSGA generate the same fog network setup (fog node placement, fog node selection, and link selection) as the exact algorithm (weighted sum). For the same capital expenditure, the results from Weighted sum, NSGA-II, SMPSO and PSONSGA give 322.1ms, 322.6ms, 323.8ms and 322.7ms delay respectively. The minor delay differences are due to the misplacement of a small number of edge-clusters (see E11 and E12 for example). These differences can be compensated through user-relocation and load-balancing schemes. This topic has been thoroughly researched in cloud computing, and mature research results can be applied on this relocation subproblem.

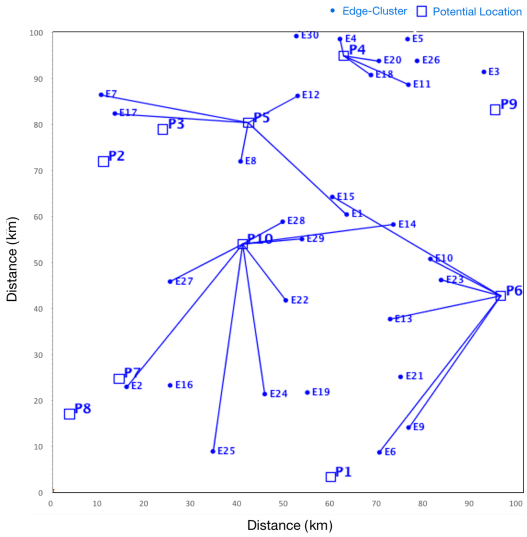

Figure 5.8: Planning result of weighted sum (CPLEX), cost: $1,004,900 delay: 322.1ms

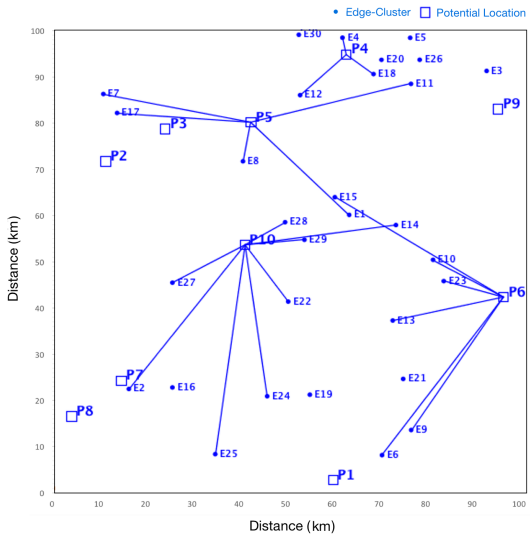

Figure 5.9: Planning result of NSGA-II, cost: $1,004,900 delay: 322.6ms

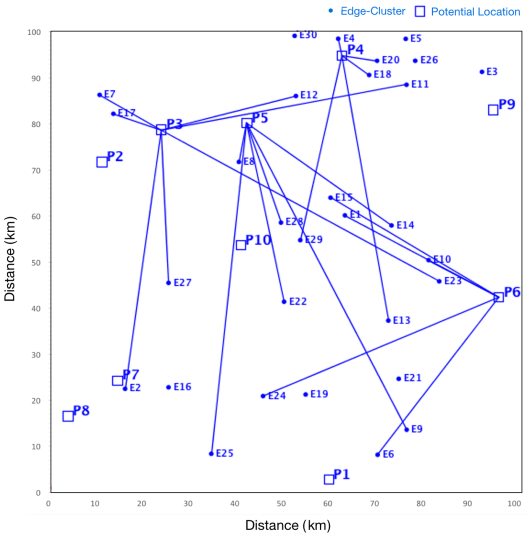

Figure 5.10: Planning result of SMPSO, cost: $1,004,900 delay: 323.8ms

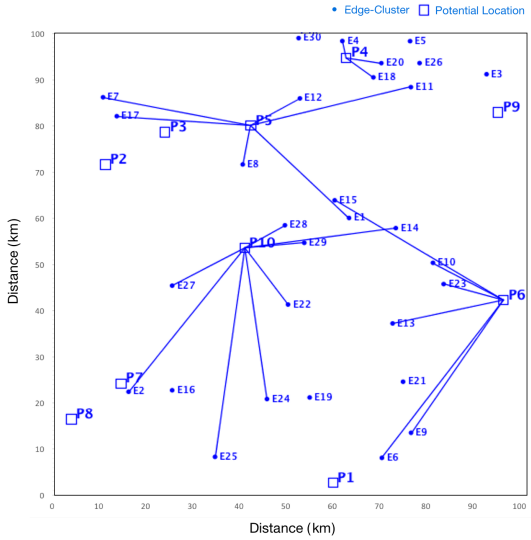

Figure 5.11: Planning result of PSONSGA, cost: $1,004,900 delay: 322.7ms

5.4 Result Analysis

In this section, we present the experiment results to assess the performance of the algorithms. Four instances of 26 different problem sizes are solved by the weighted sum, NSGA-II, SMPSO and PSONSGA algorithms. A population size of 100 with a number of iterations of 10,000 is used for the three evolutionary algorithms. We use the same parameter settings as the ones described in the last section.

5.4.1 HV Indicator Comparison

Since the output of each MO algorithm is a non-dominated solution set, we employ the HV indicator [29] to evaluate and compare the solution quality between the weighted sum algorithm and the three evolutionary algorithms. Table 5.9 shows the HV results for the first instance of the 26 problems. The first column shows the problem number which corresponds to the first column of Table 5.4. The following two columns contain the HV results and the CPU times obtained by solving FPP with the weighted sum method. Column 4 shows the gap (expressed as a percentage) between the solution HV value obtained with the weighted sum, and the best HV value among the four algorithms. The following three columns show the HV value, the corresponding CPU time as well as the gap for the NSGA-II algorithm. The next three columns are the results for the SMPSO algorithm. Finally, similar information for the PSONSGA algorithm is provided in the last three columns. The results for the other three instances are presented in Appendix B1, B2 and B3.

Table 5.9: HV value and CPU time for instance-1

	Weighted Sum			NSGA-II			SMPSO			PSONSGA		
Problem #	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**
1	0.488	1	14.26	0.558	3	0.00	0.558	1	0.00	0.558	1	0.00
2	0.461	1	27.25	0.582	3	0.69	0.575	1	1.91	0.586	2	0.00
3	0.545	1	15.86	0.632	3	0.00	0.623	1	1.44	0.610	2	3.61
4	0.506	7	20.52	0.567	3	7.58	0.591	1	3.21	0.610	2	0.00
5	0.519	8	19.76	0.531	3	17.14	0.614	1	1.30	0.622	2	0.00
6	0.427	10	36.14	0.545	3	6.61	0.554	2	4.87	0.581	4	0.00
7	0.508	9	20.20	0.522	3	17.05	0.578	2	5.71	0.611	4	0.00
8	0.498	46	20.32	0.486	4	23.25	0.566	2	5.83	0.599	5	0.00
9	0.526	45	21.77	0.507	5	26.23	0.640	2	0.00	0.613	7	4.40
10	0.404	41	36.21	0.338	5	62.72	0.510	3	7.84	0.550	7	0.00
11	0.483	50	20.83	0.438	10	33.11	0.549	3	6.19	0.583	12	0.00
12	0.506	62	18.55	0.459	18	30.72	0.548	6	9.49	0.600	16	0.00
13	0.552	698	20.23	0.544	184	22.06	0.639	286	3.91	0.664	251	0.00
14	0.509	854	25.06	0.451	1747	41.02	0.598	301	6.35	0.636	343	0.00
15	0.548	2777	23.25	0.506	1729	33.60	0.676	315	0.00	0.653	339	3.52
16	0.484	7390	29.89	0.425	1757	48.00	0.592	320	6.25	0.629	1324	0.00
17	0.523	4478	22.72	0.467	1891	37.47	0.601	333	6.82	0.642	2538	0.00
18	0.548	8204	16.00	0.432	2055	47.22	0.595	341	6.89	0.636	2724	0.00
19	0.461	9392	28.27	0.340	2145	73.82	0.551	554	7.26	0.591	2833	0.00
20	0.547	20921	17.95	0.411	2175	56.93	0.611	659	5.56	0.645	4858	0.00
21	0.476	22022	27.36	0.362	2078	67.40	0.533	954	13.70	0.606	4962	0.00
22	0.470	11147	27.35	0.300	2126	99.67	0.561	1236	6.77	0.599	3805	0.00
23	0.321	19403	74.07	0.327	2952	70.95	0.559	1444	0.00	0.535	4424	4.49
24	0.356	19476	61.55	0.271	3621	112.18	0.515	1314	11.65	0.575	4172	0.00
25	0.487	25422	18.29	0.295	3323	95.25	0.542	1489	6.27	0.576	4847	0.00
26	0.452	19510	27.86	0.249	5353	132.13	0.518	1639	11.58	0.578	5464	0.00

Since the HV indicator examines both the convergence and diversity properties of a solution set, in these perspectives, PSONSGA algorithm displays its superiority in finding a good quality solution within reasonable computation time. In fact, for all four problem instances (a total of 104 different problems), PSONSGA provides the best HV value for 95 problems (91.3%). The NSGA-II and SMPSO each provides

the highest HV value for 19 different problems (18.2%). The weighted sum approach only provides the best HV value for 2 problems (1.92%).

Figure 5.12 provides a comparison of the HV values amongst the weighted sum, NSGA-II, SMPSO and PSONSGA algorithms for the first instance.

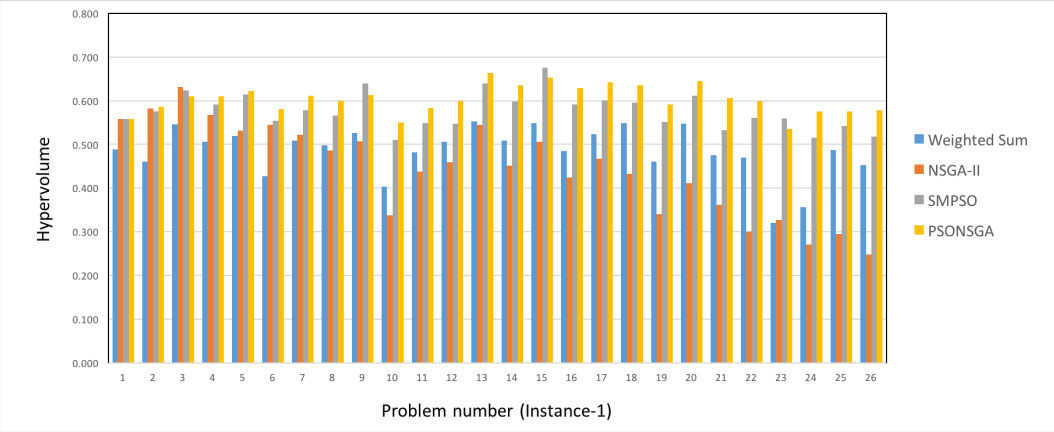

Figure 5.12: HV indicator comparison (instance-1)

As shown in Figure 5.12, the PSONSGA algorithm achieves the best balance between convergence and diversity amongst the three evolutionary algorithms. The weighted sum approach produces worse HV values, because each individual branch and bound search can only produce one single solution. We can also notice that for large-scale problems, NSGA-II's HV values are worse than all three other algorithms. This is due to the larger search space in large-scale problems and the concentrated effect of the non-dominated sorting in the NSGA-II algorithm. Figure 5.13 further presents a complete HV comparison over the four instance sets with a 95% confidence interval. As shown in Figure 5.13, we can reach the same conclusion that PSONSGA outperforms the weighted sum method and the two existing EMO algorithms in HV tests.

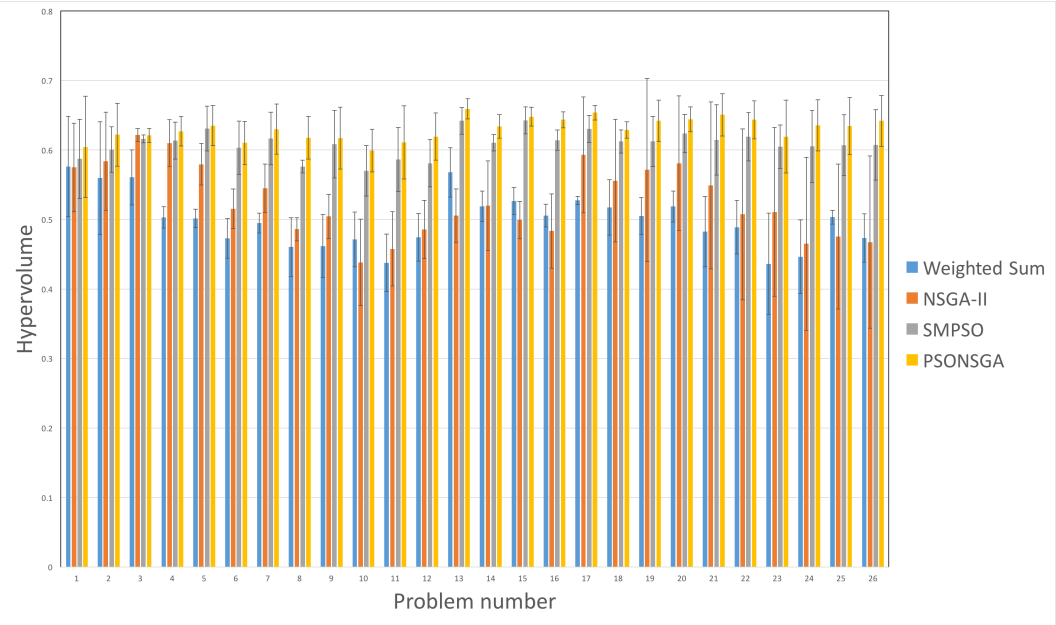

Figure 5.13: HV indicator comparison (over four instance sets)

5.4.2 IGD Indicator Comparison

In this section, we examine the solution quality of the weighted sum, NSGA-II, SMPSO and PSONSGA algorithms by using the IGD quality indicator. As discussed in Chapter 2, the IGD indicator evaluates the quality of a solution set using a reference set. It generates the IGD value through calculating the average euclidean distance between the solution sets and the reference sets. In our evaluation, we use the best non-dominated solutions returned by all three methods as the reference set. The IGD values for the four algorithms are presented in Table 5.10. The first column shows the problem number. Columns 2, 3, 4 and 5 display respectively, the average IGD value for the weighted sum, NSGA-II, SMPSO and PSONSGA. As shown in Table 5.10, PSONSGA displays its superiority over the weighted sum method and the two existing evolutionary algorithms in this evaluation. The average IGD values for PSONSGA's solution achieves the best IGD in 25 of the 26 problems. This improvement over NSGA-II and SMPSO can be explained by the information exchange between the PSO and GA phases in the PSONSGA procedure. This two-phase procedure has been proven to reach better Pareto front results as shown by the IGD indicator comparison.

Table 5.10: IGD indicator comparison (average over four instances)

Problem #	Weighted sum	NSGA-II	SMPSO	PSONSGA
1	$2.16E-01$	$1.01E-02$	$2.01E-03$	$1.43E-03$
2	$1.31E-01$	$9.22E-03$	$2.09E-03$	$1.24E-03$
3	$9.81E-02$	$4.95E-03$	$1.96E-03$	$1.11E-03$
4	$4.11E-02$	$1.24E-02$	$2.46E-03$	$1.97E-03$
5	$3.57E-02$	$1.30E-02$	$3.16E-03$	$3.02E-03$
6	$3.57E-02$	$1.32E-02$	$3.55E-03$	$2.49E-03$
7	$6.12E-02$	$1.17E-02$	$3.73E-03$	$2.81E-03$
8	$9.72E-02$	$1.99E-02$	$3.68E-03$	$2.44E-03$
9	$9.83E-02$	$1.79E-02$	$4.44E-03$	$3.28E-03$
10	$1.13E-01$	$1.73E-02$	$5.32E-03$	$3.83E-03$
11	$1.34E-01$	$2.06E-02$	$4.98E-03$	$3.52E-03$
12	$1.40E-01$	$2.08E-02$	$5.54E-03$	$4.33E-03$
13	$4.47E-02$	$1.57E-02$	$2.37E-03$	$1.48E-03$
14	$5.21E-02$	$1.78E-02$	$2.59E-03$	$1.69E-03$
15	$7.05E-02$	$1.68E-02$	$2.40E-03$	$1.69E-03$
16	$1.24E-01$	$1.70E-02$	$3.11E-03$	$2.05E-03$
17	$1.18E-01$	$1.51E-02$	$3.53E-03$	$2.53E-03$
18	$1.82E-01$	$1.70E-02$	$3.50E-03$	$2.33E-03$
19	$1.70E-01$	$1.52E-02$	$3.32E-03$	$2.59E-03$
20	$3.03E-01$	$1.76E-02$	$3.43E-03$	$2.25E-03$
21	$2.62E-01$	$1.64E-02$	$3.86E-03$	$3.00E-03$
22	$3.03E-01$	$1.92E-02$	$3.61E-03$	$2.03E-03$
23	$3.48E-01$	$1.53E-02$	$3.66E-03$	$2.78E-03$
24	$4.03E-01$	$1.97E-02$	$3.34E-03$	$2.17E-03$
25	$3.68E-01$	$1.65E-02$	$3.66E-03$	$3.97E-03$
26	$5.32E-01$	$1.93E-02$	$4.43E-03$	$4.27E-03$

5.4.3 Delay Gap Comparison

Since we solved the weighted sum formulation with CPLEX solver, each solution point from the CPLEX solver is an optimal solution. In other words, under the same expense condition, the traffic delay produced by the weighted sum and CPLEX can achieve the optimum. Using CPLEX solutions as the reference, we calculate the average delay gaps between CPLEX and the evolutionary algorithms. The average

delay gaps were computed only if the same cost exists in the weighted sum solution sets and CPLEX has found the optimal solutions (the relative MIP equals to zero). Figure 5.14 presents the comparison in terms of delay gaps between the three evolutionary algorithms for problems of first instance. As shown in Figure 5.14, NSGA-II comes in the first place in terms of the delay gaps over 26 problems with an average gap of 0.3% . PSONSGA gives the second best performance with less than or equal to the delay in NSGA-II in 12 of 26 problems with an average gap of 0.4%. An intensive statistical analysis for delay gaps is presented in Table 5.11

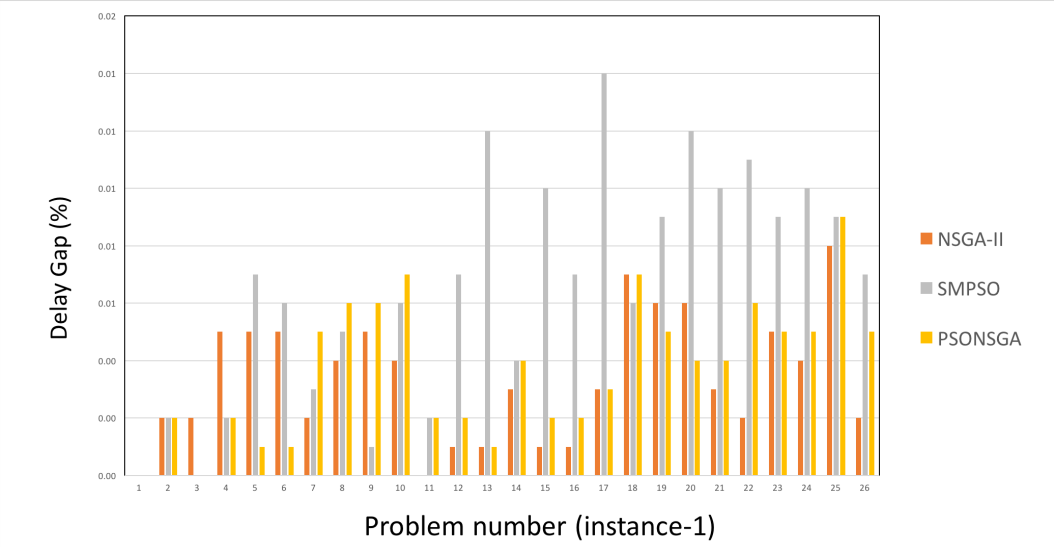

Figure 5.14: Delay gaps (instance-1)

Table 5.11 shows the statistical results over the four instance sets. The first three columns represent the minimum, the maximum, and the average delay gaps. The following two columns are the standard deviation and the 95% confidence interval for the average delay gaps. As shown in Table 5.11, without considering the diversity and distribution in solution sets, NSGA-II gives the best delay gaps to optimal solutions. The reason behind this is the non-dominated sorting in NSGA-II concentrates the solution sets towards the optimal front. However, this sorting process sacrifices the solution diversity for a better convergence quality. This is the same reason why the solutions produced by the NSGA-II algorithm provide worse HV values. PSONSGA achieves the second best among the three evolutionary algorithms.

Table 5.11: Delay gap comparison (over four instance sets)

Algorithms	**Min.gap** (%)	**Max.gap** (%)	**Ave.gap** (%)	**Std.dev** (%)	**95% C.I.** (%)
NSGA-II	0.00	7.80	0.30	0.81	0.30±0.15
SMPSO	0.00	8.50	0.60	1.19	0.60±0.22
PSONSGA	0.00	7.80	0.50	1.12	0.50±0.21

5.4.4 CPU Time Comparison

Figure 5.15 provides the comparison in terms of the CPU time between the weighted sum method and the three evolutionary algorithms for the first instance of problems. Appendixes C1, C2 and C3 contain the corresponding information for the second, third and fourth instances.

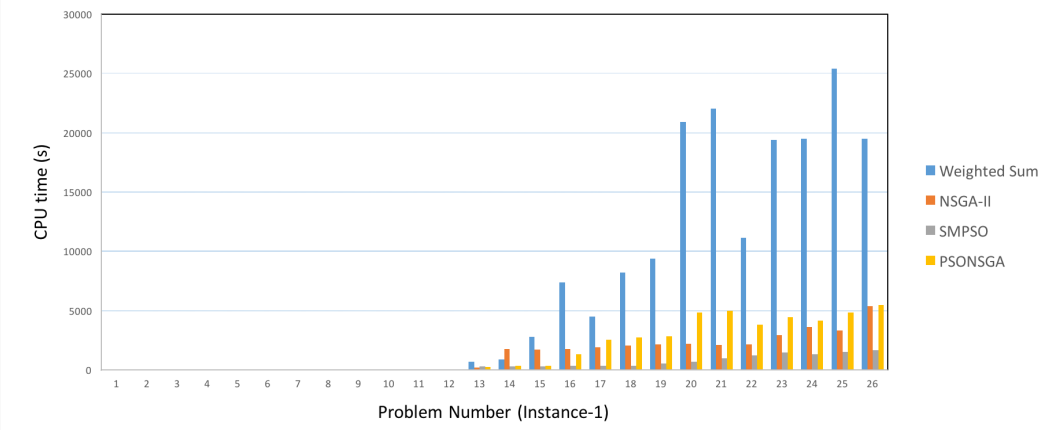

Figure 5.15: CPU time comparison (instance-1)

As shown in Figure 5.15, all three evolutionary algorithms can provide good quality Pareto frontier in a reasonable amount of time. For small-scale problems (problems 1 to 12), the three evolutionary algorithms can finish the optimization within 30 seconds. For large-scale problems, evolutionary algorithms' solution times are still within a reasonable range. For example, to solve problem 26, NSGA-II, SMPSO, PSONSGA, each respectively took 5,303 seconds, 1,639 seconds and 5,464 seconds. In comparison, the weighted sum method using the Cplex solver takes 19,510 seconds, which is

approximately 3.5 times longer than PSONSGA. We also notice that, although fast for small size problems, CPLEX's CPU time is almost increasing exponentially with respect to the problem size. This corresponds with the fact that the fog planning problem is NP-hard as described in Chapter 3.

Figure 5.16 further presents a complete CPU time comparison over the four instance sets with a 95% confidence interval. As shown in Figure 5.16, we can reach the same conclusion that all three evolutionary algorithms can generate close-to-optimal solutions in a reasonable amount of time.

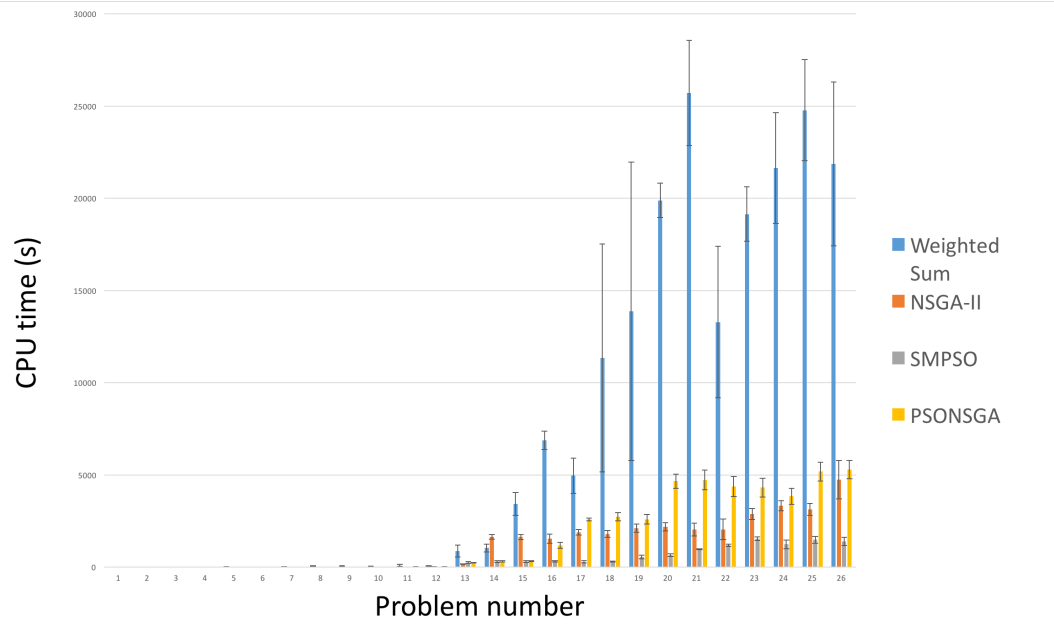

Figure 5.16: CPU time comparison (over four instance sets)

From the delay gap comparison shown in Table 5.11 and the CPU time comparison in Figure 5.16, we can conclude that the evolutionary algorithms are able to use less CPU time to generate close to optimal solutions. At the same time, PSONSGA attains a good balances between the solution convergence and diversity, compared to the two existing algorithms (NSGA-II and SMPSO).

5.5 Summary

To evaluate the FPP model, four instance sets of 26 different problem sizes were solved. The results from the weighted sum, NSGA-II, SMPSO and PSONSGA are

compared in terms of the HV indicator, IGD indicator and delay gaps. From the HV and IGD indicator perspectives, PSONSGA achieves the best HV and IGD values for most of the problem sizes. In contrast, NSGA-II was able to return good solutions in terms of the delay objective value without considering the diversity and coverage of solution frontiers.

The main idea behind approximation algorithms is to find an acceptable tradeoff between the solution quality and the computation time [89]. During the experiment process, we observe that, NSGA-II's non-dominated sorting restricts the solutions from exploring new search space. With the increase of the size of the population, the solution diversity can be improved. However, this obviously increases the computation time, because more offsprings need to be generated, evaluated and sorted in each iteration. In contrast, based on the same initial population size, SMPSO displays good coverage of the whole Pareto frontier. However, after certain iterations, SMPSO may still search a different solution space instead of converging the solution towards the true optimal front. With the increase of the total number of iterations, the solution quality increases. However, to achieve a small amount of solution quality improvement, the total number of iterations needs to be significantly increased, which results in a significant growth of the computation time.

In PSONSGA, the improvement of the solution quality can be explained by the cooperative optimizing between the PSO phase and the GA phase. Compared to the two existing evolutionary algorithms, the proposed algorithm makes use of the advantages of PSO and GA. The PSO phase explores the decision space and preserves the diversity in the solution population. Then, the GA phase emphasizes the convergence of the population towards the optimal front. This two-phase optimization can produce better solution frontiers. More importantly, this improvement of the solution quality does not require a substantial increase of the computation time. Therefore, we can conclude that the proposed PSONSGA algorithm is a promising planning tool to design fog networks.

In the real world of fog planning, the number of edge-clusters can be extremely large, and the network topology, conditions and restrictions will be more complicated. Therefore, using the weighed sum method and CPLEX can be time-consuming. In fact, the proposed algorithm can be more appropriate in this situation. PSONSGA is recommended for solving the fog planning problem as it achieves the best balance between the solution diversity, convergence and computation time.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Although, cloud computing and fog computing share the same virtualization, storage, and networking techniques, the geo-distributed deployment of fog nodes makes the planning fog networks a complex task. We have observed that, there has been little work on the planning aspects of fog networks, and that the weighted sum method has various issues such as the exponential time complexity and a smaller number of solution points.

In order to address these issues, this thesis studies the planning and design of fog computing networks by modelling the FPP as a multi-objective combinatorial model. This model combines the fog placement, the fog dimensioning, and the edge-cluster routing into a joint optimization problem. The goal of this model is to minimize the capital expenditure and the network delay in two separated objective functions in an effort to attain the Pareto optimality. Two existing heuristic multi-objective algorithms (namely NSGA-II and SMPSO) were applied to obtain the optimal solution set. Between these two algorithms, NSGA-II achieves the best performance in terms of the closeness to optimal solutions, while SMPSO performs the best in preserving the diversity in Pareto frontiers. However, both of these algorithms show some disadvantages such as the unevenly diverse results and the convergence gap towards the optimal front. To overcome these limitations, we proposed a new heuristic algorithm, called PSONSGA which combines the convergence and the diversity quality from NSGA-II and SMPSO.

Experimental results show that the PSONSGA outperforms both NSGA-II and SMPSO. Specifically, the hypervolume comparison shows that NSGA-II and SMPSO

give the best HV values, in 19 of 104 different FPPs, while PSONSGA gives the best HV values in 95 of 104 FPPs. Also, the solution comparison shows that PSONSGA algorithm has 95 percent confidence that the mean delay gap is within the interval of $[0.29\%, 0.71\%]$. The experimental results also confirm that evolutionary algorithms can be notably efficient in execution time compared to the weighted sum approach.

6.2 Thesis Contribution

The contribution of this thesis is threefold.

- First, this thesis formulates the FPP as a multi-objective model. The proposed model considers the following objectives: minimizing the total network delay and minimizing the capital expenditure.
- Second, since the exact model has various drawbacks in solving this multi-objective problem, this thesis adopts the heuristic algorithms to solve the planning problem. The EMO algorithms, as the heuristic algorithms, are employed to solve the FPP.
- Third, this thesis proposes a novel EMO algorithm (PSONSGA) which outperforms two existing EMOs in the multi-objective quality indicator tests (HV and IGD). Also, the computation time of the PSONSGA is still within a reasonable range.

6.3 Future Work

6.3.1 Employing EMO on Larger Size Networks

Cisco estimates that there will be 50 billion connected devices by 2020 [2]. The proliferation of mobile devices indicates that fog networks will contain a large number of edge-clusters. Planning and designing a fog computing network with hundreds or even thousands of edge-clusters can be an extremely complicated task. In this context, EMO techniques are considered as a promising approach to complete this task due to its low time-complexity, and convergence efficiency. When the network scales up to hundreds of nodes, the genetic algorithm and the particle swarm algorithm need

to be tuned up to fit this change. It is necessary to propose new searching schemes and fitness functions to obtain the optimal Pareto front results in large-scale FPPs.

6.3.2 Development of Dynamic Fog Planning Model

The cloud and fog infrastructures are typically subject to dynamically changing network demands. It is difficult to make a good fog planning decision that is optimized for all facility sites across all the time periods. The proposed fog network model can be extended to the dynamic fog planning model, in which each fog site can hibernate, wakeup, and relocate according to different network environment and demands. To solve this dynamic fog planning model, the EMO algorithm, especially the PSONSGA algorithm we proposed, can be employed. To be specific, a network flow model could be used to track each fog's hibernation, wakeup and relocate activity across different time slots. In this way, each fog's installation, power down and relocation activities become vertices in the network flow. Thereafter, this network flow can be transferred to the same kind of the multi-objective problem, and solved by using the EMO.

6.3.3 Fog Traffic Management with Software-Defined Networking (SDN)

The experiment results in Chapter 5 show that EMOs' solutions give 1-5% worse delay objective values compared to the optimal solutions due to the difficulties to achieve the global optimum user-allocation. However, the integrated nature of the problem does not imply that we cannot separate the fog placement problem into two subproblems. For instance, it is relatively easy to reassign requests to different facilities. The emergent SDN technique is perfect for this job. The SDN controller separates the control plane from the forwarding plane and provides abstraction between the controllers and the network elements. Since the controllers can dynamically manage the traffic load through the north-bound interface, online traffic management and optimization can be executed in the SDN controller. The cloud service provider can monitor and optimize the traffic in the fog network through workload re-balancing services executed in the SDN controller. The centralized controlling in SDN combined with the efficient multi-objective optimization algorithms such as the EMOs, is a promising research direction.

List of References

[1] W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. Rojas, "Performance modeling to support multi-tier application deployment to infrastructure-as-a-service clouds," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, (Washington, DC, USA), pp. 73–80, IEEE Computer Society, 2012.

[2] D. Evans, "The internet of things how the next evolution of the internet is changing everything," tech. rep., CISCO IBSG, Apr. 2011.

[3] D. Hanes and J. Henry, *Iot Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Fundamentals Series, Cisco Press, 2017.

[4] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[5] S. Guha and S. Khuller, "Greedy strikes back: Improved facility location algorithms," *Journal of Algorithms*, vol. 31, no. 1, pp. 228–248, 1999.

[6] T. Ibaraki and N. Katoh, *Resource allocation problems : algorithmic approaches*. Cambridge, Mass. : MIT Press, 1988.

[7] R. Kohavi, R. M. Henne, and D. Sommerfield, "Practical guide to controlled experiments on the web: Listen to your customers not to the hippo," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, (New York, NY, USA), pp. 959–967, ACM, 2007.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Trans. Evol. Comp*, vol. 6, pp. 182–197, Apr. 2002.

[9] A. Nebro, J. Durillo, J. Garca-Nieto, C. Coello Coello, F. Luna, and E. Alba, "Smpso: A new pso-based metaheuristic for multi-objective optimization," in *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, MCDM 2009 - Proceedings*, pp. 66 – 73, May. 2009.

[10] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley Interscience Series in S, New York, NY, USA: John Wiley; Sons, Inc., 2001.

[11] S. Iturriaga, B. Dorronsoro, and S. Nesmachnow, "Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters," *International Transactions in Operational Research*, vol. 24, no. 1-2, pp. 199–228, 2017.

[12] I. Harris, C. L. Mumford, and M. M. Naim, "A hybrid multi-objective approach to capacitated facility location with flexible store allocation for green logistics modeling," *Transportation Research Part E: Logistics and Transportation Review*, vol. 66, no. Supplement C, pp. 1 – 22, 2014.

[13] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *2014 Federated Conference on Computer Science and Information Systems, FedCSIS*, pp. 1–8, Sep. 2014.

[14] S. S. Lor, L. M. Vaquero, and P. Murray, "In-netdc: The cloud in core networks," *IEEE Communications Letters*, vol. 16, pp. 1703–1706, Oct. 2012.

[15] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 27–32, Oct. 2014.

[16] J. S. G. Pollock, D. Thompson and P. Goldsack., "The asymptotic configuration of application components in a distributed system," tech. rep., University of Glasgow, 2006.

[17] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[18] K. P.Saharan and A. Kumar, "Fog in comparison to cloud: A survey," *International Journal of Computer Applications*, vol. 122, pp. 10–12, July. 2015.

[19] T. Pilioura, *Network Design, Second Edition: Management and Technical Perspectives*. CRC Press, 2004.

[20] K. Deb, *Multi-objective optimization using evolutionary algorithms*, vol. 16. John Wiley and Sons, 2001.

[21] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization," *OR-Spektrum*, vol. 22, pp. 425–460, Nov. 2000.

[22] M. Visée, J. Teghem, M. Pirlot, and E. Ulungu, "Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem," *Journal of Global Optimization*, vol. 12, pp. 139–155, Mar 1998.

[23] B. Dorronsoro, P. Ruiz, G. Danoy, Y. Pign, and P. Bouvry, *Introduction to Evolutionary Algorithms*, pp. 27–47. John Wiley & Sons, Inc., 2014.

[24] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, pp. 33–57, Jun. 2007.

[25] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 257–271, Nov. 1999.

[26] 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*, vol. 7, pp. 117–132, April 2003.

[27] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," tech. rep., 1998.

[28] M. Hansen, A. Jaszkiewicz, and D. T. U. I. for Matematisk Modellering, *Evaluating the Quality of Approximations to the Non-dominated Set*. Technical report (IMM), IMM, Department of Mathematical Modelling, Technical University of Denmark, 1994.

[29] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal distributions and the choice of the reference point," in *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, FOGA '09, (New York, NY, USA), pp. 87–102, ACM, 2009.

[30] A. Gaspar-Cunha, C. Antunes, and C. Coello, *Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 –April 1, 2015. Proceedings*. No. pt. 2 in Lecture Notes in Computer Science, Springer International Publishing, 2015.

[31] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," ASE BDSI '15 Proceedings of the ASE BigData Social Informatics 2015 Article No. 28, Oct. 2015.

[32] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *Netwrk. Mag. of Global Internetwkg.*, vol. 30, no. 4, pp. 46–53, 2016.

[33] Y. Sun and N. Zhang, "A resource-sharing model based on a repeated game in fog computing," *Saudi Journal of Biological Sciences*, vol. 24, no. 3, pp. 687 – 694, 2017. Computational Intelligence Research and Approaches in Bioinformatics and Biocomputing.

[34] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA), pp. 13–16, ACM, 2012.

[35] G. I. Klas, "Open fog computing and mobile edge cloud gain momentum," tech. rep., Vodafone Plc, 2015.

[36] M. Chiang, "Fog networking: An overview on research opportunities," *arXiv preprint arXiv:1601.00835*, vol. abs/1601.00835, 2016.

[37] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. Berger, and L. Dittmann, "Cloud ran for mobile networks—a technology overview," *Communications Surveys and Tutorials, IEEE*, vol. 17, pp. 405–426, Jan. 2015.

[38] M. Peng, Y. Li, J. Jiang, J. Li, and C. Wang, "Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies," *IEEE Wireless Communications*, vol. 21, pp. 126–135, Dec. 2014.

[39] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration, IEEE IRI 2014*, pp. 16–23, 02 2015.

[40] R. Deng, R. Lu, C. Lai, T. Hao Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption," *IEEE Internet of Things Journal*, pp. 1–1, 01 2016.

[41] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J. E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges," *IEEE Network*, vol. 25, pp. 42–46, July. 2011.

[42] K. Hwang, J. Dongarra, and G. Fox, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Elsevier Science, 2013.

[43] A. Khosravi and R. Buyya, "Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art," *Advancing Cloud Database Systems and Capacity Planning With Dynamic Applications*, p. 27, 2017.

[44] K. Alhazmi, M. A. Sharkh, and A. Shami, "Drawing the cloud map: Virtual network provisioning in distributed cloud computing data centers," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2017.

[45] J. Zhang, X. Wang, H. Huang, and S. Chen, "Clustering based virtual machines placement in distributed cloud computing," *Future Generation Computer Systems*, vol. 66, no. Supplement C, pp. 1 – 10, 2017.

[46] A. Khosravi, L. L. H. Andrew, and R. Buyya, "Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data 160;centers," *IEEE Transactions on Sustainable Computing*, vol. 2, pp. 183–196, April. 2017.

[47] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *2013 Proceedings IEEE INFOCOM*, pp. 854–862, April 2013.

[48] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services.," in *NSDI*, vol. 10, pp. 28–0, 2010.

[49] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya, "Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers," *Journal of Parallel and Distributed Computing*, vol. 71, no. 6, pp. 732 – 749, 2011. Special Issue on Cloud Computing.

[50] D. Petcu, "Multi-cloud: expectations and current approaches," in *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pp. 1–6, ACM, 2013.

[51] V. Stantchev and C. Schröpfer, *Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing*, pp. 25–35. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[52] X. Wang, M. Razo, M. Tacca, and A. Fumagalli, "Planning and online resource allocation for the multi-resource cloud infrastructure," in *2014 IEEE International Conference on Communications (ICC)*, pp. 2938–2943, June. 2014.

[53] J. A. Pascual, T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, "Towards a greener cloud infrastructure management using optimized placement policies," *Journal of Grid Computing*, vol. 13, pp. 375–389, Sep. 2015.

[54] W. Chen, I. Paik, and Z. Li, "Cost-aware streaming workflow allocation on geo-distributed data centers," *IEEE Transactions on Computers*, vol. 66, pp. 256–271, Feb. 2017.

[55] R. V. den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads," in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 228–235, July. 2010.

[56] J. L. L. Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Dynamic placement of virtual machines for cost optimization in multi-cloud environments," in *2011 International Conference on High Performance Computing Simulation*, pp. 1–7, July. 2011.

[57] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, pp. 13:1–13:21, Oct. 2014.

[58] A. Ismail and V. Cardellini, *Decentralized Planning for Self-Adaptation in Multi-cloud Environment*, pp. 76–90. Cham: Springer International Publishing, 2015.

[59] X. Wang, J. Cao, and Y. Xiang, "Dynamic cloud service selection using an adaptive learning mechanism in multi-cloud computing," *Journal of Systems and Software*, vol. 100, no. Supplement C, pp. 195 – 210, 2015.

[60] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 337–345, July 2010.

[61] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys, "A constant-factor approximation algorithm for the k-median problem (extended abstract)," in *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, (New York, NY, USA), pp. 1–10, ACM, 1999.

[62] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local search heuristic for k-median and facility location problems," in *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, (New York, NY, USA), pp. 21–29, ACM, 2001.

[63] D. B. Shmoys, E. Tardos, and K. Aardal, "Approximation algorithms for facility location problems (extended abstract)," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, (New York, NY, USA), pp. 265–274, ACM, 1997.

[64] K. Jain, M. Mahdian, and A. Saberi, "A new greedy approach for facility location problems," in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, (New York, NY, USA), pp. 731–740, ACM, 2002.

[65] G. Narzisi, "Classic methods for multi-objective optimization," tech. rep., Courant Institute of Mathematical Sciences New York University, Jan. 2008.

[66] L. Meade and J. Sarkis, "Strategic analysis of logistics and supply chain management systems using the analytical network process1this work was partially supported by nsf grants 9320949 and 9505967, and texas higher education coordinating board atp grant number 003656-036.1," *Transportation Research Part E: Logistics and Transportation Review*, vol. 34, no. 3, pp. 201 – 215, 1998.

[67] H. P. Benson, "An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem," *Journal of Global Optimization*, vol. 13, pp. 1–24, Jan. 1998.

[68] X. L. Sun and D. Li, "Value-estimation function method for constrained global optimization," *Journal of Optimization Theory and Applications*, vol. 102, pp. 385–409, Aug. 1999.

[69] H. Ding, L. Benyoucef, and X. Xie, "A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 6, pp. 609 – 623, 2006. Special Section on Innovative Production Machines and Systems (I*PROMS).

[70] M. Bachlaus, M. K. Pandey, C. Mahajan, R. Shankar, and M. K. Tiwari, "Designing an integrated multi-echelon agile supply chain network: a hybrid taguchi-particle swarm optimization approach," *Journal of Intelligent Manufacturing*, vol. 19, p. 747, June. 2008.

[71] H. R. Cheshmehgaz, M. I. Desa, and A. Wibowo, "A flexible three-level logistic network design considering cost and time criteria with a multi-objective evolutionary algorithm," *Journal of Intelligent Manufacturing*, vol. 24, no. 2, pp. 277–293, 2013.

[72] C. Feng, H. Xu, and B. Li, "An alternating direction method approach to cloud traffic management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 2145–2158, Aug. 2017.

[73] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, "Scalable multi-class traffic management in data center backbone networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, pp. 2673–2684, December 2013.

[74] V. V. Vazirani, *Approximation Algorithms*. New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[75] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM Journal on Computing*, vol. 13, no. 1, pp. 182–196, 1984.

[76] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are np-complete," *Information Processing Letters*, vol. 12, no. 3, pp. 133 – 137, 1981.

[77] J. Vygen, "Approximation algorithms for facility location problem," tech. rep., Research Institute for Discrete Mathematic, University of Bonn, 2004.

[78] I. Y. Kim and O. De Weck, "Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation," *Structural and multidisciplinary optimization*, vol. 31, no. 2, pp. 105–116, 2006.

[79] Q. C. Meng, T. J. Feng, Z. Chen, C. J. Zhou, and J. H. Bo, "Genetic algorithms encoding study and a sufficient convergence condition of gas," in *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings.*, vol. 1, pp. 649–652, 1999.

[80] Y.-C. Hou and Y.-H. Chang, "A new efficient encoding mode of genetic algorithms for the generalized plant allocation problem.," vol. 20, pp. 1019–1034, 09 2004.

[81] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Trans. Evol. Comp*, vol. 6, pp. 58–73, Feb. 2002.

[82] J. OuYang, F. Yang, S. W. Yang, and Z. P. Nie, "The improved nsga-ii approach," *Journal of Electromagnetic Waves and Applications*, vol. 22, no. 2-3, pp. 163–172, 2008.

[83] L. Magnier, *Multiobjective optimization of building design using artificial neural network and multiobjective evolutionary algorithms*. PhD thesis, Concordia University, 2008.

[84] S. Önüt, B. Gülsün, U. R. Tuzkaya, G. Tuzkaya, *et al.*, "A two-phase possibilistic linear programming methodology for multi-objective supplier evaluation and order allocation problems," *Information Sciences*, vol. 178, no. 2, pp. 485–500, 2008.

[85] E. H. Sabri and B. M. Beamon, "A multi-objective approach to simultaneous strategic and operational planning in supply chain design," *Omega*, vol. 28, no. 5, pp. 581–598, 2000.

[86] L. Magnier and F. Haghighat, "Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network," vol. 45, pp. 739–746, 03 2010.

[87] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[88] C. R. Reeves, ed., *Modern Heuristic Techniques for Combinatorial Problems*. New York, NY, USA: John Wiley; Sons, Inc., 1993.

[89] D. S. Johnson, "A theoretician's guide to the experimental analysis of algorithms," *Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges*, vol. 59, pp. 215–250, 2002.

Appendix A

The Parameter Setting Experiments

A.1 Parameter Tests for Evolutionary Algorithms

To determine the best parameter settings for the fog planning problems, we use the HV quality indicator to evaluate the performance between different settings. For the purpose of parameter tuning, we choose the population size of 100 according to the research in [88] as described in Section 5.2.4. To determine the number of iterations, we apply the same experimental method in [9]. The basic idea of this method is to test the number of iterations for multi-objective algorithms to reach the 98% of the HV of the CPLEX's Pareto front. We can observe that all three EMOs can reach the 98% HV of CPLEX within 3000 iterations. Therefore, in parameter tuning tests, a population size of 100 with a number of iterations of 3000 is used for different settings. For each parameter, 10 different experiments are undertaken. The statistical result is drawn from 5 independent runs for each parameter setting. Three problem instances: FPP3010, FPP6010, FPP9010 are chosen in an attempt to cover the small size, medium size and large size of the FPP. The HV mean $\bar{x}$ and standard deviation σ results are shown in the corresponding Table.

For NSGA-II algorithm, we compare the volume of the dominated space (HV) for four parameters: mutation distribution index, crossover distribution index, mutation probability and crossover probability.

Mutation Distribution Index (MDI): An index value governs the proximity of the mutated solution to the parent solution. A large index value gives a higher probability for producing the "near-parent" solutions. In our experiment, the value of MDI takes integer between [2-20] with step size 2. As shown in Table A.1, experiment value 18 shows overall best performance.

Crossover Distribution Index (CDI): An index value governs the proximity of the mutated offsprings to their parents. In our experiment, the value of CDI takes integer between [2-20] with step size 2. As shown in Table A.2, experiment value 18 shows overall best performance.

Mutation Probability (MP): The probability to randomly change each decision variable. In our experiment, the value of MP takes value between [0,1] with step size 0.1. As shown in Table A.3, experiment value 0.1 shows overall best performance.

Crossover Probability (CP): The probability of producing offspring from crossover parents solutions. In our experiment, the value of CP takes value between [0,1] with step size 0.1. As shown in Table A.4, experiment value 0.9 shows overall best performance.

Similar to NSGA-II parameter tests, for SMPSO algorithm, a population size of 100 with a number of iterations of 3000 is used for different settings. For each parameter, 10 different experiments are undertaken. The statistical result is drawn from 5 independent runs for each parameter setting. The HV mean $\bar{x}$ and standard deviation σ results are shown in the corresponding Table. We compare the volume of dominated space for two parameters: mutation distribution index and mutation probability

Mutation Distribution Index (MDI): Similar to the NSGA-II, mutation distribution index governs the proximity of the mutated solution to the parent solution. A large index value gives a higher probability for producing the "near-parent" solutions. In our experiment, the value of MDI takes integer between [2-20] with step size 2. As shown in Table A.5, experiment value 14 shows the overall best performance.

Mutation Probability (MP): The probability to randomly change each decision variable. In our experiment, the value of MP takes value between [0,1] with step size 0.1. As shown in Table A.6, experiment value 0.1 shows overall best performance.

Table A.1: NSGA-II with different mutation distribution index

	MDI-2	MDI-4	MDI-6	MDI-8	MDI-10	MDI-12	MDI-14	MDI-16	MDI-18	MDI-20
FPP3010	$7.03e-01_{4.6e-02}$	$6.30e-01_{5.6e-02}$	$6.22e-01_{4.1e-02}$	$6.30e-01_{3.2e-02}$	$6.43e-01_{5.3e-02}$	$6.53e-01_{2.6e-02}$	$6.41e-01_{8.2e-02}$	$7.05e-01_{7.0e-02}$	$7.16e-01_{4.7e-02}$	$7.51e-01_{6.2e-02}$
FPP6010	$3.23e-01_{2.8e-02}$	$2.61e-01_{8.8e-02}$	$2.87e-01_{5.2e-02}$	$3.67e-01_{1.1e-01}$	$3.32e-01_{6.1e-02}$	$4.00e-01_{4.4e-02}$	$3.98e-01_{4.4e-02}$	$3.74e-01_{6.5e-02}$	$4.11e-01_{4.2e-02}$	$3.86e-01_{6.8e-02}$
FPP9010	$1.53e-01_{1.1e-01}$	$1.45e-01_{9.7e-02}$	$1.90e-01_{9.6e-02}$	$1.82e-01_{7.2e-02}$	$1.77e-01_{7.8e-02}$	$2.19e-01_{5.8e-02}$	$2.23e-01_{8.9e-02}$	$2.87e-01_{1.1e-01}$	$3.03e-01_{5.5e-02}$	$2.63e-01_{8.1e-02}$

Table A.2: NSGA-II with different crossover distribution index

	CDI-2	CDI-4	CDI-6	CDI-8	CDI-10	CDI-12	CDI-14	CDI-16	CDI-18	CDI-20
FPP3010	$6.40e-01_{6.4e-02}$	$6.80e-01_{3.6e-02}$	$6.80e-01_{5.9e-02}$	$6.66e-01_{5.1e-02}$	$6.87e-01_{4.5e-02}$	$7.00e-01_{5.7e-02}$	$7.15e-01_{2.9e-02}$	$7.16e-01_{5.0e-02}$	$7.33e-01_{2.6e-02}$	$7.53e-01_{4.6e-02}$
FPP6010	$2.43e-01_{4.4e-02}$	$2.75e-01_{5.2e-02}$	$3.33e-01_{7.4e-02}$	$3.60e-01_{6.4e-02}$	$3.57e-01_{1.1e-01}$	$4.19e-01_{6.0e-02}$	$4.30e-01_{5.8e-02}$	$3.78e-01_{7.5e-02}$	$4.33e-01_{9.4e-02}$	$4.23e-01_{9.9e-02}$
FPP9010	$2.02e-01_{6.9e-02}$	$2.02e-01_{7.0e-02}$	$2.42e-01_{6.3e-02}$	$1.76e-01_{3.8e-02}$	$2.92e-01_{2.5e-02}$	$2.57e-01_{4.6e-02}$	$2.47e-01_{1.2e-01}$	$2.89e-01_{3.2e-02}$	$3.48e-01_{6.9e-02}$	$2.80e-01_{9.0e-02}$

Table A.3: NSGA-II with different mutation probability

	MP-0.1	MP-0.2	MP-0.3	MP-0.4	MP-0.5	MP-0.6	MP-0.7	MP-0.8	MP-0.9	MP-1.0
FPP3010	$8.11e-01_{4.0e-02}$	$6.99e-01_{6.5e-02}$	$7.08e-01_{3.5e-02}$	$7.44e-01_{5.3e-02}$	$7.55e-01_{3.8e-02}$	$7.89e-01_{6.6e-02}$	$7.89e-01_{5.9e-02}$	$7.93e-01_{8.4e-02}$	$8.04e-01_{7.6e-02}$	$7.72e-01_{5.6e-02}$
FPP6010	$4.85e-01_{6.7e-02}$	$4.16e-01_{4.2e-02}$	$4.77e-01_{3.1e-02}$	$4.52e-01_{5.7e-02}$	$4.70e-01_{4.8e-02}$	$4.70e-01_{4.3e-02}$	$4.30e-01_{3.9e-02}$	$4.71e-01_{3.4e-02}$	$3.72e-01_{5.0e-02}$	$4.04e-01_{1.6e-02}$
FPP9010	$2.63e-01_{6.8e-02}$	$3.65e-01_{1.8e-02}$	$3.13e-01_{8.3e-02}$	$3.20e-01_{7.3e-02}$	$2.86e-01_{5.1e-02}$	$2.54e-01_{2.6e-02}$	$2.43e-01_{2.2e-02}$	$2.29e-01_{2.7e-02}$	$1.54e-01_{5.3e-02}$	$1.26e-01_{5.5e-02}$

Table A.4: NSGA-II with different crossover probability

	CP-0.1	CP-0.2	CP-0.3	CP-0.4	CP-0.5	CP-0.6	CP-0.7	CP-0.8	CP-0.9	CP-1.0
FPP3010	$5.60e-01_{4.7e-02}$	$5.69e-01_{7.9e-02}$	$6.07e-01_{7.5e-02}$	$5.91e-01_{5.2e-02}$	$6.30e-01_{5.6e-02}$	$6.29e-01_{4.4e-02}$	$6.74e-01_{4.2e-02}$	$7.22e-01_{4.9e-02}$	$7.25e-01_{5.1e-02}$	$6.68e-01_{8.2e-02}$
FPP6010	$1.84e-01_{7.5e-02}$	$2.55e-01_{4.7e-02}$	$1.50e-01_{6.5e-02}$	$1.90e-01_{4.8e-02}$	$2.42e-01_{7.2e-02}$	$3.51e-01_{5.1e-02}$	$3.43e-01_{8.5e-02}$	$4.06e-01_{9.6e-02}$	$4.90e-01_{4.8e-02}$	$4.18e-01_{7.6e-02}$
FPP9010	$1.54e-01_{8.6e-02}$	$1.37e-01_{7.0e-02}$	$2.05e-01_{4.8e-02}$	$1.66e-01_{8.6e-02}$	$1.45e-01_{2.6e-02}$	$2.40e-01_{6.2e-02}$	$1.90e-01_{6.3e-02}$	$2.42e-01_{6.2e-02}$	$3.21e-01_{7.9e-02}$	$2.57e-01_{4.8e-02}$

Table A.5: HV for SMPSO with different mutation distribution index

	MDI-2.0	MDI-4.0	MDI-6.0	MDI-8.0	MDI-10.0	MDI-12.0	MDI-14.0	MDI-16.0	MDI-18.0	MDI-20.0
FPP3010	$9.53e-01_{3.3e-02}$	$9.70e-01_{3.6e-02}$	$9.81e-01_{3.9e-02}$	$9.80e-01_{1.8e-02}$	$9.62e-01_{3.8e-02}$	$9.74e-01_{3.2e-02}$	$9.91e-01_{1.9e-02}$	$9.77e-01_{2.8e-02}$	$9.68e-01_{3.0e-02}$	$9.81e-01_{2.6e-02}$
FPP6010	$8.95e-01_{3.2e-02}$	$7.66e-01_{2.3e-01}$	$9.08e-01_{1.2e-02}$	$8.11e-01_{2.1e-01}$	$8.89e-01_{4.4e-02}$	$9.25e-01_{2.0e-02}$	$9.41e-01_{4.8e-02}$	$8.03e-01_{1.1e-01}$	$8.54e-01_{1.6e-01}$	$9.15e-01_{1.5e-02}$
FPP9010	$8.31e-01_{8.8e-02}$	$8.82e-01_{3.9e-02}$	$7.58e-01_{1.9e-01}$	$8.49e-01_{1.5e-01}$	$8.75e-01_{5.9e-02}$	$8.79e-01_{1.0e-01}$	$8.95e-01_{4.7e-02}$	$9.27e-01_{4.5e-02}$	$8.75e-01_{5.9e-02}$	$8.36e-01_{1.6e-01}$

Table A.6: HV for SMPSO with different mutation probability

	MP-0.1	MP-0.2	MP-0.3	MP-0.4	MP-0.5	MP-0.6	MP-0.7	MP-0.8	MP-0.9	MP-1.0
FPP3010	$1.01e+00_{1.7e-02}$	$9.85e-01_{1.9e-02}$	$9.49e-01_{3.4e-02}$	$9.67e-01_{4.1e-02}$	$9.85e-01_{1.9e-02}$	$9.89e-01_{1.3e-02}$	$9.81e-01_{2.6e-02}$	$9.71e-01_{3.9e-02}$	$9.54e-01_{3.9e-02}$	$9.71e-01_{3.9e-02}$
FPP6010	$9.47e-01_{2.7e-02}$	$9.36e-01_{4.2e-02}$	$8.44e-01_{2.0e-01}$	$7.92e-01_{2.9e-01}$	$8.99e-01_{2.5e-02}$	$8.53e-01_{1.2e-01}$	$8.36e-01_{1.4e-01}$	$9.08e-01_{1.2e-02}$	$8.12e-01_{1.1e-01}$	$7.29e-01_{2.3e-01}$
FPP9010	$9.32e-01_{3.9e-02}$	$9.29e-01_{2.6e-02}$	$8.89e-01_{2.7e-02}$	$8.15e-01_{2.0e-01}$	$7.51e-01_{2.9e-01}$	$7.14e-01_{2.8e-01}$	$8.52e-01_{1.3e-01}$	$8.97e-01_{5.4e-02}$	$9.07e-01_{3.8e-02}$	$8.60e-01_{7.8e-02}$

Appendix B

Experiment Results

B.1 HV Value and CPU Time for Instance 2

In this appendix, we present the results for the remaining three instances. Similar to the table presented in Section 5.4.1, the first columns of the tables below represent the problem number. The following two columns contain the HV results and the CPU times obtained from the weighted sum method. Column 4 shows the gap (expressed as a percentage) between the solution HV value obtained with the weighted sum, and the best HV value among the four algorithms. The following three columns show the HV value, the corresponding CPU time as well as the gap for the NSGA-II algorithm. The next three columns are the results for the SMPSO algorithm. Finally, similar information for the PSONSGA algorithm is provided in the last three columns.

	Weighted Sum			NSGA-II			SMPSO			PSONSGA		
Problem #	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**
1	0.627	1	0.00	0.507	1	23.64	0.507	1	23.64	0.507	2	23.64
2	0.550	2	4.34	0.574	2	0.00	0.574	1	0.00	0.574	2	0.00
3	0.620	1	1.85	0.613	1	3.10	0.614	1	2.93	0.632	2	0.00
4	0.478	10	25.85	0.586	2	2.56	0.585	1	2.74	0.601	2	0.00
5	0.510	14	18.69	0.578	3	4.67	0.596	1	1.51	0.605	3	0.00
6	0.469	10	25.52	0.533	2	10.51	0.589	1	0.00	0.589	4	0.00
7	0.476	21	21.62	0.502	3	15.34	0.579	1	0.00	0.579	5	0.00
8	0.467	81	26.31	0.460	2	28.26	0.590	1	0.00	0.590	5	0.00
9	0.456	78	19.64	0.453	3	20.31	0.530	2	2.83	0.545	8	0.00
10	0.504	57	18.17	0.434	3	37.10	0.595	3	0.00	0.595	8	0.00
11	0.388	177	39.73	0.384	10	41.15	0.542	2	0.00	0.542	16	0.00
12	0.489	66	21.71	0.432	11	37.73	0.563	5	5.68	0.595	18	0.00
13	0.596	1388	14.31	0.496	171	37.30	0.675	169	0.89	0.681	233	0.00
14	0.487	1369	30.29	0.508	1468	25.00	0.615	211	3.25	0.635	290	0.00
15	0.504	4463	31.48	0.524	1418	26.53	0.636	214	4.25	0.663	345	0.00
16	0.527	7317	25.51	0.514	1160	28.79	0.632	307	4.75	0.662	912	0.00
17	0.531	6653	22.74	0.564	1683	15.60	0.624	186	4.49	0.652	2475	0.00
18	0.563	22163	14.36	0.644	1665	0.00	0.611	266	5.40	0.644	2904	0.00
19	0.510	28054	29.69	0.630	1759	5.08	0.633	382	4.58	0.662	2394	0.00
20	0.529	20389	17.70	0.623	1827	0.00	0.606	653	2.81	0.623	5169	0.00
21	0.508	29579	35.33	0.688	1454	0.00	0.668	916	2.99	0.688	4933	0.00
22	0.529	13543	27.44	0.552	1105	22.10	0.655	1236	2.90	0.674	5273	0.00
23	0.488	16865	33.70	0.532	2746	22.56	0.626	1444	4.15	0.652	3450	0.00
24	0.461	25971	47.02	0.609	2970	11.33	0.651	828	4.15	0.678	4318	0.00
25	0.507	26141	21.51	0.505	2758	21.98	0.591	1177	4.23	0.616	5496	0.00
26	0.485	26926	37.47	0.538	3051	23.98	0.639	1033	4.38	0.667	4437	0.00

B.2 HV Value and CPU Time for Instance 3

	Weighted Sum			NSGA-II			SMPSO			PSONSGA		
Problem #	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**
1	0.668	1	0.46	0.553	2	21.34	0.631	1	6.34	0.671	2	0.00
2	0.690	1	0.00	0.488	3	41.46	0.598	1	15.44	0.637	1	8.37
3	0.509	1	23.72	0.630	3	0.00	0.619	1	1.78	0.630	2	0.00
4	0.510	6	26.73	0.636	3	1.57	0.627	1	3.03	0.646	2	0.00
5	0.490	6	28.57	0.613	3	2.77	0.630	1	0.00	0.630	2	0.00
6	0.496	11	22.75	0.469	3	29.85	0.609	2	0.00	0.609	4	0.00
7	0.510	10	30.70	0.594	4	12.29	0.647	2	3.09	0.667	5	0.00
8	0.389	40	72.43	0.508	4	31.89	0.570	2	17.54	0.670	6	0.00
9	0.470	37	40.43	0.543	5	21.55	0.660	2	0.00	0.660	8	0.00
10	0.498	37	26.62	0.474	5	33.12	0.605	3	4.30	0.631	6	0.00
11	0.477	41	43.71	0.533	11	28.52	0.660	3	3.79	0.685	11	0.00
12	0.415	56	45.35	0.510	18	18.24	0.574	5	5.05	0.603	18	0.00
13	0.516	544	24.60	0.446	156	44.17	0.626	314	2.72	0.643	246	0.00
14	0.536	1050	17.08	0.628	1572	0.00	0.601	322	4.49	0.608	329	3.29
15	0.510	3082	22.63	0.514	1781	21.60	0.625	277	0.00	0.625	352	0.00
16	0.496	6207	29.64	0.438	1775	46.80	0.623	358	3.21	0.643	1178	0.00
17	0.535	4344	25.35	0.671	1891	0.00	0.651	349	3.07	0.650	2615	3.23
18	0.461	6399	34.77	0.508	1870	22.24	0.603	324	2.99	0.621	2370	0.00
19	0.534	7513	24.81	0.667	2166	0.00	0.646	632	3.25	0.667	2889	0.00
20	0.485	19875	31.09	0.636	2284	0.00	0.606	719	4.95	0.636	4129	0.00
21	0.543	24004	17.76	0.622	2348	2.89	0.609	982	5.09	0.640	5211	0.00
22	0.523	19875	25.77	0.638	2296	3.13	0.637	1137	3.30	0.658	4299	0.00
23	0.514	19209	31.58	0.676	3336	0.00	0.642	1574	5.30	0.676	4778	0.00
24	0.472	18113	36.06	0.440	3549	45.91	0.625	1498	2.72	0.642	3629	0.00
25	0.514	27456	29.20	0.534	3555	24.34	0.637	1623	4.24	0.664	4507	0.00
26	0.432	25411	54.52	0.554	4711	20.40	0.639	1508	4.38	0.667	5410	0.00

B.3 HV Value and CPU Time for Instance 4

	Weighted Sum			NSGA-II			SMPSO			PSONSGA		
Problem #	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**	**HV**	**CPU (s)**	**Gap %**
1	0.522	1	30.66	0.682	3	0.00	0.653	1	4.44	0.682	1	0.00
2	0.537	1	28.57	0.691	3	0.00	0.656	1	5.34	0.691	1	0.00
3	0.567	1	7.87	0.612	3	0.00	0.608	1	0.66	0.612	2	0.00
4	0.520	8	25.31	0.651	3	0.00	0.651	1	0.00	0.651	2	0.00
5	0.487	8	40.46	0.596	3	14.77	0.684	1	0.00	0.684	3	0.00
6	0.500	8	32.37	0.515	3	28.54	0.662	2	0.00	0.662	5	0.00
7	0.485	9	36.78	0.562	3	17.97	0.663	2	0.00	0.663	4	0.00
8	0.488	48	25.13	0.490	4	24.69	0.579	2	5.53	0.611	5	0.00
9	0.396	54	64.24	0.515	5	26.41	0.604	2	7.78	0.651	7	0.00
10	0.480	31	29.38	0.507	5	22.49	0.570	3	8.95	0.621	7	0.00
11	0.403	61	57.13	0.476	8	33.19	0.594	3	6.73	0.634	9	0.00
12	0.488	62	39.17	0.542	20	25.28	0.639	6	6.26	0.679	15	0.00
13	0.607	836	6.89	0.537	145	20.86	0.628	234	3.34	0.649	199	0.00
14	0.543	851	20.93	0.493	1779	33.27	0.628	334	4.62	0.657	262	0.00
15	0.545	3407	19.56	0.453	1613	43.71	0.633	358	2.84	0.651	291	0.00
16	0.515	6591	24.36	0.557	1490	14.90	0.610	264	4.92	0.640	1284	0.00
17	0.520	4349	28.92	0.671	2116	0.00	0.646	268	3.87	0.671	2663	0.00
18	0.497	8557	28.79	0.640	1583	0.00	0.640	263	0.00	0.614	2936	4.23
19	0.515	10547	26.02	0.649	2364	0.00	0.619	591	4.85	0.649	2252	0.00
20	0.514	18369	30.92	0.654	2456	2.91	0.672	514	0.15	0.673	4460	0.00
21	0.403	27241	65.84	0.524	2271	27.67	0.648	1014	3.24	0.669	3786	0.00
22	0.434	8572	48.33	0.540	2629	19.07	0.624	1116	3.04	0.643	4079	0.00
23	0.423	21111	45.25	0.509	2501	20.63	0.592	1704	3.72	0.614	4574	0.00
24	0.497	23001	30.13	0.540	3136	19.81	0.630	1284	2.70	0.647	3225	0.00
25	0.506	20109	34.84	0.568	2874	20.07	0.658	1655	3.65	0.682	5850	0.00
26	0.525	15647	25.00	0.529	5851	24.01	0.633	1364	3.63	0.656	5792	0.00

Appendix C

CPU Time Results

In this appendix, we present the comparison in terms of the CPU time between the weighted sum method and the three evolutionary algorithms for the second, third and fourth instance of the problems. As can be seen from the three figures, the CPU time of weighted sum method increases exponentially with respect to the problem size. In comparison, all three evolutionary algorithms can output high-quality Pareto front solutions within reasonable amount of CPU time.

C.1 CPU Time for Instance 2

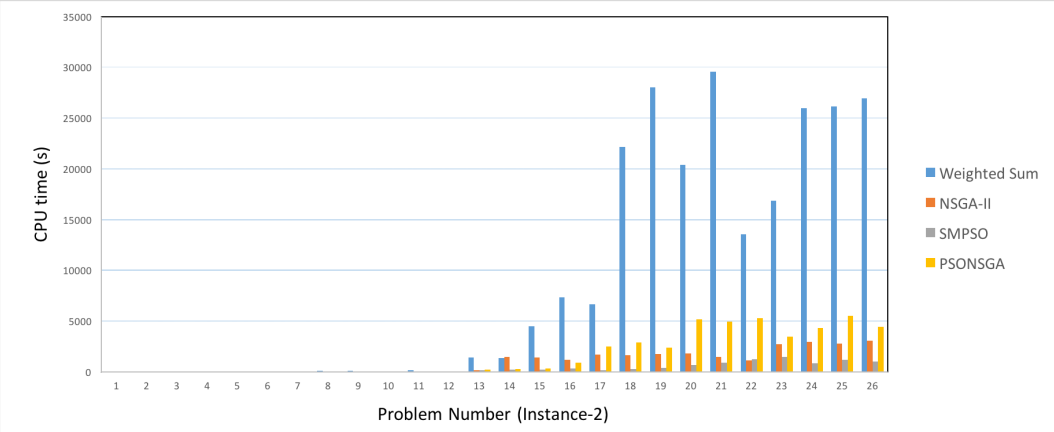

Figure C.1: CPU time comparison (instance-2)

C.2 CPU Time for Instance 3

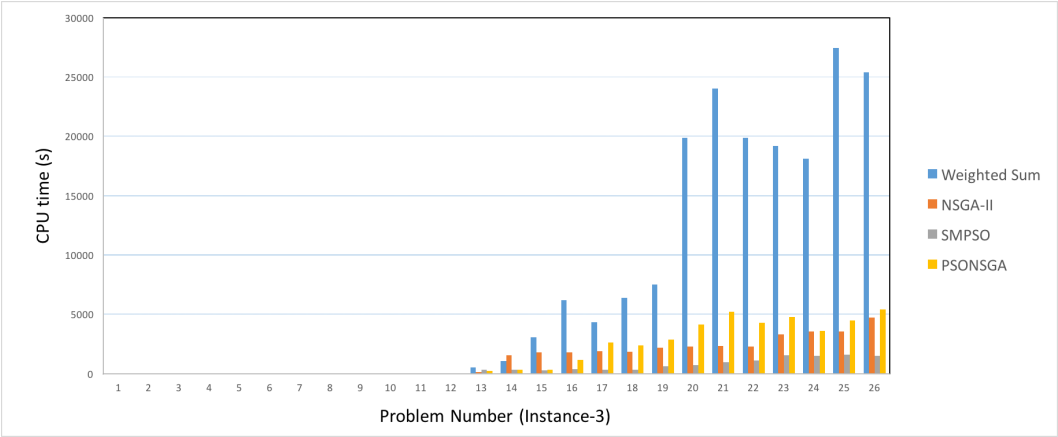

Figure C.2: CPU time comparison (instance-3)

C.3 CPU Time for Instance 4

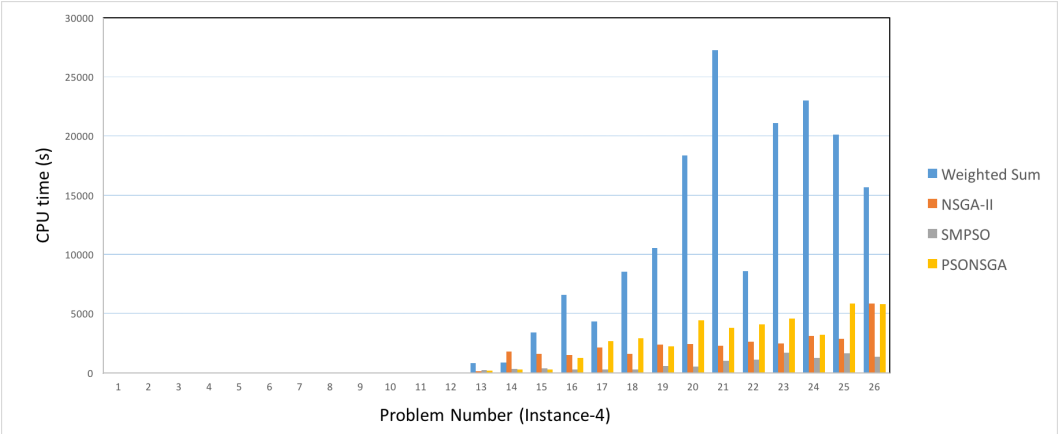

Figure C.3: CPU time comparison (instance-4)