UNIVERSITY OF SOUTHAMPTON

Models and Algorithms for Workforce Scheduling and Routing Problems in Emergency Response Services

by

Fulin Xie

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Social, Human and Mathematical Sciences
Mathematical Sciences

May 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FAULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES
MATHEMATICAL SCIENCES

Doctor of Philosophy

by Fulin Xie

Emergency response services play a key role in protecting public safety and health, and therefore developing effective and efficient response systems is of critical importance. In this thesis, we focus on the workforce scheduling and routing problems (WSRPs) that are commonly faced by emergency response organisations.

We first present a simulation model for real-time emergency vehicle dispatching and routing, developed based on a case study of a British company providing emergency road services. The developed model is used to evaluate system performance, test scenarios and compare the effectiveness of different dispatching policies.

The results of simulation study motivate us to design more advanced heuristic algorithms for the static WSRP. To this purpose, we develop a simple and fast algorithm based on the iterated local search (ILS) framework. The performance of the proposed algorithm is evaluated on benchmark instances against an off-the-shelf optimizer and an existing adaptive large neighbourhood search algorithm. The proposed ILS algorithm is also applied to solve the skill vehicle routing problem, which can be viewed as a special case of the WSRP.

To further improve the decision making, we exploit the stochastic information about future requests and integrate this part of information into the solution method for the dynamic WSRP. A stochastic set-partitioning model is described and integrated with a sampling-based approach. The proposed model uses a two-stage framework, where the first-stage is concerned with finding a set of feasible routes covering known requests, while the second-stage estimates the effect of the same routes with respect to future requests. The performance of the proposed model is evaluated against a deterministic model and a naive greedy heuristic within a simulation framework.

南安普顿大学

摘要

社会、人文与数学科学学院
数学科学系

博士学位

谢福林

应急响应服务在社会安全和公共健康领域中扮演着重要的角色，因此，开发有效并且高效的应急响应系统至关重要。在本论文中，我们重点研究在应急中心以及相关管理部门中经常面对的难题：人员调度和路径问题。

首先，根据英国的一个道路紧急救援服务公司的实际案例，我们设计了一个实时的应急车辆调度与路径规划的仿真模型。该模型主要用于评估现有系统的性能，测试各类运行策略以及比较不同调度方案的有效性。

仿真实验结果表明，开发一个更为有效的启发式算法去解决静态人员调度和路径问题是有必要的。因此，基于迭代局域搜索算法的构架，我们提出了一个简单并且高效的算法。为了验证算法的高效性，我们采用了一些典型算例进行测试，并与现有的优化软件和一个自适应大规模领域搜索算法比较。此外，我们的算法还用来解决带技能约束的车辆路径问题。该类型的问题可以被视为人员调度问题的特列。

为了进一步提高决策质量，我们探索了关于未知服务需求的随机信息，并将此信息整合到动态人员调度问题的解决方法中。我们提出了一个引入采样思想的集合划分模型。该模型使用两阶段的框架：第一阶基于已知的需求信息，设计出可行的人员路径；第二阶段评估该路径方案对未来服务需求的表现。为了证明模型的有效性，我们在一个仿真模型下进行测试，并与一个确定性模型和一个简单的贪婪算法进行比较。

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my deepest gratitude to my superviors, Professor Chris Potts and Professor Tolga Bektaş, who has supported me throughout my PhD studies with their patience, knowledge and kindness. This thesis would not have been possible without their support and guidance.

Secondly, I would like to thank the Automobile Association (AA), which has partilly funded this project. Particular thanks go to Dr. Simon Jones of the AA, for sharing his expert knownledge of the real system and giving suggestions that have helped me develop simulation model and improve visulisation. I would also like to thank Clare Horscroft and James Boffin for their assistant with data collection and analysis.

Moreover, I would like to thank Shenglong Zhou, Shu Pan, Walton P. Coutinho and other fellow PhD students across the maths department for their support and accompny over the years.

I would like to metion my parents for providing moral and financial support throughout my MSc and PhD studies at Southampton. Also, my brother, sister and friends, who are always supporting and encouraging me with their best wishes.

Lastly, I would like to thank my wife, Shanshan Yang, for standing by me through all this.

Declaration Of Authorship

I, **Fulin Xie**, declare that this thesis entitled **Models and Algorithms for Workforce Scheduling and Routing Problems in Emergency Response Services** and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as: Xie, F., Potts, C.N., and Bektaş, T. (2017). Iterated local search for workforce scheduling and routing problems. *Journal of Heuristics*, 23(6):471–500.

Singed:

Date:

Chapter 1

Introduction

This chapter consists of four sections. Section 1.1 provides a brief background of our study, while Section 1.2 gives a detailed description of the problem considered. In Section 1.3, the main goal and objectives of our study are discussed, and finally in Section 1.4, the outline of the thesis is presented.

1.1 Background

The application of operational research (OR) techniques to emergency response services (ERSs) began during the World War II. Since that time, extensive OR work has been carried out on various aspects of ERSs. For example, the survey of Simpson and Hancock (2009) has identified 361 ERSs-related OR articles published between 1965 and 2007. The motivation for the huge amount of research efforts in this field is very simple. ERSs play a key role in protecting public safety and health, and hence developing effective and efficient emergency response systems is of critical importance. In the meantime，various OR techniques including modelling, simulation and visualisation have been recognised as useful tools to aid decision making for complex real-world problems. Despite the large body of OR studies on ERSs, there still exist many areas that have not been explored (Simpson and Hancock, 2009). Moreover, recent advances in computer and telecommunications technologies bring unprecedented opportunities for developing new models and algorithms to enhance the performance of emergency response systems.

The ERSs involve a wide range of decisions and actions taken to address the effects of different emergencies. The most common emergency response organisations are police, fire department and emergency medical services, which belong almost exclusively to the public sector, receiving extensive attention from OR researchers. Some classical problems considered in this domain are designing police patrol districts, locating fire stations, and dispatching emergency vehicles such as ambulances. In addition to the above emergency services, the emergency road service, which provides repair and recovery for disabled vehicles, is also within the context of ERSs. This type of services is usually provided by the private sector as part of the main mission of their business. Compared to other emergency services mentioned above, only few OR studies have been conducted in this area.

Our study focuses on the emergency road service by carrying out a case study of our partner company, the Automobile Association (AA), which is the largest breakdown cover organisation in the UK, maintaining about $2,500$ roadside assistance patrols and recovery trucks in order to provide emergency services for their customers. The company responds to an average of $10,000$ service requests every day.

A brief description of the AA emergency breakdown services is given as follows. When customers have their vehicle break down, they send service requests to the call centre. The system collects information from customers, and determine whether or not to accept the requests. If a requests is accepted, it must be assigned to a technician with required skills and tools, and a service time window is assigned, within which a technician is expected to perform roadside assistance at the customer location. Since a technician may have been assigned several tasks, the system needs to determine the sequence of tasks to be carried out by each technician. If no sufficient resources are available to provide the required service within the time window or if better operational performance can be achieved, tasks can be outsourced (rejected) to a third party, albeit at the expense of additional costs. Since service requests arise continuously and randomly over time, the AA response system is required to make the request acceptance/rejection decisions as well as scheduling and routing decisions in an ongoing fashion as new requests arrive.

The above description implies that the AA emergency services contain two classical optimisation problems: the general scheduling problem and the vehicle routing problem. The combination of these two problems gives rise to the workforce scheduling and routing problems (WSRPs), which are the main problems addressed in our study.

The WSRPs are commonly faced by many service providers, and have applications of home health care, field technician scheduling, security personnel routing and manpower allocation. Due to the complexity of the problems, the majority of existing studies focus on either the routing aspect or the personnel scheduling aspect, and the developed approaches are usually not suitable for the WSRPs. Therefore, it is important to develop new algorithms that are effective and efficient in dealing with WSRPs.

1.2 Problem Description

The workforce scheduling and routing problem (WSRP) refers to a class of optimization problems where service personnel are required to carry out tasks at different locations.

For example, nurses visiting patients at their homes, and technicians performing maintenance jobs in different companies can each be modelled as a WSRP. As service personnel need to travel between different locations, minimising their distances and times for travel is usually considered as one of the objectives when making operational decisions. This results in a routing problem of finding a set of least cost routes for a given workforce, where each route consists of a sequence of locations. Sometimes, tasks have associated time windows, within which service must start. This type of problem can be modelled as an extension of the vehicle routing problem with time windows (VRPTW), which is a well-known variant of the classical vehicle routing problem (VRP).

Service personnel often specialize in different skill domains, and possess skills at different levels. The tasks themselves have different skill requirements. For example, in the telecommunications industry, tasks may include maintenance, installation, construction and repair jobs, and technicians are trained in skills that allow them to only be able to service a subset of these tasks. Thus, skill compatibility must be taken into account to ensure that tasks are performed only by qualified personnel. The associated scheduling problem involves the assignment of tasks to service personnel. In some applications, tasks can be outsourced to a third party, albeit at the expense of additional cost, if appropriate resources are not available to provide the required service, or better operational performance can be achieved. The version of the WSRP that we consider allows for outsourcing.

A closely related problem to the WSRP is the skill vehicle routing problem (Skill VRP), which is introduced by Cappanera et al. (2011). The main feature of the Skill VRP is that the routing costs are defined based on the travelling distance and the technician in such a way that increasing the skill level of the technician causes an increase in costs. The algorithm presented in our study can also be applied to solve Skill VRP.

Depending on the problem setting, the WSRP can be either static or dynamic. In the former case, all parameters are assumed to be known before plans are made and that the designed routing plans are fixed during the execution stage. In the latter case, all or part of the input data are revealed dynamically during the execution stage, and scheduling and routing plans are designed in an ongoing fashion. Our study will consider both the static and dynamic versions of the WSRP.

1.3 Research Objectives

This study aims to develop models and algorithms for scheduling and routing problems arising in emergency response services. The main goal of the study is split into three objectives, given as follows:

1. To develop a simulation model for real-time emergency vehicle dispatching and routing.
2. To design algorithms that are effective and efficient in solving the static WSRP.
3. To exploit stochastic information about future requests to improve decision making for dynamic WSRP.

The first objective is essentially to provide an analytical tool allowing managers to have a better understanding of the real system and also evaluate the performance of different operational strategies. The second objective focuses on heuristic methods to solve the static WSRP. More specifically, we aim to develop a fast and simple algorithm that can be easily adapted to tackle different variants of the WSRP, since the literature review (Chapter 2) reveals that most of the existing algorithms for WSRP are sophisticated and highly problem-oriented. The third objective is concerned with exploiting stochastic knowledge about future requests and incorporating this part of information into the solution methods for the dynamic WSRP.

1.4 Outline of the Thesis

The rest of the thesis is organised as follows. Chapter 2 provides a detailed review of relevant OR work within the context of ERSs, and discusses various solution methods including mathematical programming, heuristics, simulation, and hybrid methods. The relevant body of literature on personnel scheduling problems and vehicle routing problems is also review.

Chapter 3 describes the development of a simulation model for real-time emergency vehicle dispatching and routing. The process of model development including procedures used for data collection, data analysis, distribution fittings, model design as well as model verification and validation are introduced in great detail. The evaluation of model performance and comparison of different operational strategies are also presented in this chapter.

Chapter 4 presents an iterated local search (ILS) for the static WSRP. The proposed algorithm is evaluated on benchmark instances against a mixed integer programming formulation of the problem solved by using an off-the-shelf optimizer and an existing adaptive large neighbourhood search approach. The proposed ILS algorithm is also applied to solve the Skill VRP.

Chapter 5 investigates how to exploit stochastic information about future requests and integrate this part of information into solution methods for the dynamic WSRP. Four mathematical models are developed in this chapter, where two of them are deterministic without using any stochastic information, and the other two incorporate stochastic information about future requests. The proposed models are evaluated within a simulation framework using realistic instances.

Finally, 6 summarises the work of this thesis, discusses the limitation of our models and possible future research directions.

Chapter 2

Literature Review

This chapter begins with an overview of relevant operational research (OR) studies within the context of emergency response services (ERSs). Then various solution methods including mathematical programming, heuristics, simulation, and hybrid methods are discussed. Finally, the relevant studies on general scheduling problems, vehicle routing problems as well as the combination of these two problems are described.

2.1 Emergencies and Operational Research

Over the last several decades, OR techniques have been applied in addressing various decision problems arising in ERSs. The earliest OR articles on ERSs include the studies of the fire stations location problem by Valinsky (1955) and Hogg (1968), and a simulation study of the emergency ambulance service by Savas (1969). Since that time, extensive OR work has been carried out to address various problems arising in ERSs. For example, the survey of Simpson and Hancock (2009) has identified 361 ERSs-related OR articles published between 1965 and 2007. The motivation for the large amount of work in this field is very simple. ERSs play a key role in protecting public safety and health, and hence developing effective and efficient emergency response systems is of critical importance. Moreover, continuing advances in computer and telecommunications technologies have enabled researchers to develop new models and algorithms to enhance the performance of emergency response systems.

The survey of Simpson and Hancock (2009) summarises the research focus of the majority of ERSs studies into the following categories: (1) urban emergency response services including fire, police, patrol, ambulance and emergency medical services; (2) disaster services such as evacuation and rescue; (3) specific hazards which includes typhoon, terrorism, flood, earthquake and wildfire; and (4) other general emergency services. The emergency road services considered in our study fall into the first category. Additionally, the survey has identified the most frequently used OR techniques in the domain of ERSs, which are the mathematical programming, probability and statistics, simulation, decision theory, system dynamics, and queuing theory. These methods will be discussed in detail in Section 2.2.

In addition to the above survey offering a broad overview of OR and ERSs, two earlier reviews conducted by Green and Kolesar (2004) and Altay and Green (2006) focus on the application of OR techniques to specific areas of ERSs. Green and Kolesar (2004) examine the relevant OR studies within the context of urban emergency response systems, while Altay and Green (2006) summarise OR applications to disaster operations. More recent review papers of OR research in disaster operations management are provided by Galindo and Batta (2013) and Caunhye et al. (2012). The former aims to identify recent developments in the area, and also supply a continuation of the survey conducted by Altay and Green (2006), while the latter focuses on optimisation models by discussing the model types, decision variables, objectives and constraints. The location problem of emergency response facilities is also frequently studied in the ERS literature. For this specific subject, Li et al. (2011) conduct a survey to examine various covering models proposed in the area.

The majority of existing ERS studies are concerned with the following problems (Jotshi et al., 2009; Haghani et al., 2004):

(1) Location Problem: identify the locations of response vehicles or service stations within a region in order to satisfy selected performance criteria;

(2) Staffing Problem: determine the minimal number of response vehicles or service stations required in order to fulfil the demand within a given area;

(3) Dispatching and Routing Problem: dispatch response vehicles to service requests under consideration of a number of factors such as priority, skill requirements, stochastic travelling time, and redeployment of resources.

The above problems are often addressed separately by different ERS models. However, there are some studies considering multiple problems simultaneously. For example, Schmid (2012) considers a dynamic ambulance relocation and dispatching problem, where two decisions are required to be made in real-time: (1) after a request arises, an appropriate vehicle needs to be dispatched; (2) after completing the service, the vehicle needs to be relocated to a designated waiting location. Similar relocation and dispatching strategies have been investigated by Andersson and Värbrand (2007a) and Zhen et al. (2014).

With the above overview of relevant OR work on ERSs, the following section focuses on solution methodologies by discussing different OR techniques and which situations they have been applied to. As there is an abundant literature relating to OR work within the context of ERSs, the examination of all the relevant literature would require extensive efforts and time. Moreover, the purpose of our review is not to offer a comprehensive survey of all OR work that have been done in this area, but to provide a general overview of what OR techniques have been applied to ERSs and how. Therefore, the review predominantly considers the relevant papers published in or after 2000.

2.2 Methodology

The surveys of Simpson and Hancock (2009) and Altay and Green (2006) provide a general ranking of OR techniques appeared in the ERS literature, identifying that the mathematical programming including heuristics is the most frequently used method, followed by probability and statistics, and then by simulation. Although the probability and statistics ranks second, they are usually integrated into other OR methods, such as mathematical programming, to handle stochastic aspects of the problems arising in ERSs. Therefore, the following sections mainly discuss mathematical programming, heuristics, simulation and hybrid methods.

2.2.1 Mathematical Programming

The mathematical programming (MP) consists of a number of methods, including linear programming (LP), integer programming (IP), mixed integer programming (MIP), and goal programming, which are described respectively below.

Linear programming (LP) "is an optimisation method for finding the maximum or minimum of a linear objective function, subject to a set of linear equality or linear inequality constraints" (Winston and Goldberg, 2004). It is an important OR technique that has been widely used to solve optimisation problems in various industries such as manufacturing, transportation, banking and education. "A LP problem in which some or all variables are restricted to be non-negative integers is called an integer programming (IP) problem" (Winston and Goldberg, 2004). Since optimisation problems in ERSs usually contain some or all decision variables required to be non-negative integers, the majority of LP models proposed in this area also fall into the scope of the IP.

IP is a powerful technique widely used to formulate real-world problems. It has been applied to solve location, scheduling and routing problems arising in emergency services. For example, Gendreau et al. (2006) develop an IP model to address a location and relocation problem for emergency vehicles, where the objective is to maximise the expected coverage . Curtin et al. (2010) propose an IP model with a maximal covering formulation and a backup covering formulation for the location problem of police patrols. Abounacer et al. (2014) develop a multi-objective IP model to address a location and transportation problem for disaster response.

In terms of scheduling problems, Church et al. (2001) use IP to model a police patrol scheduling system. The application of their model to San Francisco reveals that the model provides better manpower personnel deployment compared to that of the real system. Yan and Shih (2007) propose an IP model to address the work team scheduling problem after a major disaster. Liu et al. (2011) apply IP to solve a hospital and response vehicle scheduling problems to rescue the victims from chemical and biological terrorist attacks.

In addition to location and scheduling problems, IP has also been used to solve routing problems by a number of studies. For instance, Viswanath and Peeta (2003) use IP to formulate a multi-commodity maximal covering network design problem in order to identify critical routes for earthquake response. Chiu and Zheng (2007) apply IP to solve an emergency response and evacuation problem with consideration of different destinations and varying priorities. Zhang et al. (2012) use IP to formulate an emergency response problem with multiple resources, multiple depots and consideration of secondary disasters.

Mixed integer programming (MIP) is an IP in which decision variables are a mixture of those required to be non-negative integers and those which can take real number values (Winston and Goldberg, 2004). It is a useful OR technique that has been practised on a wide range of optimisation problems, such as general scheduling problems, vehicle routing problems and assignment problems. For example, Barbarosoğlu et al. (2002) use MIP to model the helicopter mission planning in disaster relief operations. Their model consists of two levels of MIP models. The top level deals with the helicopter fleet, pilot assignment and the total number of tours to be constructed, while the base level model solves the vehicle routing of helicopters from the operation depot to service locations. Sathe and Miller-Hooks (2005) propose a MIP model with two objectives, maximizing coverage and minimizing cost, in order to solve a location and relocation problem for a fixed fleet of response unites in a transportation network with uncertainty in travel conditions. Pal and Bose (2009) apply MIP to address a location problem of incidence response depots and an assignment problem of response vehicles to these depots.

Goal programming is an variant of LP, which applies when a problem can be formulated as a LP with multiple, normally conflicting, objectives (Winston and Goldberg, 2004). For instance, Alsalloum and Rand (2006) use goal programming to solve a station location problem of emergency response vehicles. Their model contains two goals: the first one is concerned with locating stations to maximise the expected number of demands that are covered within a predefined response time; while the second goal ensures that any demand appearing within the service area is covered by at least one response vehicle. The goal programming is also applied by Kanoun et al. (2010) to address a location problem for a fire and emergency service station in Tunisia, where the conflicting goals are proximity to industrial firms and response time. Lastly, Zhan and Liu (2011) use goal programming to solve a multi-objective model for a location and allocation problem in emergency logistics networks.

2.2.2 Heuristics and Metaheuristics

Although mathematical programming (MP) approaches are very useful, the problem size they can handle is rather limited. Heuristics and metaheuristics provide alternative approaches for dealing with challenging optimisation problems, especially for large-scale problems.

Heuristics are defined as "techniques which seek good (near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases how close to optimality a particular feasible solution is" (Reeves, 1995). Nevertheless, the time taken by an exact method to find the optimal solution to a complex problem, if such a method exists, is normally in a much greater order of magnitude than the heuristic one (Martí and Reinelt, 2011). The computational time is often considered as a crucial measurement which must be taken into account when developing practical algorithms. Therefore, heuristic methods are commonly applied to solve real-world optimisation problems.

Heuristics can be designed as solution approaches for MP models. For example, Yan and Shih (2007) integrate a heuristic algorithm into a mathematical solver to address an IP model proposed for a disaster response team scheduling problem. Their algorithm first decomposes the original problem into smaller sub-problems, each solved by the mathematical solver, and then the obtained solutions are combined to construct a feasible solution which is finally improved by a heuristic method. Zhang et al. (2012) apply a heuristic approach based on linear programming and network optimisation to solve an IP model developed for an emergency response problem with multiple resources, multiple depots and consideration of secondary disasters.

Heuristic algorithms can also be developed without using knowledge of the MP, but such heuristics are often highly problem-oriented, aiming at a specific type of problems or instances. For example, Yan et al. (2009) propose a heuristic method to solve a project scheduling problem of maritime disaster rescue. Their algorithm applies five tailored rules to determine the insertions of tasks in order to generate a schedule with the minimum project duration and the activities start times. Chern et al. (2010) develop a heuristic method to handle a routing problem in the emergency supply chain management. In their algorithm, demands are first grouped and sorted based on their information such as the required product, the due dates and the distances to the depot. Then, these demands are planned individually using a shortest travelling time tree and a minimum cost production tree.

Metaheuristics have also been widely applied to address problems arising in ERSs. Talbi (2009) defines metaheuristic methods as "upper level general methodologies that can be used as guiding strategies in designing underlying heuristics to solve specific optimisation problems". There exist a wide variety of metaheuristics in the literature, and different taxonomy have been proposed, such as nature inspired versus nonnature inspired, memory usage versus memoryless, deterministic versus stochastic, iterative versus greedy, and single-solution based versus population based (Talbi, 2009). In the following review, we use the last criterion to group metaheuristics.

Single-solution based algorithms manipulate and improve a single solution during the search (Talbi, 2009). Commonly used single-solution based metaheuristics include simulated annealing, tabu search, iterated local search, variable neighbourhood search, and guided local search.

Simulated annealing (SA) is a probabilistic technique that enables the local search to escape from the local optima in order to find good solutions for difficult problems. A general framework of a SA algorithm is described as follows. First an initial solution is generated using simple construction algorithms such as greedy heuristic. Then some local search operators are applied to the initial solution to find a neighbourhood, which replaces the incumbent solution immediately if it offers improvement; otherwise, the neighbourhood is only accepted with a given probability, where the probability reduces as time passes. The above local search procedure repeats until the predefined termination rule is met (Van Laarhoven and Aarts, 1987).

Tabu search also accepts worse moves in order to escape from local optima if no improving move can be found, but this is achieved by maintaining a tabu list of the search area that has been previously visited (Talbi, 2009). The tabu list is a memory structure describing the visited solutions or user defined rules. If a solution has been visited within a certain period (usually defined as tabu strength), or it has violated a rule, it is set to "tabu" so that the local search is prohibited from revisiting it until its "tabu" status has been released or it meets a specific rule which is usually called the aspiration criterion (Glover and Laguna, 1997).

In contrast to simulated annealing and tabu search, the iterated local search (ILS) does not accept non-improving solutions, but it ensures the local search to escape from the local optima by performing a perturbation mechanism on the previously visited local optimal solution and restarting the local search from this modified solution (Lourenço et al., 2003). The ILS is one of the most simple and robust algorithm as it has a relatively simple structure and a small number of parameters (Burke et al., 2010).

Variable neighbourhood search (VNS), proposed by Mladenović and Hansen (1997), is a metaheuristic method which successively explores a set of defined neighbourhood structures to get different local optima and to escape from local optima (Talbi, 2009). The fundamental idea of VNS is that a local optima for a neighbourhood structure is not necessary a local optima for another neighbourhood structure, and thus by changing neighborhood structures, different local optima may find and among them there exists a global optimal with respect to all possible neighbourhood structures (Mladenović and Hansen, 1997). Like the ILS, VNS usually contains a perturbation mechanism, also called shaking phase, which is applied to local optima if no improvement can be found by all neighbourhood structures, and then the VNS procedures restart from this modified solution again.

Unlike the metaheuristics described above, the guided local search (GLS) helps the local search escape from the local optima by dynamically modifying the objective function according to the features of already obtained local optima (Talbi, 2009). The key of GLS is the scheme applied to modify the objective function.

The above single-solution based metaheuristics have been used to solve a wide variety of optimisation problems. For applications in the area of emergency services, D'Amico et al. (2002) and Zhang and Brown (2014a) both use simulated annealing to address a partitioning problem of police jurisdiction. Ghandehari and Abdollahi (2014) develop a two-stage simulated annealing algorithm to deal with location and routing problems of large-scale emergency response facilities. Blais et al. (2003) apply tabu search to solve a home-care districting problem. Zheng et al. (2013) also use tabu search to address an emergency equipment maintenance problem in disaster operations. VNS has been used by Grujičić and Stanimirović (2012) and Mišković et al. (2015) to solve location problems in different emergency service networks.

In contrast to single-solution based metaheuristics, population based algorithms maintains a population of solutions, generates new solutions according to population characteristics, and integrates those new solutions into the population based on some selection criteria (Talbi, 2009). Population based metaheuristics can be classified into two main categories: evolutionary algorithms and swarm intelligence.

Evolutionary algorithms are stochastic metaheuristics, which are based on the evolution of a population of artificial individuals (Talbi, 2009). They have been successfully applied to solve complex optimisation problems in various domains. The most popular class of evolutionary metaheuristics is genetic algorithms (GAs), which is first introduced by Holland (1975). The general framework of a GA can be illustrated as followed: first, a population of candidate solutions, coded in genetic representations, are initialised; then new solutions, usually called offspring, are generated using a crossover operator, plus a mutation operator to randomly modify the individuals in order to diversify the population; next, a selection process based on a fitness function is applied to determine survival offspring to replace the parents. The generation of new solutions and the selection process are repeated until certain termination criteria are met (Talbi, 2009).

Swarm intelligence is a category of metaheuristics which are inspired by a collective behaviour of species such as ants, bees, fish and birds (Bonabeau et al., 1999). The main characteristics of swarm intelligence based algorithms are simple agents interacting by an indirect communication medium, and moving in decision space (Talbi, 2009). The most popular metaheuristics within this category are ant colony and particle swarm optimisation.

Population-based metaheuristics have been successfully applied to solve a wide variety of real-world problems. In the area of emergency services, Yang et al. (2007) use a genetic algorithm to address a fuzzy multi-objective programming model proposed for a fire station location problem. Yi and Kumar (2007) apply ant colony to deal with routing and multi-commodity dispatching problem arisen from disaster relief operations. Ant colony has also been used by Liu et al. (2011) to solve a hospital and response vehicle scheduling problems for rescuing victims from chemical and biological terrorise attacks.

2.2.3 Simulation

Simulation has now become one of the most widespread modelling approaches in operations research, management science, and engineering (Rubinstein and Melamed, 1998). Pegden et al. (1995) define simulation as “the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behaviour of the system and evaluating various strategies for the operation of the system”.

Simulation models can be classified based on three dimensions: deterministic or stochastic, static or dynamic, and continuous or discrete (Law, 2007), as shown in Figure 2.1. The term deterministic refers to models which do not contain any random variables, while stochastic models contain random or unpredictable components. In terms of static and dynamic, the former represents a system at a particular time point and the passage of time has no impact on the system, while the latter refers to a system which evolves over time. Lastly, a continuous simulation model represents a system in which the state variables change continuously with respect to time, while in a discrete model, the state variables change instantaneously at separate time points (Law, 2007).

The majority of real-world systems are characterised as dynamic and complex due to the variability, interconnectedness and complexity of the nature of the system (Robinson, 2004). The variability refers to the variations exist in the system, such as stochastic demand and uncertainties in traffic conditions. The interconnectedness means the correlated relationships between components of a system, and thus a change in a component may affect other components that are related to it. The complexity indicates the number of components in the system and the interaction between them.

The complex nature of the real-world systems implies the demand of powerful techniques to help understand and analyse the complex behaviour of systems. Simulation is one of such techniques that allows managers to predict system performance, compare

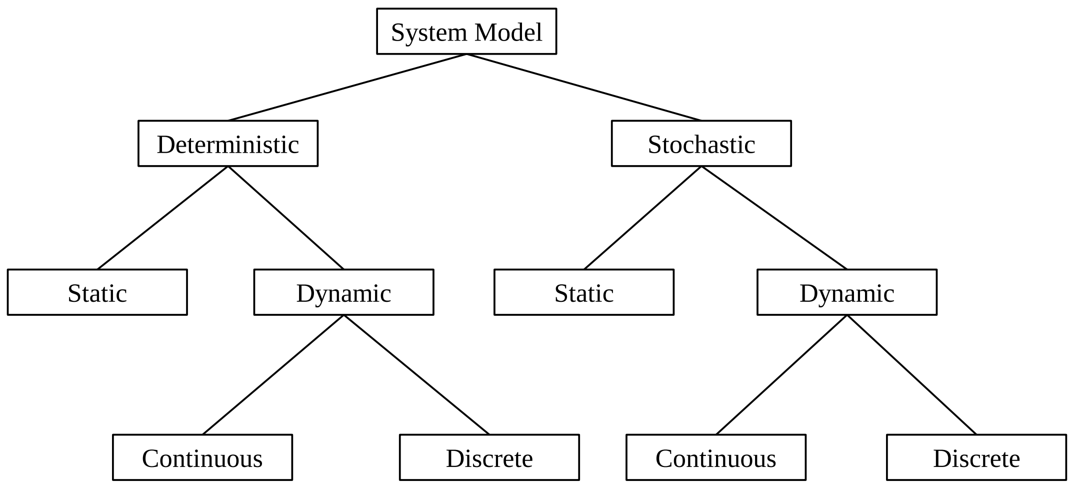


Figure 2.1: Illustration of simulation classification (Law, 2007)

different design strategies and investigate the impact of changes in some components of the system.

Simulation has been widely used in the field of ERSs, especially in emergency medical services (EMSs) due to the increased realism and accuracy it provides (Henderson and Mason, 2004; Yue et al., 2012; McCormack and Coates, 2015). For example, Ingolfsson et al. (2003) develop a discrete-event simulation model for the EMSs in the city of Edmonton in order to evaluate the performance of a single station strategy, where all ambulances begin and end their shift at the same station. The model is also used to examine the effect of various scenarios including the addition of stations, the addition of ambulance, different shifts, and a different redeployment strategy.

The technique of discrete-event simulation has also been used by Haghani et al. (2004), Henderson and Mason (2004), Yue et al. (2012), and McCormack and Coates (2015) to deal with the ambulance planning problems, which involve a number of operational decisions such as ambulance dispatching and routing, ambulance shift design, base station location, and ambulance redeployment strategies. In addition, Andersson and Värbrand (2007b) design a continuous simulation model for the ambulance dispatching and relocation problem, where the model is driven by time using a step of one minute.

In addition to applications of simulation techniques to ambulance planning problems, Zhang and Brown (2013) develop an agent-based simulation model to evaluate the performance of different districting plans for police patrols, where the model is referenced in a later study by Zhang and Brown (2014b) to compare with a discrete event simulation model, proposed for the same problem by Zhang and Brown (2014a). Moreover, Lee et al. (2006) develop a simulation model to assist public health administrators to investigate clinic design and staffing scenarios in emergency response to bioterrorism and infectious disease outbreak.

The problem considered in our study affords a great amount of similarities to the ambulance planning problems. For example, emergency service requests arrive online, response vehicles are required to carry out services at different task locations, and dispatching decisions must be made within a short time frame. Therefore, the above review confirms that simulation is a useful approach to tackle the emergency vehicle dispatching and routing problem at hand.

2.2.4 Hybrid Methods

There are some studies focusing on the development of hybrid methods that combine components of different OR techniques in order to deal with complex problems arising in emergency response services (ERSs). The hybridisation of different algorithmic concepts has become one of the most successful trends in optimisation, as it combines strength and advantages of the individual pure strategies (Blum et al., 2010).

The most commonly used hybrid method is integrating mathematical programming (MP) with heuristics. For example, Iannoni et al. (2011) propose a hybrid approach combining a hypercube queuing model and a greedy heuristic to solve the location and districting problems of large-scale ambulance operations on highways. Geroliminis et al. (2011) integrate a genetic algorithm with a queuing model and a location model to solve a deploying problem of large-scale emergency response vehicles. Marić et al. (2013) develop an algorithm which consists of the evolutionary approach with variable neighbourhood search (VNS) method to address a location problem for long-term health care facilities. Toro-Díaz et al. (2013) develop a hybrid model which combines an integer programming (IP) model and a hypercube queuing model for the location and dispatching problems of emergency medical service, where the model is solved by a genetic algorithm.

In addition to the combination of MP and heuristics, some studies integrate simulation with other OR techniques. For example, McCormack and Coates (2015) integrate a genetic algorithm into a simulation model in order to solve ambulance fleet allocation and base station location problems. Yue et al. (2012) embed an efficient greedy algorithm into a simulation model designed for ambulance fleet allocation and redeployment problems. Lastly, Zhang and Brown (2014a) integrate a simulated annealing algorithm into a simulation model to solve a districting problem for police patrols.

2.2.5 Discussion of Methods

The above review has shown that mathematical programming (MP) methods have been widely applied to address various optimisation problems arising in emergency response services (ERSs). Among those MP approaches, the integer programming (IP) and mixed integer programming (MIP) are the most frequently used techniques, since a large number of real-world applications can be formulated as linear models with some or all decision variables required to be non-negative integers. Although IP and MIP offer very useful ways to deal with the location, scheduling, dispatching and routing problems, the problem size they can handle is rather limited, and therefore heuristics as well as metaheuristics may be preferable, especially for large-scale problems.

Heuristics can be developed based on the knowledge of MP models and used as solution approaches. Also, they can be developed without using information from MP models and applied directly to solve optimisation problems. However, this type of heuristic algorithms are often highly problem-oriented. In contrast to those heuristics, metaheuristics provide a higher-level framework for designing effective and efficient algorithms. There exist a number of metaheuristics, such as simulated annealing, tabu search, generic algorithm, variable neighbourhood search and iterated local search. Those methods have been successfully applied to solve a wide range of scheduling and routing problems.

Simulation is an effective modelling approach to deal with complex systems in ERSs. A large number of researchers have used simulation to evaluate system performance, compare different operational strategies and investigate the impact of changes in some components of the system. The review of relevant literature in this area suggests that the discrete-event simulation is a useful approach for the real-time emergency vehicle dispatching and routing problem considered in our study.

2.3 Personnel Scheduling Problems

Personnel scheduling problems are commonly faced by many industry sectors and service providers, such as call centres, healthcare systems, tourism, retail and manufacturing. The survey of Ernst et al. (2004b) has identified main modules of personnel scheduling problems, including demand modelling, days off scheduling, shift scheduling, line of work construction, task assignment and staff assignment. These modules are briefly described as follows.

Demand modelling is concerned with finding how many staff required at different times over the planning horizon. For example, hospital mangers need to determine the number of nurses required at different times of the day in order to satisfy the nurse-patient ratios guided by healthcare departments. In some organisations, such as telecommunication company, employees are trained in different skill domains and customers have different skill requirements, which forms a complex problem of designing a multi-skilled workforce with different number of employees for each skill domain. This class of problems are often solved using forecasting techniques.

Days off scheduling determines how the rest days are scheduled for each employee over the planning horizon. Recent studies in this area include Elshafei and Alfares (2008), Kyngas and Nurmi (2011) and van Veldhoven et al. (2016).

Shift scheduling involves shift design (e.g., shift length, start time and end time), determination of the number of employees for each shift as well as allocation of suitably qualified employees to specific shifts in order to meet demand. There exist a large number of solution algorithms in this area, including both exact methods (e.g., Maenhout and Vanhoucke (2010), Côté et al. (2013) and Boyer et al. (2014)) and heuristics (e.g., Quimper and Rousseau (2010), Dahmen and Rekik (2015) and Ciancio et al. (2018)).

Line of work construction, also known as tour scheduling, combines both the days off and shift scheduling problems, aiming to design work hours of the day and workdays of the week for each employee. Tour scheduling is one of the most studied problems in the personnel scheduling literature, as it offers flexibility in handling personnel preferences and flexible shift patterns (e.g., different shift start times, shift lengths, and daily break windows) (Van den Bergh et al., 2013). A comprehensive survey of relevant studies in this area can be found in Alfares (2004), where the author has summarised solution techniques for tour scheduling problems into ten categories: (1) manual solution, (2) integer programming, (3) implicit modelling, (4) decomposition, (5) goal programming, (6) working set generation, (7) LP-based solution, (8) construction and improvement, (9) metaheuristics, and (10) other methods.

Task and staff assignment considers what tasks to be carried out during each shift, as well as the assignment of qualified staff to the corresponding shifts in order to meet the specific demand of services. Some studies (e.g., Caramia and Giordani (2009), Cordeau et al. (2010), and Montoya et al. (2014)) have focused on the assignment of tasks to a limited amount of resources, where tasks may require different skills, associate with different priority levels or have time-dependent constraints (e.g., one task needs to start after the completion of another task). In such problems, the objective is usually concerned with finding a feasible scheduling plan such that the number of served tasks is maximized or the schedule length (i.e., the makespan) is minimised. This class of problems can be integrated with vehicle routing problem (VRP), which gives rise to the workforce scheduling and routing problem (WSRP), described in Section 2.5.

The above modules form a wide variety of personnel scheduling problems, which have attracted extensive research efforts in the past decades. For instance, Ernst et al. (2004a) have identified about 700 references in personnel scheduling. Instead of providing a comprehensive review of all relevant studies in this area, we refer the interested reader to the following excellent surveys: Ernst et al. (2004b), Van den Bergh et al. (2013) and De Bruecker et al. (2015). Among them, the first two consider various aspects of personnel scheduling problems by discussing application areas, solutions approached and models, while the last review paper focuses on the skill aspect of scheduling problems and examines various skill modelling approaches. These surveys have identified that mathematical programming, heuristics, and simulation are the most frequently used techniques in the area of personnel scheduling.

2.4 Vehicle Routing Problem

Vehicle routing problem (VRP) is a classical combinatorial optimisation problem, which concerns with finding a set of least cost routes for a fleet of vehicles to serve a set of customers (Golden et al., 2008). Since VRP was first proposed by Dantzig and Ramser (1959), a huge number of studies have been dedicated to applications of VRP to a wide variety of real-world problems, such as school bus routing, goods delivery service, mail distribution and emergency services. There are two important dimensions associated with VRP applications: *evolution* and *quality* of information (Psaraftis, 1980). The former reflects that the available information may change during the execution of the routes, while the latter implies possible uncertainties in the available data. Based on these dimensions, Pillac et al. (2013a) propose four categories for VRPs (the same taxonomy for VRPs has also been introduced in Bektaş et al. (2014)), as described below:

(1) Static and Deterministic

All input data are known in advance and the routing plan is fixed during the execution stage. This class of problems have been studied extensively in the literature, using both exact and approximation methods. The existing exact methods for VRP can be classified into following types: branch-and-band algorithm, dynamic programming, branch-and-cut algorithm, and column generation algorithm. Although exact methods guarantee optimality, they may not always be applicable, especially for large size instances. Instead of finding the optimal solution, approximation algorithms are able to produce good quality solutions in much shorter computational times. There exists a wide body of literature describing various heuristics and meta-heuristics algorithms for VRPs. For recent surveys on these methods, interested readers are advised to Baldacci et al. (2007); Laporte (2009); Vidal et al. (2013a) and Toth and Vigo (2014).

(2) Static and stochastic problems

Part of input parameters are stochastic or unknown at the time of making decisions. In such situations, an a priori plan is established before any random parameters being revealed. During the execution of the plan, the information of random parameters is revealed and only minor adjustments to the a priori plan are allowed. The main sources of uncertainty considered in stochastic VRPs (SVRPs) literature are stochastic demands, stochastic customers and stochastic times (service times, travelling times or both) (Gendreau et al., 2016). Among them, the VRP with stochastic demands (VRPSD) is the most common variant of SVRPs. In VRPSD, the demand of each customer is only revealed on arrival at the customer location. When the demand realization exceeds the remaining capacity of the vehicle, recourse actions are applied to the a priori routes (e.g. vehicle returns to the depot to replenish the capacity and then continues its route). Some of recent studies in this field include Lei et al. (2011), Goodson et al. (2012), Gauvin et al. (2014), Jabali et al. (2014), Biesinger et al. (2016) and Mendoza et al. (2016).

Compared to VRPSD, the VRP with stochastic customers (VRPSC) has been much less investigated. In VRPSC, the presence of customers is considered to be stochastic. The stochastic information on the expected number of customers is available and can be used to design an a priori routing plan. At the execution stage, the set of present customers is revealed and vehicles perform the planned routes by skipping the absent customers. This class of problems has been studied by Bertsimas (1988) and Waters (1989). Some researchers have investigated the VPR with stochastic customers and demands (VRPSCD), which combines the characteristics of VRPSD and VRPSC. For example, Gendreau et al. (1995) propose the first exact algorithm for the VRPSCD based on the integer L-shaped method. However, their algorithm can only solve relatively small instances to optimality. For the same problem, Gendreau et al. (1996) develop a tabu heuristic algorithm.

Some studies have investigated the VRPs with stochastic times (VRPST). In this class of problems, travelling times or service times are considered as random parameters. The problem is usually concerned with finding an a priori routing plan that minimizes the expected routing costs and the expected penalty costs associated with time related constraints (e.g., route duration constraints and customer time windows). Recent studies on VRPST include Li et al. (2010), Lei et al. (2012), Taş et al. (2013a), Taş et al. (2013b), Taş et al. (2014) and Adulyasak and Jaillet (2015)). For more details about static and stochastic VRPs, interested readers are advised to a comprehensive survey conducted by Gendreau et al. (2016).

(3) Dynamic and Deterministic

All or part of the input data are unknown and revealed dynamically during the execution of the routing plan. In this context, the routing plans are designed in an ongoing fashion. This class of problems are usually referred as dynamic, real-time or online VRPs in the literature. The most common source of dynamism is the arrival of customer requests, but demands, travel times and service times are also possible dynamic elements. The solution methods for dynamic VRPs can be divided into two categories: *periodic* and *continuous* reoptimisation. The former refers to the policy that the reoptimisation procedure is only carried out at some predefined time points or at periodic intervals (e.g., Yang et al. (2004); Montemanni et al. (2005); Chen and Xu (2006) and Rizzoli et al. (2007)), while the latter reflects the strategy of performing reoptimisation whenever there is a change or update to the available data (e.g., Gendreau et al. (1999); Ichoua et al. (2000, 2003);

Attanasio et al. (2004) and Bent and Van Hentenryck (2004)). For recent reviews on dynamic and deterministic VRPs, interested readers are referred to Pillac et al. (2013a) and Bektaş et al. (2014).

(4) Dynamic and Stochastic

Similar to the previous group, some input data are unknown and revealed dynamically during the execution of the routes, but the stochastic information on dynamically revealed parameters is available and used in the construction of the routing plan. The solution methods developed in this area can be divided into three categories: anticipatory strategies, stochastic modelling approaches and sampling-based algorithms. The relevant studies for each category of solution methods are discussed in detail in Chapter 5.

2.4.1 Vehicle Routing Problem with Time Windows

The classical VRP is often integrated with additional characteristics and constraints in order to model real-world problems, which results in a significant number of VRP variants, such as multi-depot vehicle routing problem (MDVRP), open vehicle routing problem (OVRP), vehicle routing problem with pickup and delivery (VRPPD), and vehicle routing problem with time windows (VRPTW). The family of those VRP variants have been the subject of intensive research for more than 50 years, which yields an extremely large amount of literature in this area. For example, Eksioglu et al. (2009) have identified 1,021 journal articles with VRP as the main topic, published between 1959 and 2008, while Braekers et al. (2016) have revealed 277 articles on the same subject published between 2009 and 2015. To keep our review manageable, the emphasis of our review is given to VRPTW, since it is highly related to the problem considered in our study. For surveys on other VRP variants, we refer the reader to Eksioglu et al. (2009), Laporte (2009) and Braekers et al. (2016).

A general description of the VRPTW is given as follows: a set of geographically scattered customers with known demands and time windows are serviced by a fleet of vehicles with limited capacity from the same depot. The main objective is to design least cost routes under a set of constraints: (1) every route starts and ends at the same depot; (2) each customer is visited once by exactly one vehicle; (3) the total load of each route cannot exceed vehicle's capacity; (4) the service at each customer should start between the customer's time window (Toth and Vigo, 2014). In some studies, the violation of time window constraints is allowed, which is known as VRP with soft time windows in the literature.

The existing solution methods for the VRPTW can be classified into two categories: exact methods and heuristic algorithms, discussed respectively as follows.

Since the first dynamic programming algorithm for the VRPTW was proposed by Kolen et al. (1987), a number of exact algorithms have been developed for the problem, which can be roughly classified into three categories: dynamic programming, Lagrange relaxation-based methods and column generation (Larsen, 1999). The dynamic programming approach is used by Kolen et al. (1987) to address the VRPTW, where their work is partially inspired by the study of Christofides et al. (1981), which applied dynamic programming to solve the VRP. Lagrange relaxation-based methods have been used to solve the VRPTW by a number of studies, such as Fisher et al. (1997) and Kohl and Madsen (1997). The best exact methods published on the VRPTW are based on the set-partitioning (SP) model, solved by branch-and-price methods, which integrates column generation and branch-and-bound techniques. Desrosiers et al. (1984) present the first column generation algorithm for the VRPTW. Then a more efficient version of the same model is presented in Desrochers et al. (1992), which can find optimal solutions for some large instances containing up to 100 customers. This method is improved by Kohl et al. (1999) by using 2-path inequalities. Jepsen et al. (2008) introduce a branch-and-price framework with subset-row inequalities based on the SP model, which is improved by Desaulniers et al. (2008) by adding generalized $k-$path inequalities and applying a tabu search heuristic to generate negative reduced cost columns in a short computational time. Finally, Baldacci et al. (2011) describe an exact solution framework based on the SP formulation for the VRPTW, enhanced by a new route relaxation called $ng-$route. Recent surveys on exact methods for VRPTW can be found in Baldacci et al. (2012) and Kallehauge (2008).

Although exact algorithms are able to solve the VRPTW to optimality, the problem size they can handle is relatively small. Therefore, there exist a large number of studies focusing on heuristic and metaheuristic approaches for the VRPTW. Some of the successful algorithms developed in this field are tabu heuristic (e.g. Cordeau et al. (2001); Cordeau and Laporte (2001) and Cordeau et al. (2004)), genetic algorithm (e.g. Berger and Barkaoui (2004) and Vidal et al. (2013b)), simulated annealing (e.g. Chiang and Russell (1996) and BañOs et al. (2013)), variable neighbourhood search (e.g. Bräysy (2003); Polacek et al. (2004) and Paraskevopoulos et al. (2008)), adaptive large neighbourhood search (e.g. Ropke and Pisinger (2006)) and iterated local search (e.g. Hashimoto et al. (2008); Michallet et al. (2014) and Vansteenwegen et al. (2009)). For comprehensive surveys on those methods, the interested reader is advised to two excellent review papers by Bräysy and Gendreau (2005a,b), where the former focuses on traditional route construction methods and local search algorithms, while the later examines existing metaheuristics for the VRPTW.

2.5 Workforce Scheduling and Routing Problems

Workforce scheduling and routing problems (WSRPs) combines characteristics of the general scheduling problem and vehicle routing problem (VRP), and refers to a class of optimization problems where service personnel are required to carry out tasks at different locations (Castillo-Salazar et al., 2012). A review of relevant studies on WSRPs is presented in Chapter 4.

2.6 Conclusion

In this chapter, we have reviewed the relevant body of literature on applications of operational research (OR) techniques to various aspects of emergency response services (ERSs). Mathematical programming (MP), heuristics including metaheuristics and simulation have been identified as useful approaches to handle optimisation problems arising in ERSs. Simulation provides an effective way to evaluate system performance, test scenarios and compare the effectiveness of different scheduling and routing policies. This motivates us to develop a simulation model for real-time emergency vehicle dispatching and routing, as presented in Chapter 3.

Among various MP methods, the integer programming (IP) and mixed integer programming (MIP) are the most frequently used techniques in ERSs, since a large number of real-world applications can be modelled as linear models with some or all decision variables required to be integers. In Chapter 4, a MIP model is proposed for the workforce scheduling and routing problem (WSRP). Moreover, the literature review on the WSRPs (as given in Chapter 4) indicates that most of the existing heuristic algorithms for the WSRP are sophisticated and highly problem specific, and hence our study aims to provide a simple and fast heuristic algorithm, based on the iterated local search (ILS) framework. Our work is also the first application of the ILS to the WSRP.

Finally, the review on dynamic and stochastic vehicle routing problems (as presented in Chapter 5) shows that exploiting stochastic information about future requests is advantageous for making dynamic decisions, and sampling-based algorithms provide an effective way for this purpose. However, due to the complexity of sampling-based models, the existing studies have focused on the integration of sampling-based approaches into heuristic algorithms. In Chapter 5, we present a sampling-based model for the dynamic WSRP, where the proposed model can handle reasonable size instances of a real-life problem in short computational times without the need for using heuristic approaches.

Chapter 3

Simulation of Real-Time Emergency Vehicle Dispatching and Routing

Simulation has now become one of the most widespread modelling approaches in operations research (OR), management science, and engineering (Rubinstein and Melamed, 1998). One of the main objectives of this doctoral work is to provide an analytical tool to help managers have a better understanding of the real system and also evaluate the performance of difference operational strategies. To this purpose, a simulation model of the real-time emergency vehicle dispatching and routing is developed based on a case study of a British company providing emergency road services. This chapter starts with a detailed description of the real system and its main components. Then the process of model development including procedures used for data collection, data analysis, distribution fittings, model design as well as model verification and validation are described. Next the results of simulation experiments are presented, and finally the chapter ends with some concluding remarks.

3.1 Background

The emergency road service, which provides repair or recovery for disabled or crashed vehicles, is also within the context of emergency response services (ERSs). However, it has received much less attention from the OR community, compared to other emergency response organisations, such as police, fire department and emergency medical services. Our study focuses on the emergency road service by carrying out a case study of a British company, the Automobile Association (AA), which is the largest breakdown cover organisation in the UK, maintaining about $2,500$ roadside assistance patrols and recovery trucks in order to help their customers whose cars break down on their way.

The AA service operates 24 hours a day and 7 days a week, and responds to an average of $10,000$ requests every day. Since service requests arise continuously and randomly over time, the AA response system is required to make the request acceptance/rejection decisions as well as scheduling and routing decisions in an ongoing fashion as new requests arrive. The following sections illustrate how those decisions are made by discussing the AA resources, tasks, and the decision making process.

3.1.1 AA Resources

The AA motoring breakdown service consists of roadside repair service and recovery operation. The former is carried out by mobile patrols with skilled technicians. If a roadside repair is not possible, the recovery operation is required and is carried out by trucks with recovery technicians (hereafter referred to as truckers) or those patrols that are equipped with the vehicle recovery system allowing patrols to lift the front or rear of disabled vehicles in order to move them from original locations to recovery destinations. However, the patrol recovery operation has restrictions on weight, transmission, the number of passengers (patrols only have two passenger seats), and recovery distance (up to 60km from patrol home bases). If a recovery task violates any of the above restrictions, it must be assigned to a recovery truck.

Technicians are specialised in different skill domains, which can be classified into following categories: (1) battery technician, who has skills of dealing with battery issues (e.g., flat battery) and minor repair tasks (e.g., tyre punctures); (2) fuel technician, who is specialised in handling fuel issues (e.g., fuel drain) and minor repair tasks; (3) key technician, who addresses key issues only, such as key lost and key locked in the vehicle; (4) JLR technician, who is specialised in particular brands of vehicles, such as Jaguar and Land Rover; (5) general technician, who can deal with both minor and major repair tasks (e.g., engine and gearbox issues).

Moreover, each technician have an associated home base, at which the technician starts and ends the shift. A period of 45 minutes must be reserved to allow technicians to drive back to their home bases at the end of their shifts. In contrast to technicians, truckers start and end their shifts at designated depots.

In addition to the AA resources, requests can be outsourced to the garage (the third party providing motoring breakdown services), albeit at the expense of additional costs. The garage resource is split into two categories: garage A and garage R. The former is equivalent to AA patrols that can carry out roadside repair tasks, while the latter is similar to AA trucks which can perform recovery operation.

3.1.2 AA Tasks

When a service request arrives at the call centre, the call handler collects information from the customer, and then creates a task characterised by location, faulty type, and a service time window within which the service is expected to commence. The fault type (e.g., tyre punctures, engine issues, and flat battery) determines the skill requirement of the task, which belongs to one of the following categories: (1) battery assistance; (2) fuel assistance; (3) key assistance; (4) JLR (Jaguar/Land Rover) assistance; (5) repair 1 (minor); (6) repair 2 (major); (7) recovery 1 (truck not required) and (8) recovery 2 (truck required). Recall that technicians are trained in different skills, which allow them to only be able to perform a subset of task categories. Therefore, skill compatibility (as illustrated in Figure 3.1) must be taken into account to ensure that tasks are assigned only to qualified personnel.

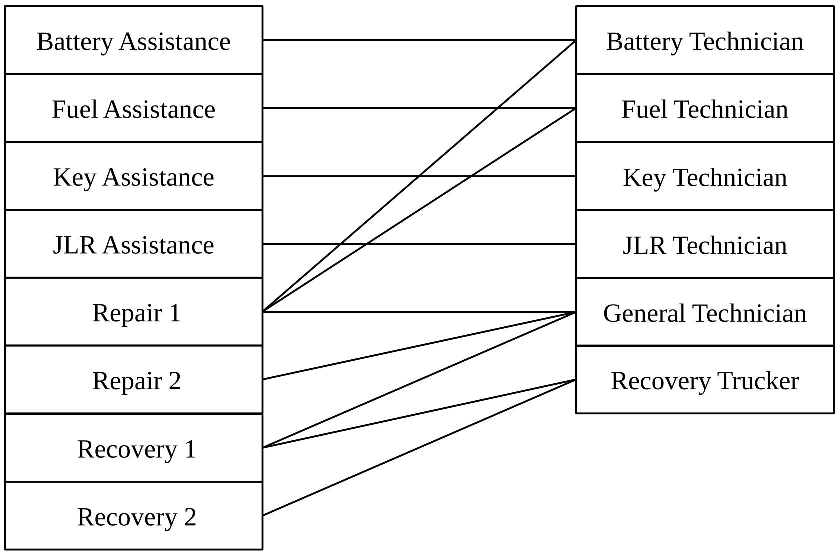


Figure 3.1: Skill compatibility between tasks and resources

3.1.3 Service Process

Figure 3.2 illustrates the decision making process of the real system. When customers have their vehicles break down, they send service requests to the call centre. The call handlers collect information from customers, create tasks and determine the skill requirements. Then the system identifies whether there are available resources to perform tasks within the required time limit (e.g., two hours). If no resource is available or better operational performance can be achieved, tasks are outsourced to the garage; otherwise, tasks are assigned to appropriate resources under the consideration of skill compatibility. Repair tasks must be assigned to technicians, while recovery tasks can be assigned to either technicians or truckers depending on the requirement of truck.

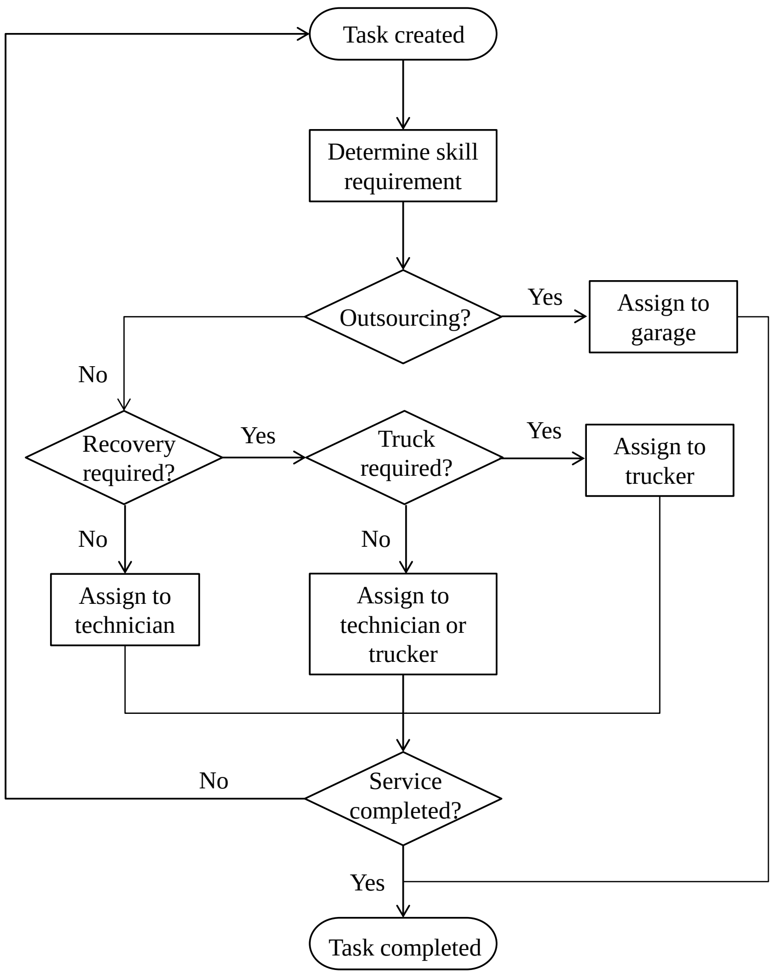


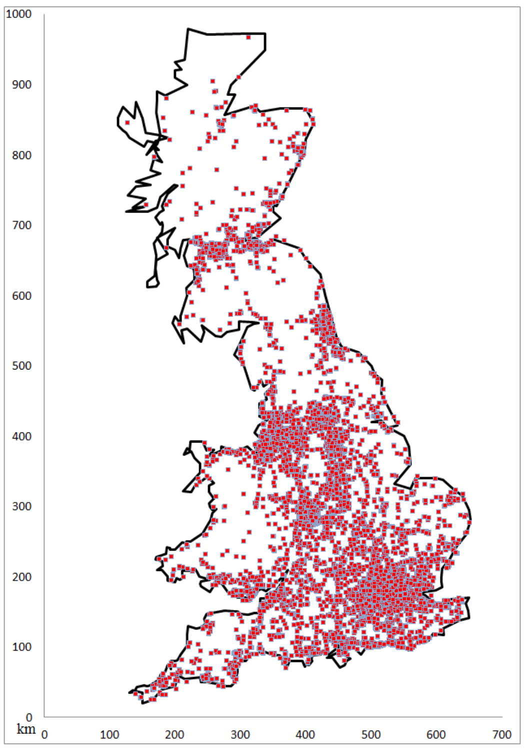
Figure 3.2: A flowchart of the decision making process at the AA

Once assignment decisions are made, tasks are inserted into job plans of the technicians or truckers assigned. Each job plan consists of all tasks that have been assigned but not yet completed. The system needs to determine the sequence of tasks to be carried out by the associated resource. Once a technician or trucker starts travelling to a task, they cannot serve another task until the current one is completed. In this case, the task is defined as locked, which means that it cannot be reassigned to another resource. Technicians and truckers are allowed to swap unlocked tasks in their job plans, however, the skill compatibility must be always taken into account.

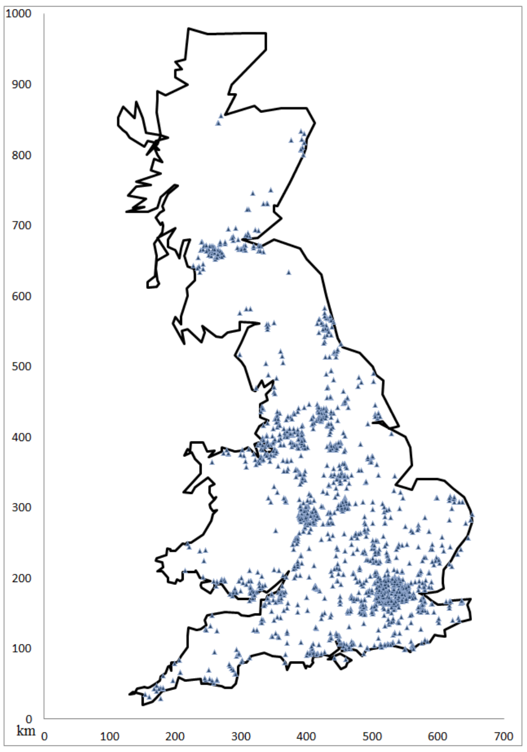
After arriving at the task location, the technician carries out detailed vehicle diagnostics to identify what the fault is, whether the issue can be repaired at roadside and how long the service will take. This part of information is sent back to the system to determine whether further actions are required. If the service cannot be completed by the resource assigned (e.g., the resource does not have the required skills and tools), the current task is closed and a new task is created for the same customer in order to dispatch a more appropriate resource or use the garage option. Once the current task is completed, the technician or trucker is required to travel immediately to the next task location assigned or if there are no further tasks in the job plan, the technician can stay idle at the current location but the trucker must travel back to the associated depot.

3.2 Data Collection and Description

Two data sets, task data and staff data, covering the entire UK and a time period from 01/05/2014 to 31/05/2014 are collected from the company. The task data contain the following categories: task creation time (request arrival time), task location, skill requirement, initial service duration, response vehicle arrival time, task completion time, recovery destination and recovery start time if the recovery service has been carried out. The staff data contain scheduled shifts of technicians and truckers, with the following categories: location of each technician home base, location of the associated depot of each trucker, shift start date and time, shift end date and time, shift type (e.g., regular, standby, call-back and overtime) , and duty type (e.g., roadside service, meeting, training, maintenance and break). The above data are used to generate inputs for the simulation model.



(a) Service Request



(b) Service Vehicles

Figure 3.3: Geographic distributions of service requests and home bases of technicians

The total number of service requests received during the study period is 303, 212, which gives an average of 9, 781 requests per day. Figure 3.3a shows the geographical distribution of service requests received on the 01/05/2014 in the UK. The area that receives the highest density of service requests is London, followed by other urban areas including Manchester, Birmingham, Leeds and Glasgow. Figure 3.3b presents the geographical distribution of home bases of technicians who were on duty on 01/05/2014. It can be seen that the distribution of technicians follows similar patterns to that of service requests.

As stated in Section 3.1.2, tasks are split into eight categories based on their skill requirements. Figure 3.4 shows the percentage of each task category, where Battery, Fuel, Key, JLR, Level 1, and Level 2 form the six types of repair tasks, and Recovery 1 and Recovery 2 are tasks that require recovery services, but the former does not require recovery trucks while the latter requires. More than 70% of tasks are repairable, where the level 2 tasks account for the largest proportion among tasks (42.86%), followed by the level 1 which has a percentage of 25.50%. The total proportion of other repair tasks is less than 7%. In terms of recovery tasks, the percentage of tasks requiring recovery truckers is about half of that of tasks not requiring truckers.

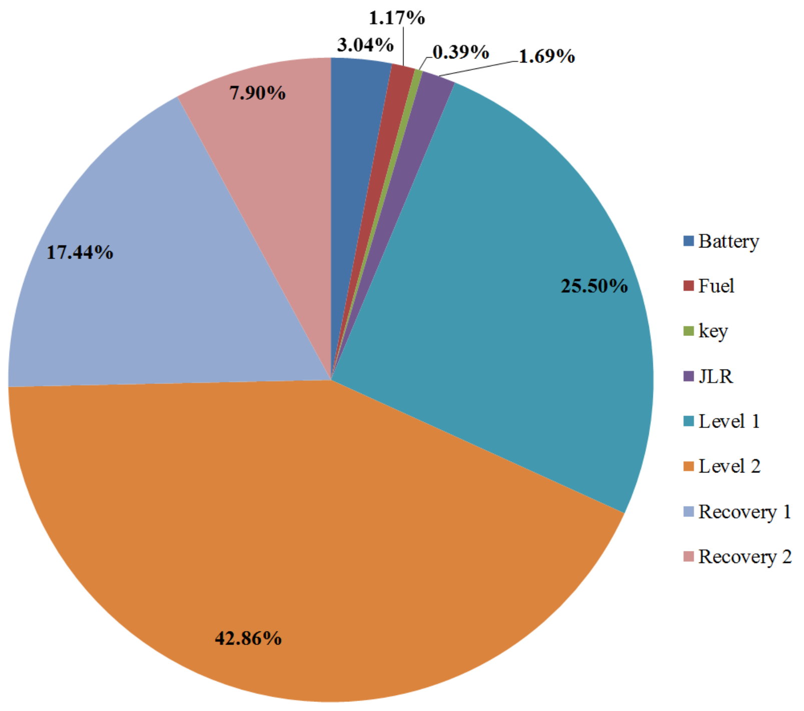


Figure 3.4: Percentage distribution of task categories

The total numbers of technicians and truckers are 2293 and 515, respectively. As described in Section 3.1.1, technicians are grouped into five categories: battery, fuel, key, JLR and general technicians. The percentage of each category is reported in Table 3.1. Among those resources, general technicians have the largest proportion (73.93%), followed by truckers which account for 18.34%. The total proportion of other resources is only about 8%. The percentage distribution of resources follows similar patterns to that of tasks.

Resource Type	Battery	Fuel	Key	JLR	General	Trucker
Count	72	63	16	66	2076	515
Percentage (%)	2.56	2.24	0.57	2.35	73.93	18.34

Table 3.1: Percentage distribution of resource categories

3.3 Distribution of Service Requests

This section describes the process used to identify the arrival distribution of service requests. We first discuss possible seasonality and trends in the time series of service requests, and then define the demand zones. Lastly, we describe how the arrival distribution is constructed for service requests.

3.3.1 Seasonality and Trends of Service Requests

Figure 3.5 presents the average number of service requests by day of the week and by hour of the day. During the study period, there are two bank holidays (public holiday in the UK), 05/05/2014 and 26/05/2014, and therefore we create a separate group, namely "BankHoliday", as shown in Figure 3.5.

The arrival rate, defined as the average number of service requests received per hour, exhibits a significant daily pattern with a strong morning peak appearing between 8:00 and 10:00, a long and off-peak period between 10:00 and 18:00, and low values during the night.

In addition to the daily pattern, there exist a clear weekly pattern in the graph. Monday receives more requests than other days of the week. Particular, the arrival rate during the morning peak of Monday is nearly 1,200, which is about one-third greater than the values during the same time period of other days. This is due to the fact that a large number of people may visit other cities during the weekend and holidays, and they often return to their home cities on the Monday morning, which leads to an increased volume of traffic, and as a result the demand of emergency road services increases. Tuesday seems to receive a slight higher volume of service requests than Wednesday, Thursday and Friday. This can be explained by the two bank holidays mentioned above, where those holidays were on Monday, which makes the following Tuesday present a similar daily pattern to that of the regular Monday. In contrast to weekdays, the arrival rate during weekend and holidays does not present morning and afternoon peaks.

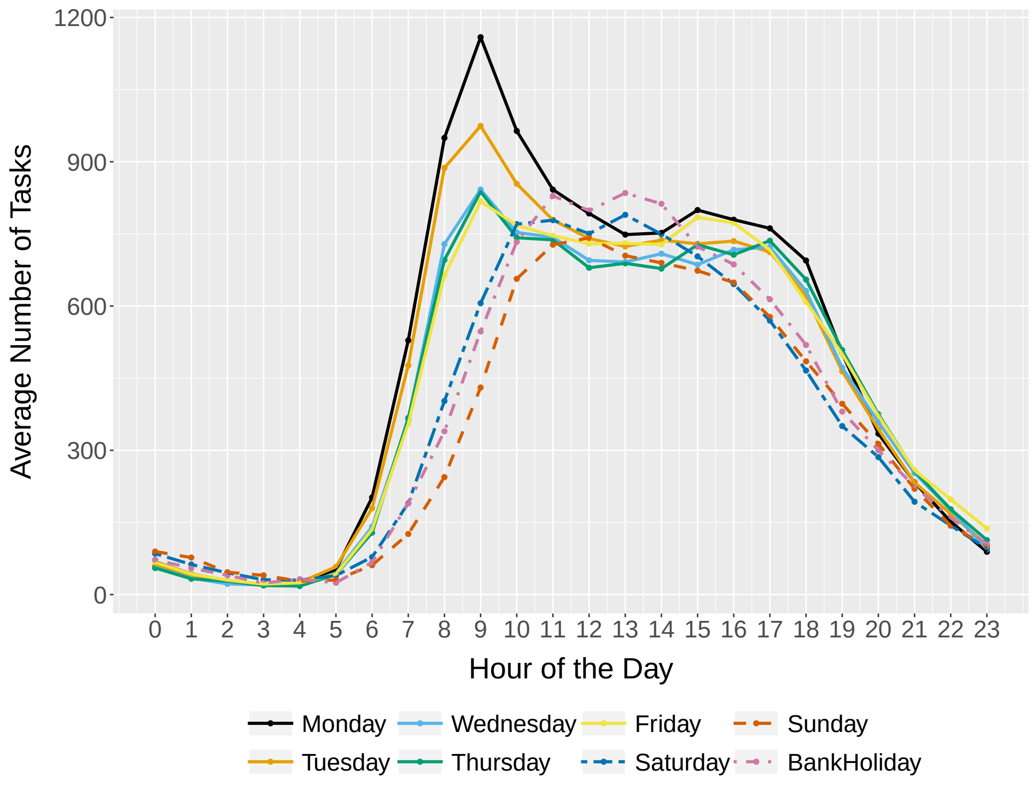


Figure 3.5: The arrival distribution of service requests

3.3.2 Definition of Demand Zones

The geographic distribution of service requests (as shown in Figure 3.3a) indicates that the density of service requests varies significantly over different regions of the UK. Therefore, we split the UK into demand zones.

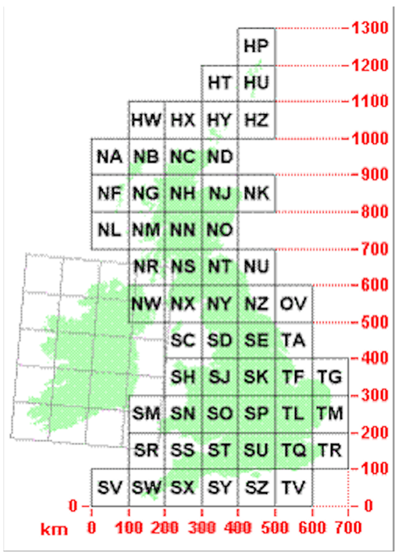


Figure 3.6: Grid Map of the UK

The grip map of Figure 3.6 shows that the UK is split into 56 square areas with a side length of 100km. To provide a more accurate estimation, we further divide each area into 100 smaller squares with a side length of 10km and discard those that do not represent land areas (e.g., ocean), which results in a total of 2723 demand zones. We assume that the daily and weekly patterns are identical over all demand zones. To capture the geographical distribution of service requests, we define a density factor α_j for each demand zone $j \in \{1, 2, ..., 2723\}$, and computed as $\alpha_j = \lambda_j / \bar{\lambda}$, where λ_j is the average number of requests received in zone j per day, and $\bar{\lambda} = \sum_{j=1}^{2723} \lambda_j$ is the sum of the averages of all demand zones.

3.3.3 Identification of Arrival Distribution

The Poisson distribution is commonly used to model the arrival process of queuing systems (Law, 2007). For example, Fujiwara et al. (1987) and Cortés et al. (2011) have used the Poisson process to model the arrival of calls within an ambulance deployment system and a technician dispatching system, respectively. In the following section, we will show that the arrival of service requests considered in our study also follows the Poisson process.

The analysis in Section 3.3.1 reveals that the arrival rate of service requests has strong daily and weekly patterns, and therefore a time-dependent arrival distribution is recommended. The weekly pattern implies the creation of three groups, namely, Weekend, Monday, and TueToFri, where TueToFri represent a group consisting of Tuesday, Wednesday, Thursday and Friday. The group Weekend includes public holidays, such as bank holiday. The group Monday also considers the fist weekday after a public holiday, but excluding those Mondays that are the public holiday. In terms of the hourly pattern, 24 groups are created to reflect the variations of arrival rates throughout the day. Therefore, a total of 72 groups are created and each group has an associated arrival rate of service requests.

To examine whether the arrival of service requests follows a Poisson process, we carry out the following procedure. We first extract a sample group (e.g., TueToFri & 09:00) from the task data, and then count the number of service requests received in each minute over the time period covered by the sample data. The obtained set of count data is used to construct a frequency distribution and calculate the probability of each possible count value by dividing its frequency by the sum of the frequencies. Finally, the obtained probability distribution is compared with the expected probability distribution derived from a Poisson distribution estimated based on the sample data. Figure 3.7 presents an example of the comparison of observed and estimated probability distributions using the sample group “TueToFri & 09:00”. The x-axis gives the possible number of arrivals per unit of time (minute), while the y-axis represents the corresponding probabilities. The histograms show that the Poisson distribution can provide a very good fit to the

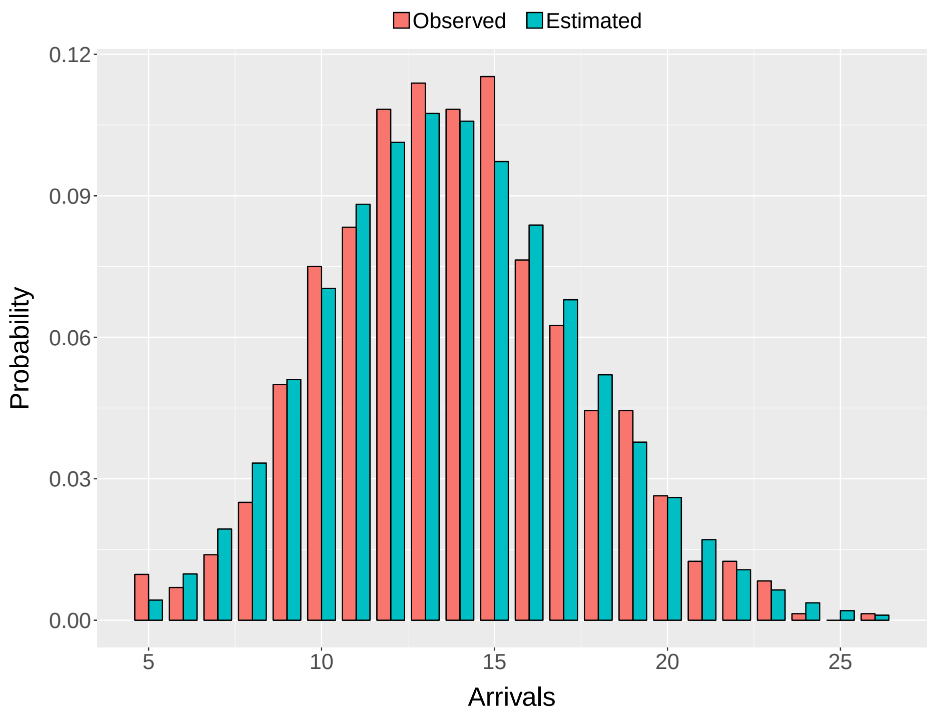


Figure 3.7: Comparison of observed and expected probabilities of each possible number of arrivals per minute

sample data. The same procedure is performed on other groups of data, and similar results have been observed. Therefore, we use a non-homogeneous Poisson distribution to model the arrival of service requests.

Once a random request is generated from the distribution, we need to determine the demand zone from which it comes. One straightforward way is to construct an arrival distribution for each demand zone, and then generate random requests for each demand zone separately. This method is particularly useful for problems with a small number of demand zones (e.g., Fujiwara et al. (1987) and Ingolfsson et al. (2003)). As our problem considers a large-scale region (2723 demand zones), it is neither practical nor efficient to apply the method described above. Therefore, we use a different procedure to determine the demand zone of a random request, described as follows.

1. Random requests are generated using the non-homogeneous Poisson distribution identified;
2. Once a request is generated, it is allocated to a demand zone using the following cumulative probabilities:

$$p_j = \begin{cases} 0 & \text{if } j = 0, \\ \sum_{i=0}^{j} \alpha_i & \text{otherwise.} \end{cases}$$

The selected zone j is determined by the condition $p_{j-1} \leq u < p_j$, where u is a random number drawn from a uniform distribution $U[0, 1]$.

3.4 Service Duration and Recovery Distance

The service duration is defined as the time gap between the starting time and the completion time of a service. For a repair task i carried out by a technician k, the associated repair duration s_i is simply computed as $s_i = c_i - a_i^k$, where c_i represents the service completion time and a_i^k denotes the arrival time of technician k at the location of task i. Technicians are assumed to start services as soon as they arrive at the scenes. For a recovery task j carried out by a technician or trucker k, its service duration s_j consists of two components, preparation duration $\hat{s}_j$ and recovery duration $\bar{s}_j$. The former refers to the amount of time that resource k spends at the scene on diagnosing issues and preparing for the recovery operation (e.g., connecting the disabled vehicle with the recovery system), computed as $\hat{s}_j = b_j - a_i^k$, where b_j represents the departure time of the recovery journey between the scene and the recovery destination. The latter represents the amount of time required for moving the disabled vehicle to the recovery destination, and it is determined based on the required recovery distance $\bar{d}_j$. Therefore, we need to identify three distributions in this section, which are: (1) the distribution of repair durations of repair tasks; (2) the distribution of preparation durations of recovery tasks; (3) the distribution of recovery distance of recovery tasks.

3.4.1 Distribution of Service Durations

Based on the given data and the equations defined above, we extract a sample S of repair durations of repair tasks and a sample $\hat{S}$ of preparation durations of recovery tasks. Since recovery tasks are classified into two categories: recovery 1 and recovery 2 (as shown in Table 3.7), we split $\hat{S}$ into two samples $\hat{S}_1$ and $\hat{S}_2$ for each of the recovery categories.

The box plots (Figure 3.8) show that the obtained samples S, $\hat{S}_1$ and $\hat{S}_2$ contain negative and extremely large values (highlighted in red colour), which are known as outliers. These outliers are identified using the following criterion: an observation point beyond either the lower bound 0 or the upper bound $Q3 + 3 * IQR$ is considered as an outlier, where IQR is the interquartile range which represents the difference between the third quartile $Q3$ and the first quartile $Q1$ of a given sample. Based on the above definition, the upper bounds of samples S, $\hat{S}_1$ and $\hat{S}_2$ are computed as 109, 82 and 49 respectively. For outliers beyond the lower bound, we simply remove them from the samples as they are obvious errors possibly caused by missing or incorrectly entered data. For those outliers having large values, we cannot simply remove them as they may reflect the

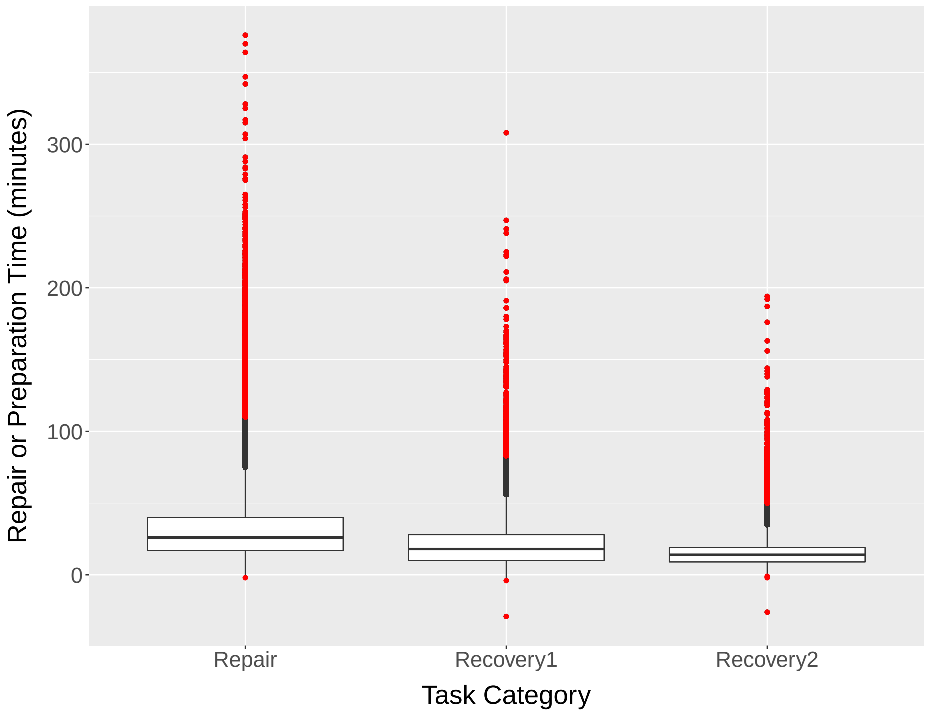


Figure 3.8: Distributions of repair and preparation durations (minutes)

actual variations of the variable considered. However, we need to ensure that unreasonably large outliers are excluded from the sample, as they may cause misleading analysis. After consultation with company experts, the upper bounds of samples S, $\hat{S}_1$ and $\hat{S}_2$ are adjusted to 240, 180 and 120 respectively. We remove the points beyond the corresponding upper bounds from the samples, and summarise the descriptive statistics of the obtained samples in Table 3.2. It can be observed that there exist significant differences between the means and standard deviations (shown as SD. in table 3.2) of samples S, $\hat{S}_1$ and $\hat{S}_2$. This indicates that these samples are drawn from different distributions.

Sample	Min.	$Q1$	Median	$Q3$	Max.	Mean	SD.
S	1.00	17.00	26.00	40.00	239.00	31.88	22.59
$\hat{S}_1$	1.00	10.00	18.00	29.00	178.00	21.87	17.18
$\hat{S}_2$	1.00	10.00	14.00	19.00	119.00	16.12	11.54
Average	1.00	12.33	19.33	29.33	178.67	23.29	17.10

Table 3.2: Descriptive statistics of repair and preparation durations (minutes)

As the data of sample S is continuous and positive, the potential distributions that may fit to S are exponential, gamma, log-normal, normal and Weibull distributions. Thus, we first fit these distributions to our sample S using the maximum likelihood method. Then we compare the empirical cumulative distribution function (CDF) of S with the theoretical CDFs of the parametric distributions obtained previously. As illustrated in Figure 3.9, the black line shows the empirical CDF while the coloured lines present the theoretical CDFs. We can see that the exponential and normal distributions do not provide a good fit to S, because the shapes of the corresponding lines are different to that of the empirical CDF and there exist significant gaps between these lines. Although the other three lines are quite close to the empirical CDF, the log-normal distribution appears to fit S best than the gamma and Weibull distributions.

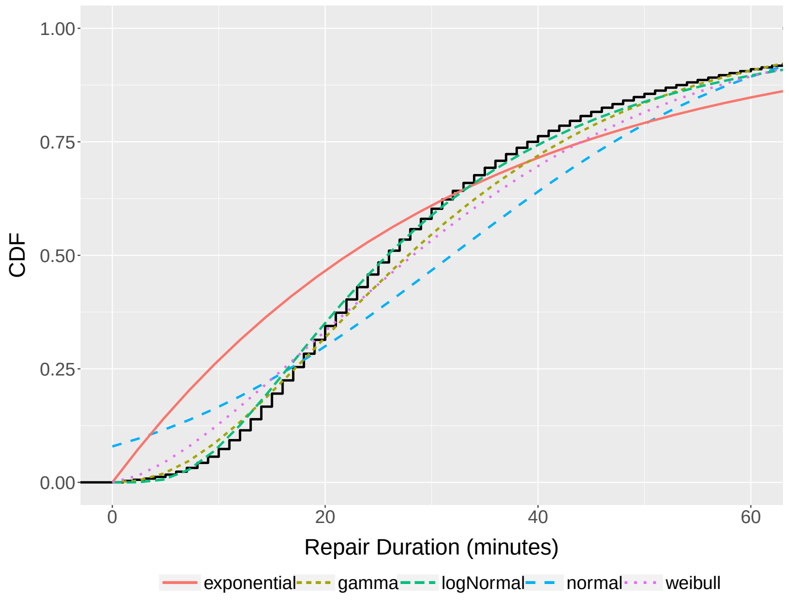


Figure 3.9: Empirical CDF of sample S against theoretical CDFs of estimated distributions

Figure 3.10 presents the probability density function (PDF) of sample S (as shown by blue bars), fitted with a log-normal distribution, of which the mean and standard deviation of the variable's natural logarithm are estimated to 3.25 and 0.67 respectively. The obtained log-normal distribution presents a very good fit to sample S, and thus we use this distribution to model the repair duration.

The same procedure used above is carried out on the samples $\hat{S}_1$ and $\hat{S}_2$. The results show that $\hat{S}_1$ can be fitted to an gamma distribution with shape 1.73 and rate 0.08, and $\hat{S}_2$ can be fitted to a log-normal distribution with parameters estimated to be 2.58 and 0.65. The details of fitting distributions on these two samples can be seen in Appendix A.1.

3.4.2 Initial Service Duration

The above distributions are used to model the actual repair or preparation duration required by a task. The information about the actual service duration is only available after the technician or trucker arrives at the scene and carries out vehicle diagnostics. When making the scheduling and routing decisions, we only know the initial service duration which is estimated based on the information collected from the customer. According to the historical data, the value of initial service duration is selected from the vector $\{15, 20, 25, 30, 35, 40, 45\}$. To investigate how the value is determined, we perform

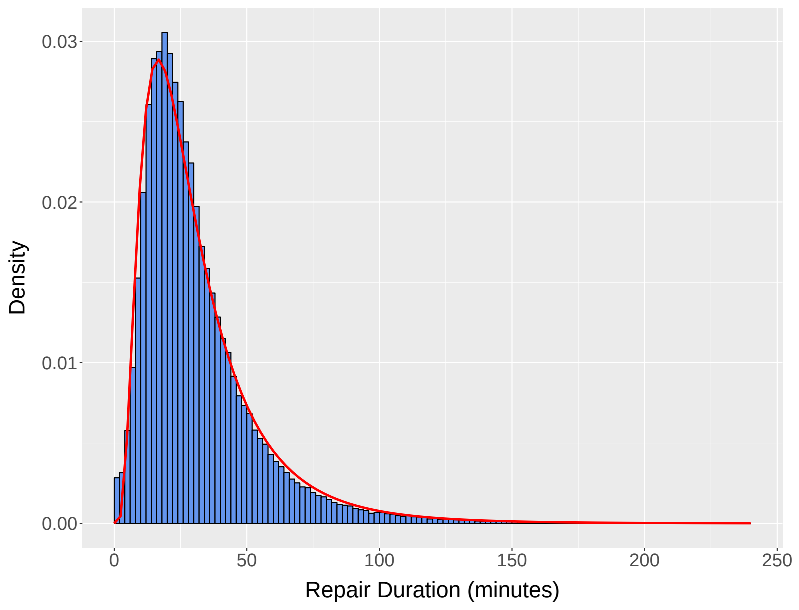


Figure 3.10: PDF of sample S with a log-normal distribution fitted

analysis using three samples of initial service durations, corresponding to repair tasks, recovery 1 and recovery 2 tasks, respectively.

Time	Repair		Recovery 1		Recovery 2		All Tasks	
(minutes)	Freq.	%	Freq.	%	Freq.	%t	Freq.	%
15	2624	13.82	140	2.89	35	1.79	2799	10.85
20	14259	75.12	3784	78.04	1423	72.75	19466	75.49
25	614	3.23	315	6.50	134	6.85	1063	4.12
30	1119	5.90	443	9.14	269	13.75	1831	7.10
35	50	0.26	28	0.58	18	0.92	96	0.37
40	215	1.13	113	2.33	62	3.17	390	1.51
45	100	0.53	26	0.54	15	0.77	141	0.55
Total	18981	100.00	4849	100.00	1956	100.00	25786	100.00

Table 3.3: Percentage distribution of initial repair durations (minutes)

Table 3.3 shows that all samples have similar percentage distributions on the values of initial service duration, where about 75% of tasks having their initial service durations equal to 20 minutes. The scatter plot (Figure 3.11) indicates that there is no clear relationship between the initial service durations and the actual service durations. In other words, the initial service duration does not provide a good estimation for the actual time required. Based on the above results, we simply use 20 minutes as the initial service duration of each random task generated by the simulation model.

3.4.3 Distribution of Recovery Distance

For a recovery task i, let (x_i, y_i) be the coordinates of the task location and $(\bar{x}_i, \bar{y}_i)$ be the coordinates of the associated recovery destination. Then the recovery distance $\bar{d}_i$ is

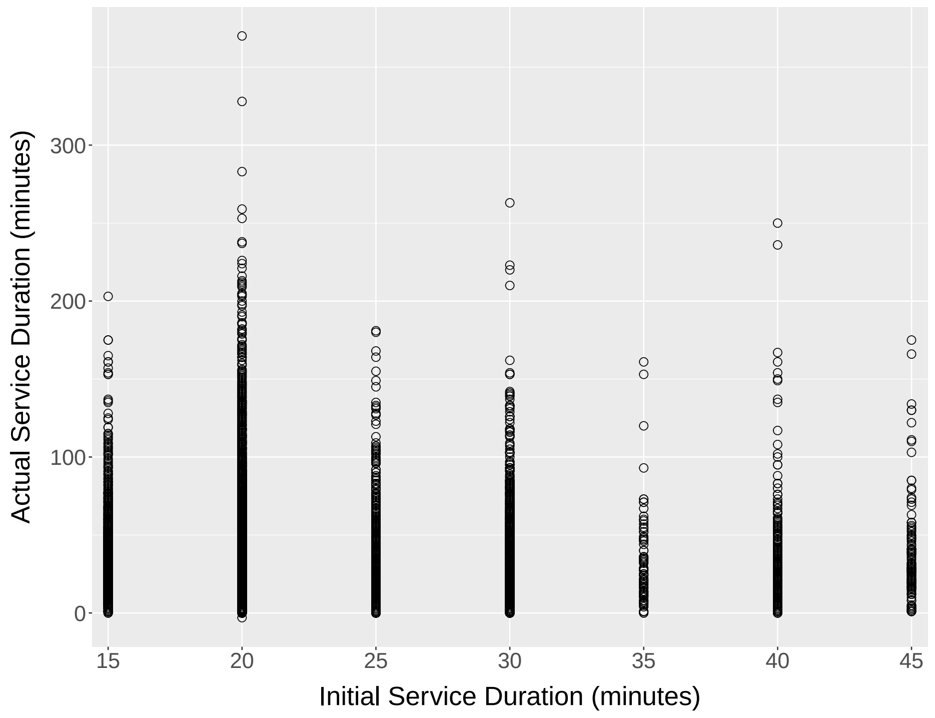


Figure 3.11: Scatter plot of initial vs. actual service durations

defined as the Euclidean distance between the task location and the recovery destination, and computed as $\bar{d}_i = \sqrt{(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2}$. Using this equation, we obtain a sample $\bar{D}$ of recovery distances from the historical data. We split $\bar{D}$ into two samples $\bar{D}_1$ and $\bar{D}_2$ to represent recovery 1 and recovery 2 tasks, respectively.

The distributions of samples $\bar{D}_1$ and $\bar{D}_2$ are displayed in Figure 3.12, where the red points represent outliers beyond the upper bounds 54.19 and 180.65 respectively, computed using $Q3 + 3 \times IQR$. Both $\bar{D}_1$ and $\bar{D}_2$ contain a considerable amount of outliers. We need to adjust these outliers to reasonable values as they may affect the results of analysis. The category Recovery1 refers to tasks that require basic recovery services which are usually performed by patrols. Patrols are subject to home constraints, which enforce that the maximum service distance is 60km from the home bases. For tasks requiring long recovery distances (more than 60km), they can only be carried out by trucks, and thus are classified into the category Recovery 2. For this reason, we adjust the upper bound of $\bar{D}_1$ to 60km. In contrast to patrols, trucks are not subject to home constraints, and so they can carry out tasks requiring long recovery distances. However, they must return to the depots at the end of their shifts. If an extremely long recovery distance (e.g., 400km) is required by a customer, the request is usually assigned to more than one trucks, each performing part of the recovery journey. In this situation, there are actually more than one tasks created for this customer, each assigned to a different truck. Thus, it is very unlikely to have a task requiring very long recovery distance. Sample $\bar{D}_2$ contains ten data points that are greater than 300km. Comparing to the

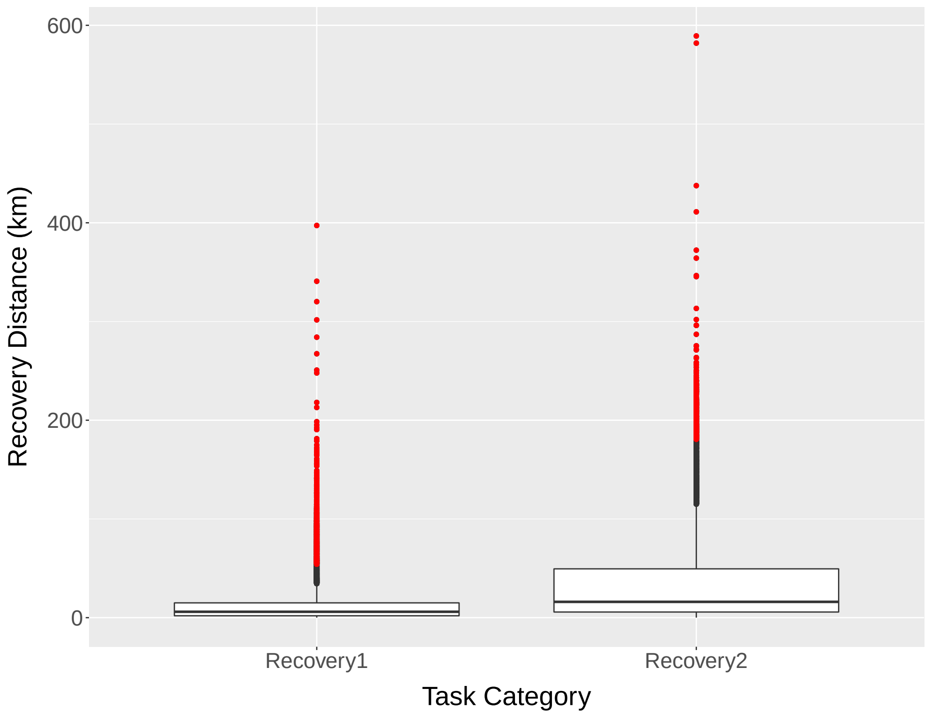


Figure 3.12: Distributions of recovery distances (km)

size of $\bar{D}_2$, which is 15561, those large outliers are negligible, and so we simply remove them from $\bar{D}_2$.

Table 3.4 presents the descriptive statistics of $\bar{D}_1$ and $\bar{D}_2$ after addressing the outliers identified above. The mean and standard deviation of $\bar{D}_2$ are significantly higher than those of $\bar{D}_1$, which implies that these two samples are drawn from different distributions. Because $\bar{D}_1$ is subject to a strict bound (recovery distance must be not greater than 60km), we construct an empirical distribution to fit $\bar{D}_1$. Figure 3.13 shows the probability distribution of the empirical distribution obtained, where the interval $(0, 60]$ is equally split into 20 bins, each having a width of 3, and each blue bar represents the probability that a number falls into the associated bin.

Sample	Min.	$Q1$	Median	$Q3$	Max.	Mean	SD.
$\bar{D}_1$	0.01	1.78	5.52	13.16	59.98	9.87	11.71
$\bar{D}_2$	0.02	5.71	15.98	49.40	296.10	36.62	46.70
Average	0.02	3.74	10.75	31.28	178.04	23.24	29.20

Table 3.4: Descriptive statistics of samples $\bar{D}_1$ and $\bar{D}_2$

To find the most appropriate distribution for $\bar{D}_2$, we carry out the same fitting procedure used previously for service durations. The results show that there is no suitable parametric distribution to fit $\bar{D}_2$, and therefore we choose to use the empirical distribution. The details of the fitting procedure for $\bar{D}_2$ can be seen in Appendix A.2.

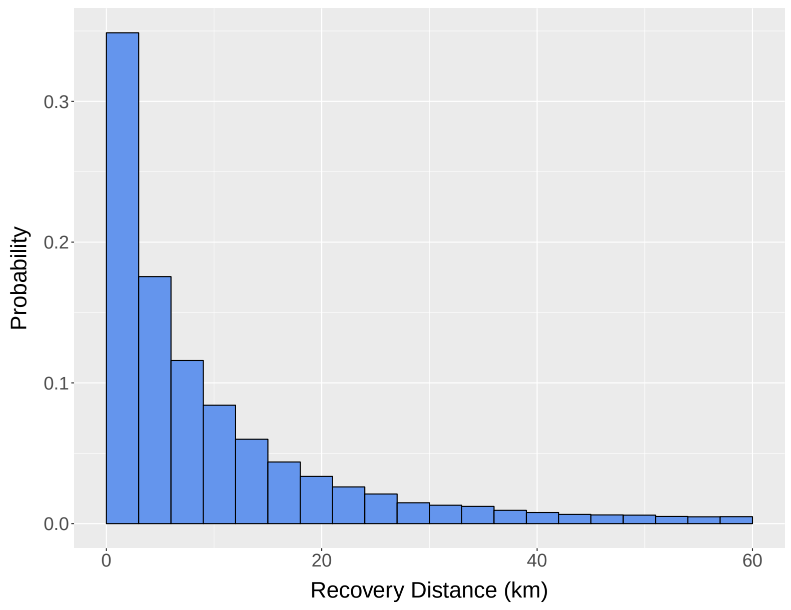


Figure 3.13: Probability of the empirical distribution fitted on sample $\bar{D}_1$

3.5 Development of Simulation Model

This section discusses the development of a simulation model to represent the real system used by the AA company. The model is developed using the programming language C++ rather than bespoke simulation software. Although the use of programming language requires extensive efforts and time, it provides better flexibility in model design and allows us to implement complicated strategies and dispatching algorithms. An overview of the simulation model is initially presented, followed by a discussion of simplifications and assumptions that have been made. Then we describe how the travel distance and time are computed. Next the model inputs and outputs as well as dispatching policies are illustrated. After that the model verification and validation are described. Finally, the user interface developed for the simulation model is presented.

3.5.1 Overview of Simulation Model

The developed simulation model is event-based, which is also known as discrete event simulation (DES). In a DES model, the state of the system is only changed at discrete points in time at which events occur, and thus the simulation can directly jump in time from one event to the next. According to the three-phase approach described by Tocher (1963), the simulation events can be classified into two categories: B (bound or booked) events and C (conditional) events. The former refers to the events that are scheduled to occur at a point in time, while the latter represents the events that are dependent on the conditions of the model. Based on the real system described in Section 3.1, a number of

Event	Type	Change in state	Future event
B1	Arrival	New request arrives and enters the decision queue.	B1
B2	Update status	Shift starts and resource logs into the system. Technician departs from the home base, and trucker departs from the depot.	B2
B3	Update status	Shift ending time is reached and resource cannot accept new tasks.	B3
B4	Update status	Break requirement is triggered. Set the resource status to 'break required'.	B4
B5	Update status	Resource finishes the break. Set the resource status to available.	
B6	Arrival	Resource arrives at task location. Set velocity to 0.	
B7	Arrival	Technician arrives at the home base. Set velocity to 0 and status to idle.	
B8	Arrival	Trucker arrives at the depot. Set velocity to 0 and status to 0.	
B9	Finish activity	Resource completes recover preparation and starts the recovery journey. Compute velocity and set status to travelling.	B11
B10	Finish activity	Technician completes repair service and removes task from the job list. Set status to idle.	
B11	Finish activity	Resource arrives at the recovery destination, completes recovery service and removes task from the job list. Set the status to idle.	
B12	Finish activity	Request is outsourced to garage and service is completed.	

Table 3.5: List of B events

B events and C events (as illustrated in Table 3.5 and 3.6) have been identified to drive the simulation model.

Figure 3.14 illustrates the framework of our simulation model, based on the three-phase approach proposed by Tocher (1963). The simulation starts with an initialisation step, which initialise input data, states and events. Then the A-phase is executed, which identifies the time of the next event and advance the simulation clock to that time. The elapsed time is also computed and the locations of resources with status of travelling are updated. In the B-phase, all B-events that are due at the current simulation clock are executed, while in the C-phase, all C-events are attempted and those for which their conditions are satisfied are executed. If no further C events can be executed, the simulation returns to the A-phase to perform another iteration of the three-phase procedure until the termination criterion is met.

Event	Condition	Change in state	Future event
C1	Decision queue is not empty	Outsourcing, scheduling and dispatching decisions are made.	B12
C2	(1) Resource type is technician; (2) Current time is less than 45 minutes to the end of the shift; (3) The job list is empty.	Technician starts returning to the home base. Compute velocity and set status to travelling.	B7
C3	(1) Resource type is trucker; (2) Trucker is away from the depot; (3) The job list is empty.	Trucker starts returning to the depot. Compute velocity and set status to travelling.	B8
C4	(1) Break requirement is triggered; (2) resource is not serving customer.	Resource starts taking the break. Set resource status to 'on break'.	B5
C5	(1) Resource is idle; (2) the job list is not empty.	Resource starts travelling to the location of the first task in the list. Compute velocity and set status to travelling.	B6
C6	(1) Technician arrives at the task location; (2) Repair task.	Technician starts repair service. Set technician status to serving	B10
C7	(1) Resource arrives at the task location; (2) Task requires recovery operation.	Resource starts the recovery preparation and set the status to serving.	B9

Table 3.6: List of C events

3.5.2 Simplifications and Assumptions

According to Robinson (2004), a simulation model must be designed as simple as possible to meet objectives of the study. A well simplified model speeds up the development process and increases the utility of the model. The main purpose of model simplification is to reduce the complexity of a model while maintaining an acceptable level of its validity or credibility. Model simplification can be achieved by removing components and interconnections that have a small influence on the model accuracy; or by using a simpler representation for complex components and interconnections while ensuring the model accuracy at certain level (Robinson, 2004). In developing the simulation model for the real system used by the company, the following simplifications are made:

1. The percentage distribution of resource categories (Table 3.1) indicates that battery, fuel, Key and JLR technicians only account for a very small proportion among

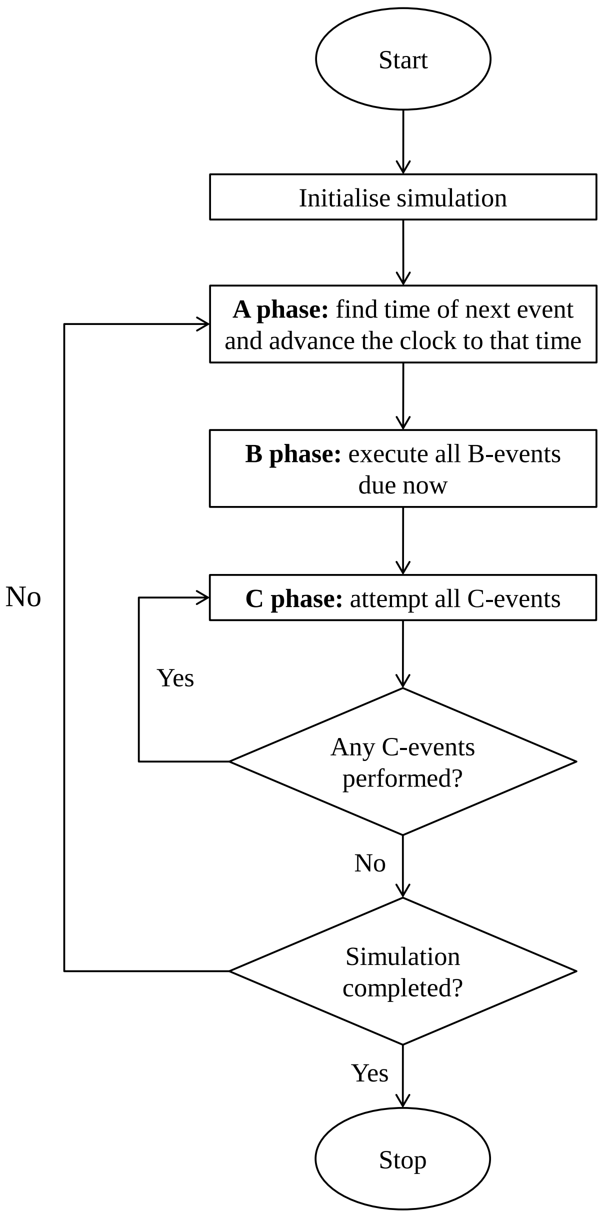

Start
Initialise simulation
A phase: find time of next event and advance the clock to that time
B phase: execute all B-events due now
C phase: attempt all C-events
Any C-events performed?
Yes
No
Simulation completed?
No
Yes
Stop

Figure 3.14: The three-phase simulation framework (Tocher, 1963)

all resources (7.72%), and hence we merge them into the group of general technicians. Therefore, the AA resources are simplified into two categories: technician and trucker.

2. With the simplification on the categorisation of resources, task categories are naturally simplified into: repair, recovery 1 and recovery 2. The first category refers to repair tasks, including battery, fuel, key, JLR, repair 1 and repair 2 tasks. The percentage distribution of the simplified task categories is presented in Table 3.7.

3. If a request is sent to the garage, the request is assumed to be completed and the simulation model does not track the service provided by the garage.

Category	Description	Probability
Repair	Tasks require technicians to carry out roadside repairing services	0.75
Recovery 1	Tasks require either technicians or truckers to provide recovery services	0.17
Recovery 2	Tasks require truckers to perform recovery services	0.08

Table 3.7: Distribution of task categories

Unlike simplifications, assumptions integrate uncertainties and beliefs about the real world into the model (Robinson, 2004). The proposed simulation model is built based on the following assumptions:

1. Service requests are independent of each other.
2. Driving speed does not vary over time and vehicle are travelling using the same speed function.
3. The skill requirements are assumed to be known at the time of receiving service requests.
4. All requests must be responded to, either by an AA resource, or by a garage, with no cancellation allowed.
5. All requests are assumed to require immediate services, and no appointment is allowed.
6. Each request can be assigned to at most one service vehicle. Once a vehicle starts travelling to the location of a task, it has to complete this task.

3.5.3 Calculation of Travel Distance

Task locations and vehicle positions are represented by points in a Euclidean plane, and the distance between two points is defined by the Euclidean distance. Therefore, the travel route from a vehicle position to a task location is estimated by the straight-line path between them. However, this method may not always be applicable, since the actual road network generally contains physical barriers such as major rivers, mountains and irregular coastlines. To overcome this issue, we introduce the via points, where if the straight-line path between a vehicle and a task crosses a barrier, it has to go via the specified point.

Figure 3.15 shows that a vehicle (the blue car icon) is required to visit a request (the red phone icon), which is located on the other side of the port, represented by the red line. The vehicle has to visit the via point (the red pin) first, and then travel from the

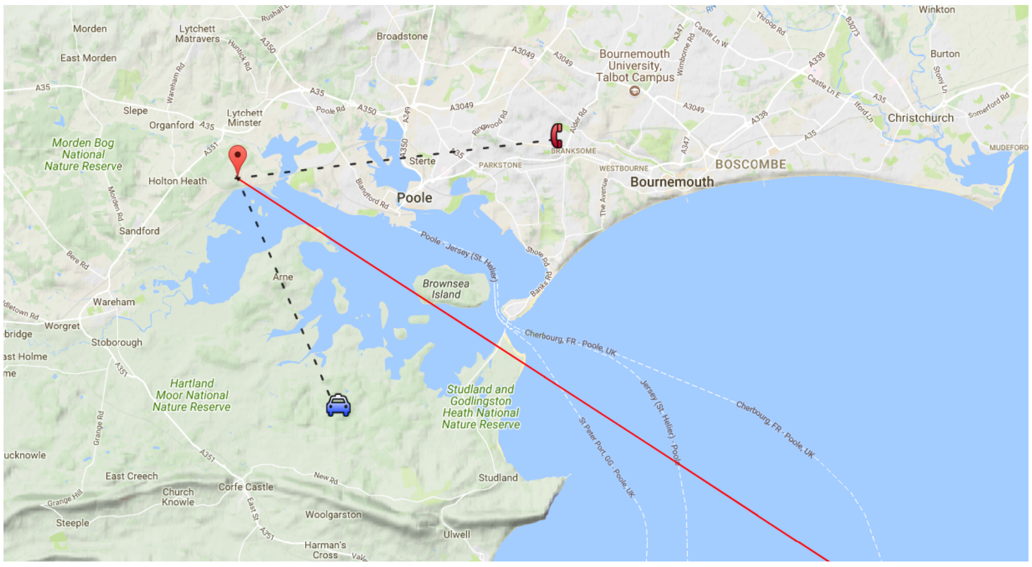


Figure 3.15: Illustration of computing travel distance

via point to the request location. Therefore, the distance between the vehicle and the customer is estimated as the total distance of two straight-line paths, which are shown as the dashed lines. The full list of barriers and via points of the UK is provided by the company, and it is taken as input by the simulation model when estimating travel distances.

3.5.4 Calculation of Travel Time

The driving speed is computed using a speed function that takes the travel distance as input and outputs the driving speed. The speed function actually estimates the non-linear relationship between the driving distance and speed, where longer distances lead to higher driving speeds (as illustrated in Figure 3.16).

In addition to the speed function, a time function that takes the travel distance and driving speed as input is used to compute the travel time required. The time function applies adjustment factors to account for the circuitous nature of the actual road network. The same speed and time functions are also used in the real system. For confidentiality reasons, the details of these functions are not presented here.

3.5.5 Model Inputs and Outputs

The inputs of the simulation model include random task instances, staff rota, barriers and via points. Task instances are generated by an instance generator, which takes the distributions identified previously as input and produces random task samples, where each task i has the following categories: arrival time t_i, coordinates (x_i, y_i) of the request

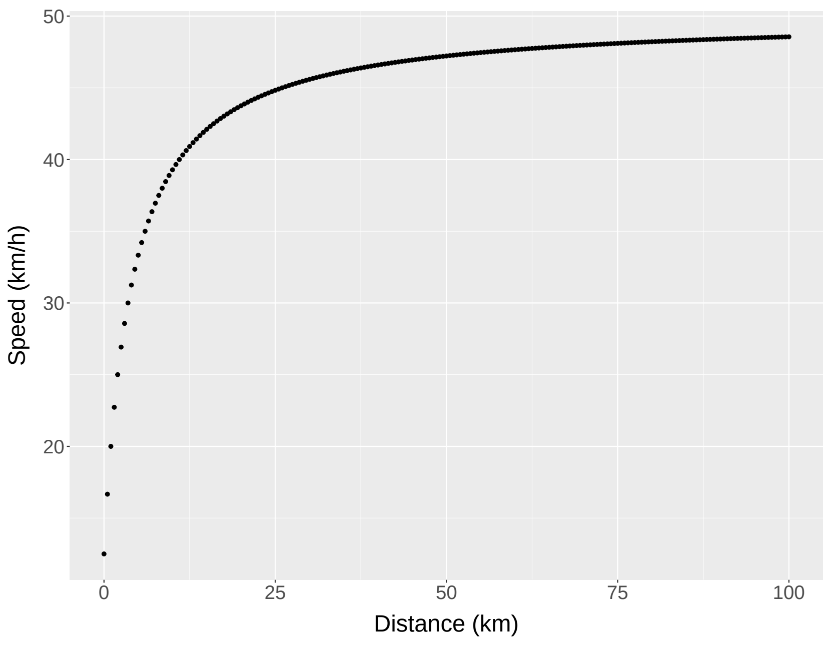


Figure 3.16: Relationship between driving distance and speed

location, roadside service duration s_i, skill requirement q_i, and coordinates $(\bar{x}_i, \bar{y}_i)$ of the recovery destination if recovery service is required.

Figure 3.17 presents the procedure of generating a random request. First, we generate a request i following the Poisson process, where the arrival time t_i is the sum of inter-arrival times generated by the corresponding exponential distribution. Since the arrival rate of our Poisson process is time-dependent, we apply the thinning approach (Lewis and Shedler, 1979) in order to ensure that the number of arrivals generated in each hour is close to the rate derived from the historical data. The idea behind the thinning approach is to generate a Poisson arrival process at the maximum rate λ^*, but accept each arrival with probability $p_j = \lambda_j/\lambda^*$, where λ_j is the arrival rate for a given time interval j. After a request i is generated and accepted, we allocate it to a demand zone using the method described in Section 3.3.3. The actual location (x_i, y_i) within the demand zone is determined according to uniform distributions along the X and Y axes. Then the skill requirement q_i is determined using the probability distribution of task categories given in Table 3.7. Based on the skill requirement, the roadside service duration s_i is determined using the corresponding duration distribution. If the request requires recovery service, we use the corresponding distance distribution to determine the recovery distance $\bar{d}_i$. Given the distance $\bar{d}_i$, the recovery destination $(\bar{x}_i, \bar{y}_i)$ is a point on a circle with the centre at (x_i, y_i) and a radius of $\bar{d}_i$. The value of $\bar{x}_i$ is determined by a uniform distribution over the interval $[x_i - \bar{d}_i, x_i + \bar{d}_i]$, and then $\bar{y}_i$ is randomly selected between two values $y_i - \sqrt{d_r^2 - (x_i - \bar{x}_i)^2}$ and $y_i + \sqrt{d_r^2 - (x_i - \bar{x}_i)^2}$ with equal probability. After performing the above steps, the attributes of a random request are determined. To generate an instance containing requests that occur over a predefined

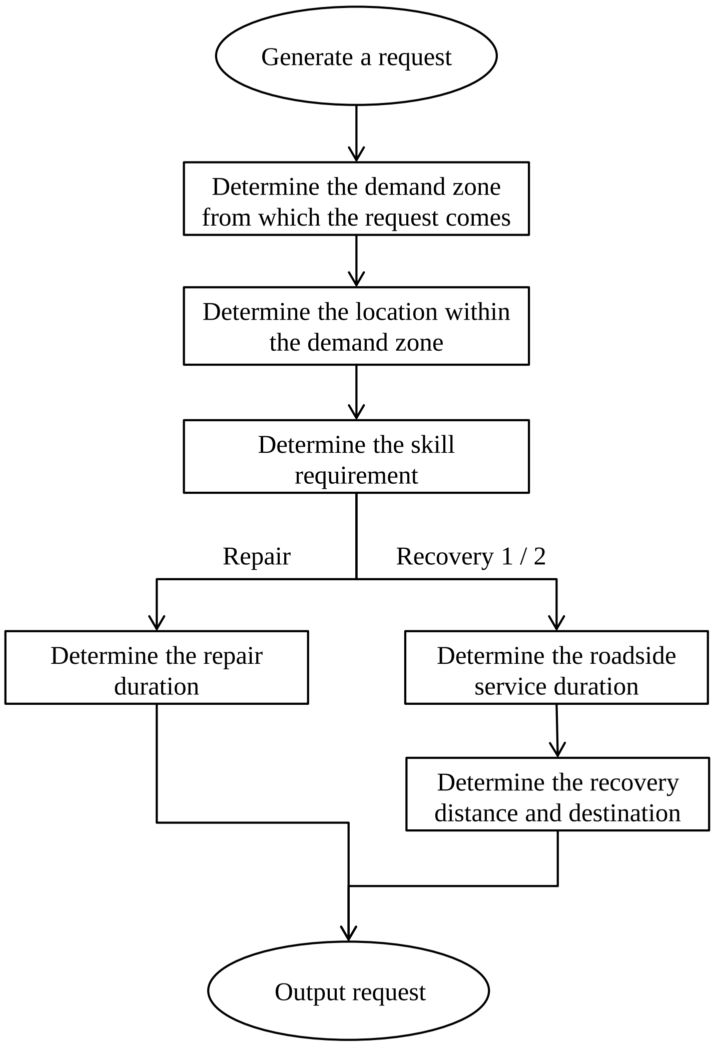

Generate a request
Determine the demand zone from which the request comes
Determine the location within the demand zone
Determine the skill requirement
Repair
Recovery 1 / 2
Determine the repair duration
Determine the roadside service duration
Determine the recovery distance and destination
Output request

Figure 3.17: Flow chart for generating a random request

period of time, the above procedure is repeatedly executed until the request arrival time reaches the time limit.

The staff rota includes the following categories: shift start time, shift end time, resource type (technician or trucker), coordinates of the technician home base and the truck depot. We choose to use the real rota data provided by the company, because the design of staff rota is very complicated and time consuming. Moreover, the use of real staff rota ensures that the simulation accurately reflects the real system.

The outputs of the simulation model are the average response time, the total number of rejected (outsourced) requests and the total driving distances. The average response time and the total driving distance are computed for requests responded by the AA resources, since the simulation model does not track the services provided by the garage.

3.5.6 Dispatching Policies

The dispatching policy determines the acceptance/rejection decisions as well as the scheduling and routing decisions for every request arriving at the system. We first consider a quickest response (QR) policy, which reflects the dispatching policy used by the company. The QR policy is based on a greedy algorithm, of which the pseudo-code is illustrated below.

Algorithm 1 The quickest response (QR) policy

```
1: Input: new request $i$, vehicle set $K$, upper limit of response time $T^{Max}$, upper
   limit of service distance $D^{Max}$, and a sufficiently large constant $M$
2: Output: the minimum expected response time $t^*$ and the corresponding vehi-
   cle $k^*$
3: set $t^* = M$
4: for each vehicle $k$ in $K$ do
5:     if SkillCompatibility(i, k) = true then
6:         if Distance($x_i, y_i, x_k, y_k$) $< D^{Max}$ then
7:             Insert $i$ at the end of the job queue of $k$
8:             Compute the expected response time $t$
9:             if $t < t^*$ then
10:                $t^* \leftarrow t$
11:                $k^* \leftarrow k$
12:            end if
13:        end if
14:    end if
15: end for
16: if $t^* >= T^{Max}$ then
17:     $k^* \leftarrow$ Garage
18: end if
```

The inputs of the QR policy are the new request i with coordinates (x_i, y_i) , a set of active vehicles K, the maximum response time allowed T^{Max}, the maximum service distance allowed D^{Max} and a sufficiently large constant M that is used to initialise the variable t^*. The current location of each vehicle $k \in K$ is given by (x_k, y_y). The lines 4 to 15 present the main loop of the *QR* policy. For each vehicle k in K, we check the skill constraint and the home constraint between i and k. If both constraints are satisfied, we insert i at the end of the job queue of the vehicle k and compute the expected response time t for request i. If the value of t is strictly less than the current best response time t^*, then t^* and v^* are replaced by t and v respectively. The main loop terminates until all vehicles in K are examined. If the minimal response time t^* is greater or equal to the time limit T^{Max}, request i is rejected to the garage; otherwise, the function outputs the best vehicle v^* selected.

In addition, we consider a shortest path (SP) policy, which is developed based on the same framework of the QR policy, but it selects the one producing shorter driving distance when comparing two candidate vehicles.

3.5.7 Model Verification and Validation

One of the most important step during the development of a simulation model is the model verification and validation. Specifically, verification refers the process of confirming that the conceptual model has been correctly transformed into a computer model, while validation is process of ensuring that the simulation model provides a sufficiently accurate representation of the real system (Robinson, 2004). The following section describes the methods that have been applied to verify and validate our simulation model.

Model verification:

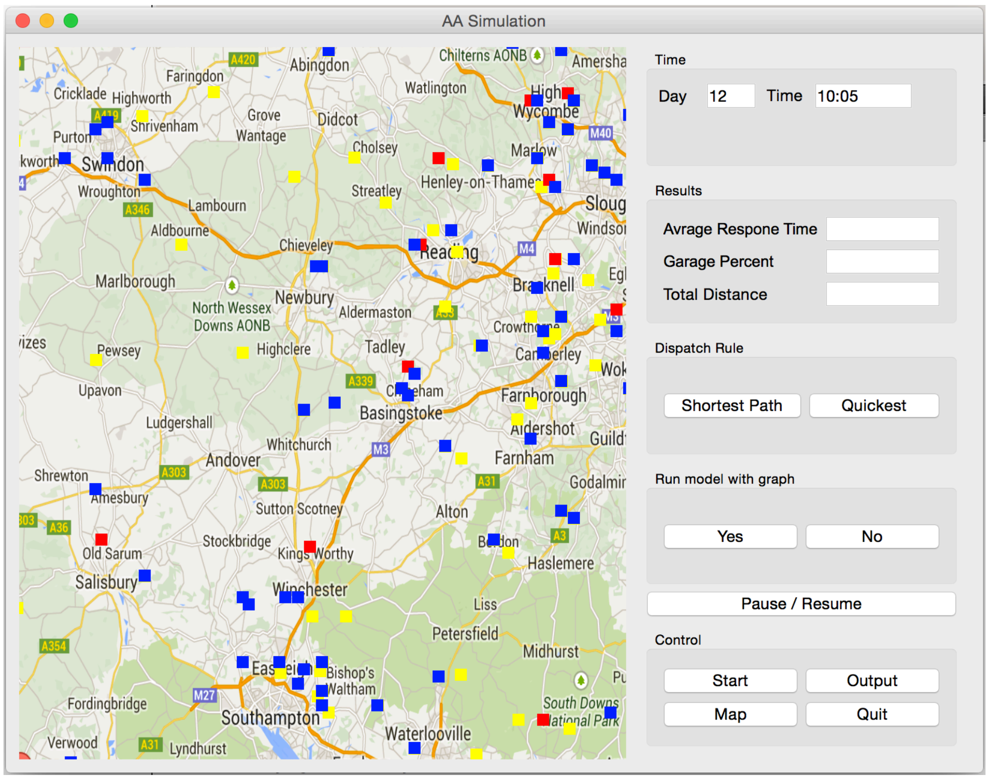
1. To ensure that the implementation of the instance generator is correct, the generated task samples are verified to ensure that they follow the input distributions;
2. To ensure that the simulation model performs correctly, small instances have been used to test the model and the outputs are carefully checked to guarantee that the service flow of each vehicle is logically correct and calculations of time information and driving distance are correct;
3. To ensure that the dispatching policies work as expected, small instances have been used to test dispatching policies and the results are verified with those obtained from a spreadsheet implementation of the same algorithm;
4. Visual checks have been conducted by linking the simulation with Google map and watching how each element of the model behaves.

Model validation:

1. Prior to developing the model, extensive consultations have been held with the company experts who have excellent knowledge of the real system in order to give us a clear understanding of the problem;
2. Once the model is set up, it is discussed with the company experts to ensure that the model accurately reflects the operation of the real system;
3. The simulation model is compared with the real system using the correlated inspection approach, where the simulation model is driven using historical data and the model outputs are compared with those obtained from the real system. In such scenario, the input data is also known as traces, where a trace represents a stream of data that describes a sequence of events collected from the real system (Robinson, 2004). Applying traces is particularly useful for validating the model, since it allows the modeller to examine whether the behaviour of the simulation is close to that of the real system when the same event occurs. To compare the outputs, we compute the mean values and standard deviations of the response times obtained from the simulation model and the real system respectively. We also apply a pairwise comparison method to examine the mean differences and the associated confidence intervals. The detailed results are presented in Section 3.6.

3.5.8 User Interface

To enable users to interact with the model and have a visual view of the simulation, we have developed a user interface, which is presented in Figure 3.18. The developed user interface consists of two main components: view window and control panel. The former gives the visualisation of the real-time emergency vehicle dispatching and routing, where the yellow, blue and red points represent the service vehicles, the locked tasks and the waiting tasks respectively. The control panel include a time window to display the current simulation time and a result window to display the average response time, the garage percentage and the total driving distance of service vehicles at the end of a simulation run. The buttons names Shortest Path and Quickest specify the dispatch policy applied by the simulation model, and they correspond to the SP and QR policies respectively. The rest of buttons allow users to configure the model, such as whether or not to display the visualization, pause/resume during a simulation run and export the detailed results to an external file. In addition, the simulation model is linked with an external map provided by the Google. By clicking the button titled Map on the control panel of the user interface, users can view the current locations of service vehicles and requests, and the scheduled vehicle routes on the real map.


AA Simulation

Time

Day 12 Time 10:05

Results

Avrage Respone Time

Garage Percent

Total Distance

Dispatch Rule

Shortest Path Quickest

Run model with graph

Yes No

Pause / Resume

Control

Start Output

Map Quit

Figure 3.18: The user interface of the simulation model

3.6 Simulation Results

This section presents the results of simulation experiments. We first discuss the comparison of the simulation model and the real system. Then we assess the performance of the simulation model under different demand levels. Next we examine the effect of using different values of time limit T^{Max}. Lastly, we compare the performance of two dispatching policies described previously, which are the QR and SP policies.

3.6.1 Simulation Setup

The developed model is a non-terminating simulation, since the real system operates 24 hours a day and 7 days a week. For non-terminating simulations the output must only be collected when the model reaches the steady state, otherwise it causes the initialisation bias. At the start of our simulation, all job queues of vehicles are empty and there is no work in progress. This is obviously not the realistic condition. In order to ensure that the simulation output is accurate, we apply a warm-up period to run the model until it reaches the steady state and only collect results after this point. To determine the length of the warm-up period, we apply the time-series inspection method (Robinson, 2004) to the simulation output. As recommended by (Robinson, 2004), at least five replications should be performed, since the output data of a single run can be noisy and so it is difficult to identify the initialisation bias through the time-series. For this reason, we perform ten replications, each covering a duration four weeks (28 days) with each week starting on Monday and ending on Sunday. In addition, the simulation model uses random staff shift data derived from the given historical data. The mean averages of the output data of the ten replications are recorded hourly over the four weeks' period and the results are illustrated as follows.

Figure 3.19 displays the time-series of the average response time. The plot presents that there are strong fluctuations during the first week. Then the data appear to be steady state with no significant upward or downward trend. This indicates that the simulation requires one week to reach the steady state. The time-series of other performance indicators including garage percentage and the average driving distance can be seen in Appendix A.3. Those time-series also present similar patterns to the one described above. Therefore, we set the warm-up period to one week.

In addition to the initialisation bias, we need to ensure that enough output data is collected from the simulation in order to delivery accurate estimation of the model performance. This can be achieved by selecting an appropriate run-length of the simulation and performing multiple replications. Accounting to Law (2007), the run-length of a non-terminating simulation must be much longer than the warm-up period, otherwise there may still exist some bias in the model output. Banks et al. (2005) suggest that the run-length is at least ten times the length of the warm-up period. We follow this

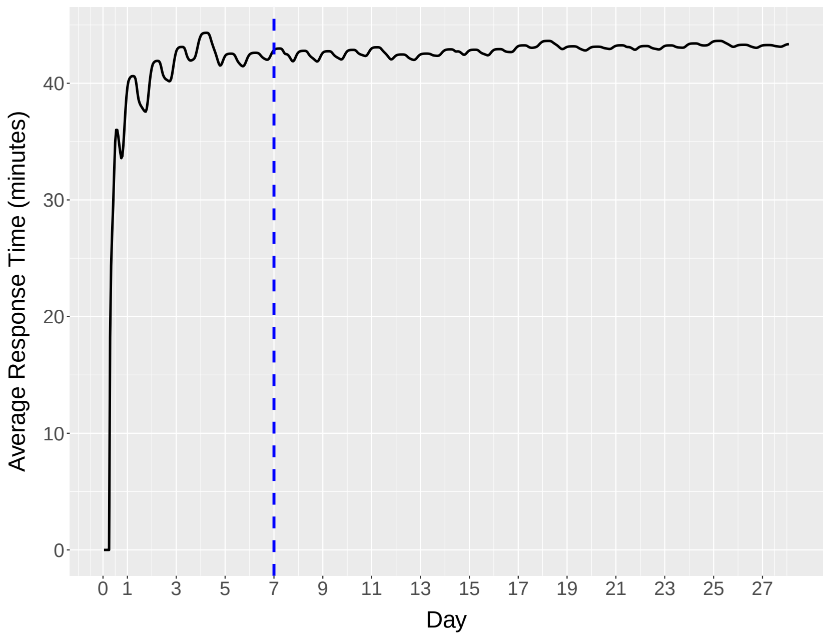


Figure 3.19: Time-series of average response time from ten replications

recommendation, and set the duration of a simulation run to ten weeks (70 days). To determine the number of replications required, we examine the changes of the mean averages of the simulation output when different number of replications are used. The results are presented below.

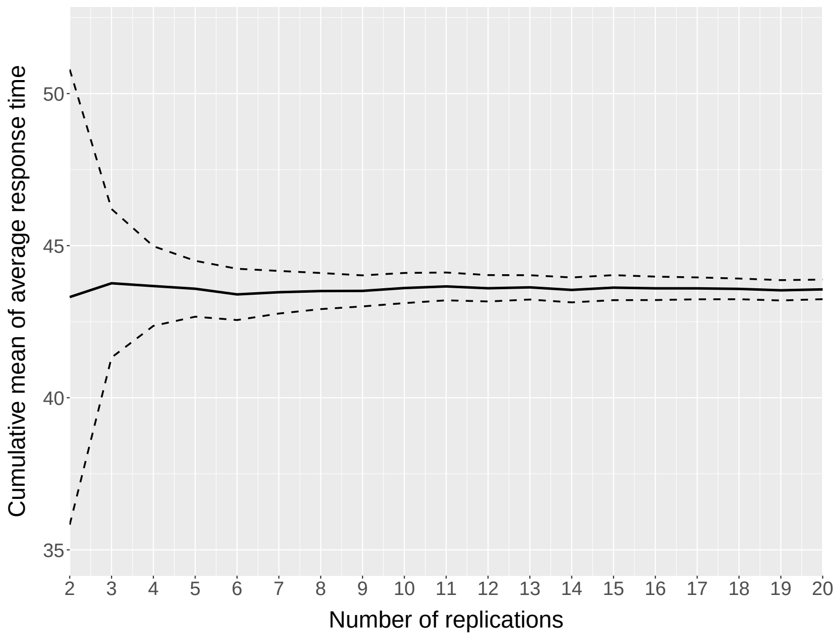


Figure 3.20: Cumulative mean of average response time with 95% confidence intervals

In Figure 3.20, the solid line displays the cumulative mean of average response responses, while the dashed lines show the corresponding confidence intervals at the 95% level. It can be observed that the confidence interval becomes narrow as the number of replications increases. Particularly, the wideness of the confidence interval is significantly reduced as the number of replications increases from 2 to 5. Then only small improvements can be achieved event the number of replications rises up to 20. This suggests that performing five replications is appropriate. Similar results have been obtained for other performance indicators including garage percentage and the average driving distance (see Appendix A.4). Therefore, our simulation experiments use the following setting: five replications, each corresponding to a simulation run of ten weeks with the first week used for the warm-up purpose.

3.6.2 Results of Simulation Validation

As described in Section 3.5.7, we use the historical data to drive the simulation and compare the model output with those obtained from the real system. The data collected from the company covers the time period from 01 May 2014 to 31 May 2014. We discard the data of the last three days (28/05/2014 to 31/05/2014) and split the rest equally into four groups, each corresponding to a task instance covering a time period of seven days. In addition, we extract the corresponding shift data covering the same period of each task instance. As described in the previous section, a warm-up period must be applied to the simulation model in order to avoid initialisation bias. Thus, we also generate four initialisation sets, each consisting of random tasks and shift data covering a time period of one week. The simulation starts with running an initialisation set. Once the initialisation is completed, the model runs with a real instance and collects the results. Instead of performing multiple replications, we only conduct one simulation run for each of the real instances since there is no randomness in those instances.

	Real System		Simulation		Mean	95% Confidence	
Instance	Mean	SD.	Mean	SD.	Difference	Interval	
1	44.78	38.76	43.26	42.09	−1.53	−2.78	−1.82
2	44.11	41.77	44.94	38.72	0.83	−1.23	−0.28
3	45.64	41.07	46.26	45.58	0.61	−0.25	0.78
4	46.30	43.19	45.87	41.92	−0.44	−2.29	−1.28
Average	45.21	41.20	45.08	42.08	−0.13	−1.63	−0.65

Table 3.8: Comparison of response times produced by simulation model and real system

Table 3.8 presents the statistical comparison of the response times produced by the simulation model and the real system. Columns titled Mean and SD. show the mean values and standard deviations of the obtained output data. In addition, we compute the mean differences between the values of the simulation model and the real system, and also calculate the 95% confidence intervals of the mean differences using the method described in (Robinson, 2004). The results are reported in the last three columns of the table. The average response times produced by the simulation model are very close to those of the real system, with an overall mean difference of -0.13. The confidence interval for the instance 3 covers zero, which indicates that the simulation output for instance 3 is not significantly different from the output of the real system in terms of the response times. The confidence intervals for other instances are on the left-hand side of zero, which suggests that the average response times produced by the simulation model are slightly lower than those of the real system.

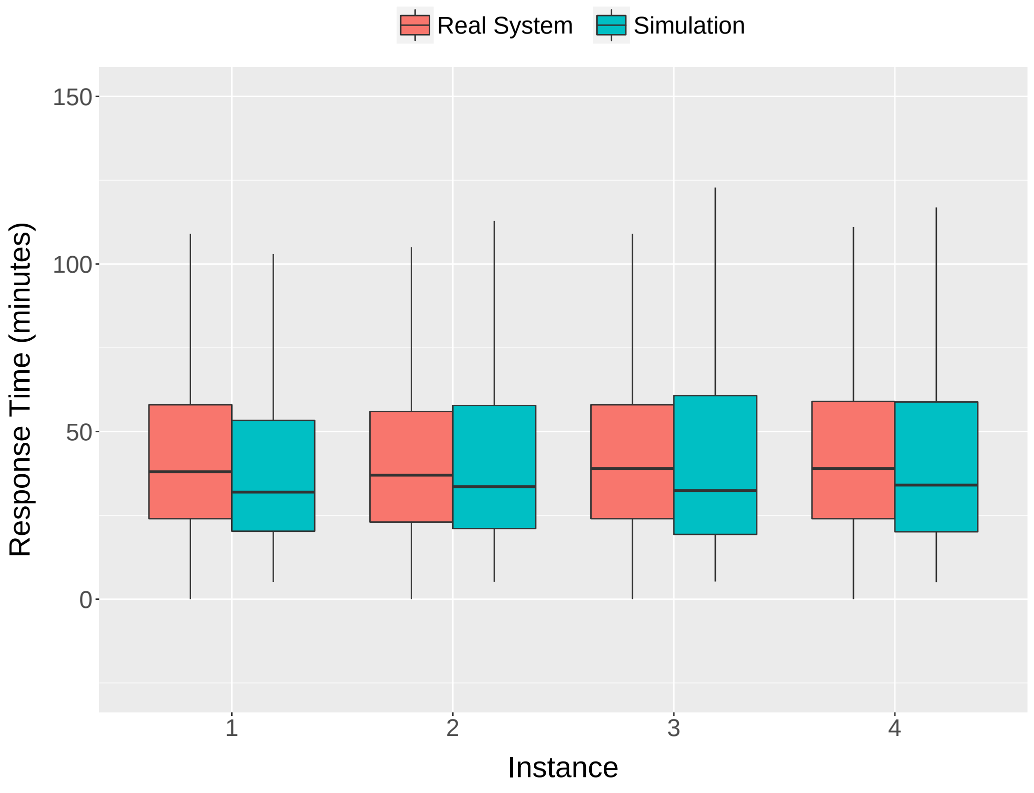


Figure 3.21: Distribution of response times produced by simulation and real system

The standard deviations of the simulation and the real system outputs are both around 40. In addition, we use box plots to examine the spread of the output data. Figure 3.21 presents that the output data of the simulation and the real system have similar patterns and distributions. It can be concluded that the developed model can provide reasonable estimation to the real system in terms of the response times.

Figure 3.22 shows the comparison of garage percentages, where the red bars and blue bars present the results obtained from the real system and the simulation model respectively. For each instance tested, the simulation model produces about 1% higher garage percentage than the real system. This can be explained by the differences in the staff data used by the simulation and the real system. In addition to the scheduled staff shifts, the real system has a group of standby staff that can be assigned to perform

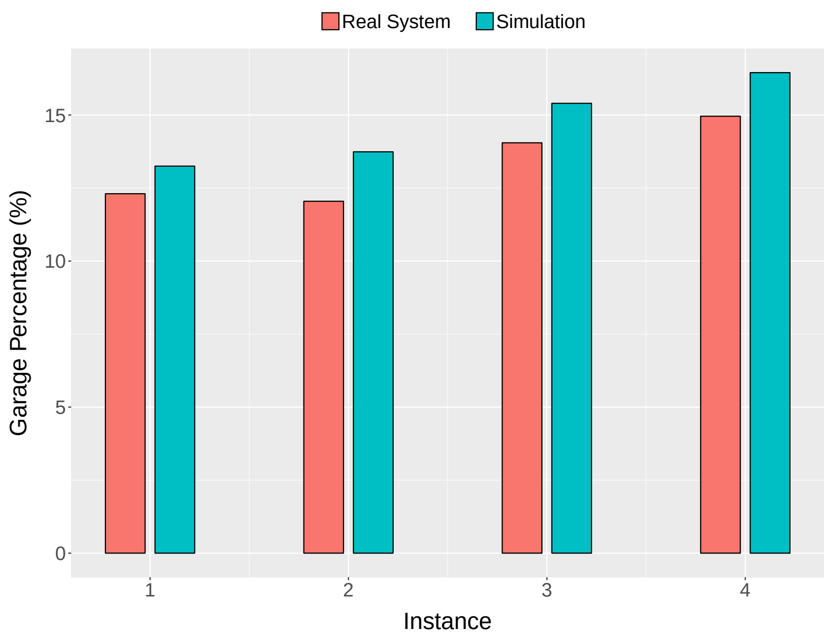


Figure 3.22: Comparison of the garage percentage

services if the actual demand is higher than expected. Moreover, the staff may work overtime if necessary. This implies that the real system can dynamically adjust the staff rota based on the real-time information of the actual demand. However, the simulation model does not consider the standby staff and overtime possibility, and thus it has to reply on the garage option when facing unexpectedly high demand. We have examined the performance of the simulation model when the standby staff is used and overtime is allowed. The obtained average response times and garage percentages are significantly lower than those of the real system. This indicates that simply adding standby staff and allowing the overtime possibility leads to an over estimation of the actual staffing level. The real system actually has experienced dispatchers to make decisions on when to use those additional resources. However, it is difficult to incorporate this feature into the simulation model. Although there exist gaps between the garage percentages obtained from the model and the real system, the data appear to have the same pattern, where the instances 3 and 4 have higher garage percentages than instance 1 and 2. This implies that the simulation model is able to accurately reflect the changes of the garage rate when different task instances are used. Therefore, it can be concluded that our simulation model provide a sufficient accuracy level for the purpose at hand.

3.6.3 Evaluation of the Model under Different Demand Levels

This section presents the results of evaluating the model performance under different demand levels. Let β be the demand level. We examine 16 values of β, selecting from $\{50\%, 60\%, ..., 200\%\}$. The current demand level β^* is represented by 100%. Thus,

$\beta = 50\%$ indicates the scenario of reducing β^* by 50%, while $\alpha = 200\%$ refers to the scenario of doubling the value of β^*. To generate a task instance for a given β value, we first adjust arrival rate $\lambda_i, i \in \{1, 2, .., 72\}$ by $\beta\lambda_i$, and then use the adjusted arrival rates as input to run the instance generator described in Section 3.5.5. The performance of the model under different demand levels is assessed through simulation experiments using the setting discussed in Section 3.6.1. The results are illustrated as follows.

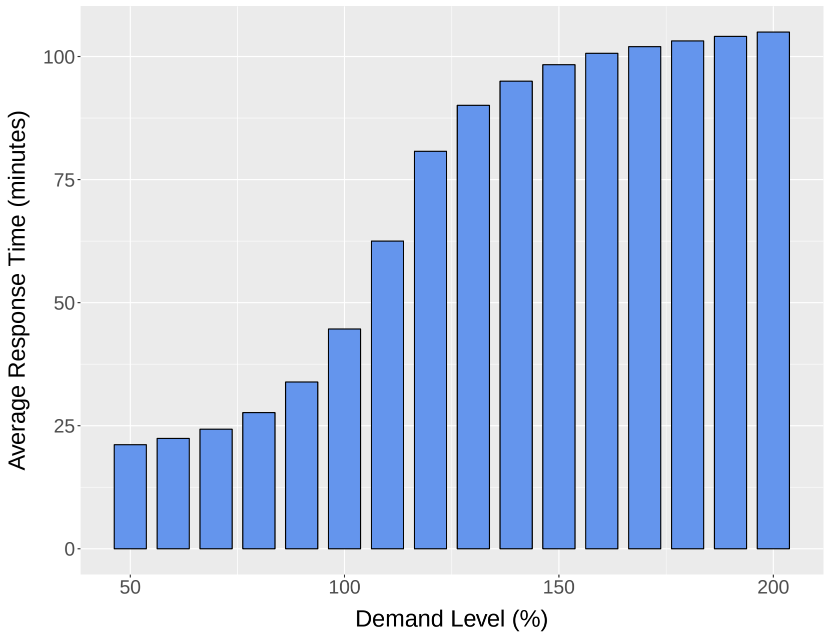


Figure 3.23: Average response time vs. demand level

Figure 3.23 shows the average response times obtained from the simulation model under different demand levels. It can be observed that when the demand level increases from 50% to 70%, the average response time remains roughly on the same level, which is 24 minutes. This actually implies the lower bound of the average response time. When the demand level rises from 80% to 140%, the average response time increases from 27 minutes to 94 minutes. However, the increasing rate of the average response time drops quickly as the demand level goes up from 130% to 150%. It can be expected that the average response time will reach a peak value when the demand level increases up to a certain point. This is due to the restriction of the time window constraints, where the expected response time must be less than 120 minutes when making the dispatching decision for each request. Therefore, when the job queue of a vehicle has been filled with a certain number of tasks, we cannot assign new requests to this vehicle as it violates the time window constraints.

The green bars of Figure 3.24 present the average number of requests responded by the AA resources per day. When the demand level increases from 140% to 150%, the increase in the average number of accepted requests is very small. This implies that the current staffing level has a capacity limit, which is about $12,000$ requests per day. When

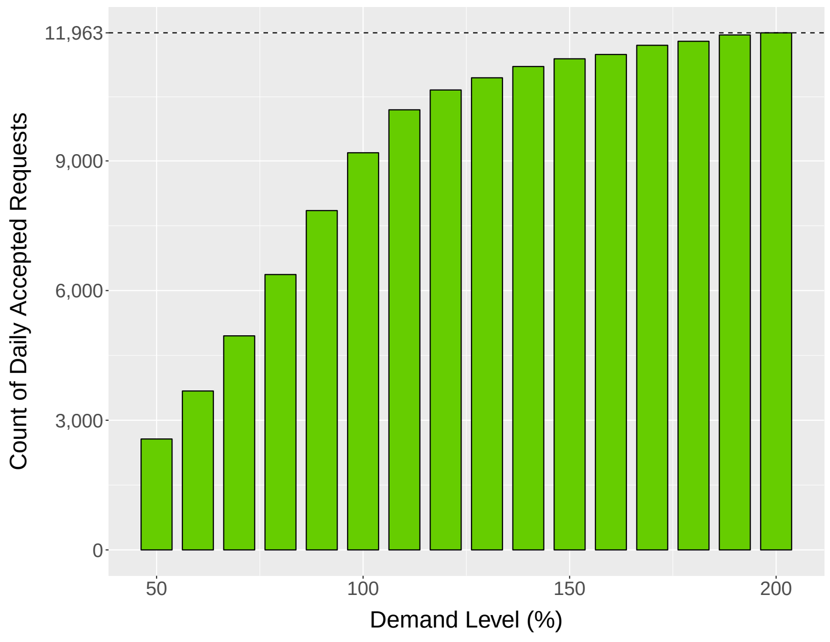


Figure 3.24: Average number of accepted requests per day vs. demand level

the demand level is higher than this limit, the system has to relay on the garage option, which results in an increasing garage percentage (as illustrated in Figure 3.25).

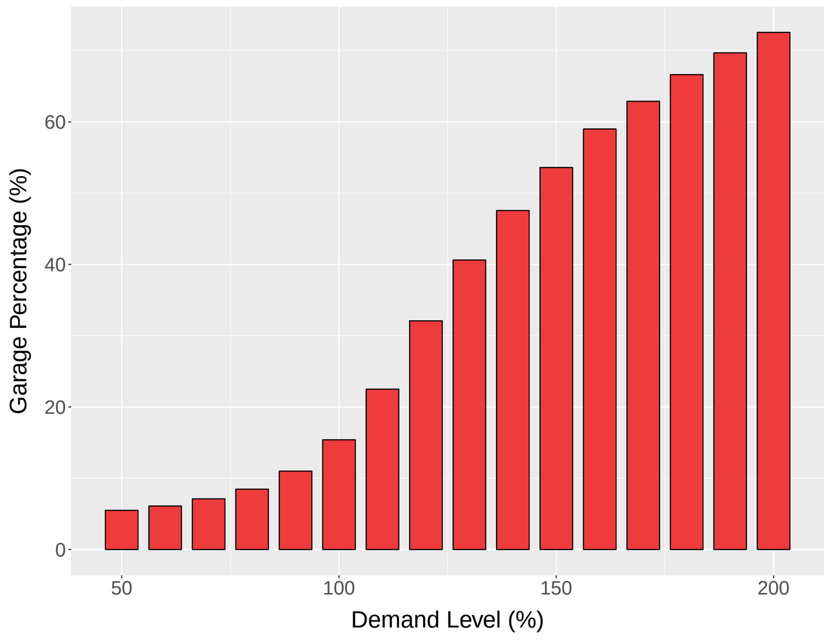


Figure 3.25: Garage percentage vs. demand level

3.6.4 The Effect of the Response Time Limit

To evaluate the effect of the response time limit T^{Max}, we conduct simulation experiments using a normal demand level ($\beta = 100\%$). We examine 13 values of T^{Max}, selecting from 30 minutes to 150 minutes using a step size of 10 minutes. The obtained results are displayed in figures below.

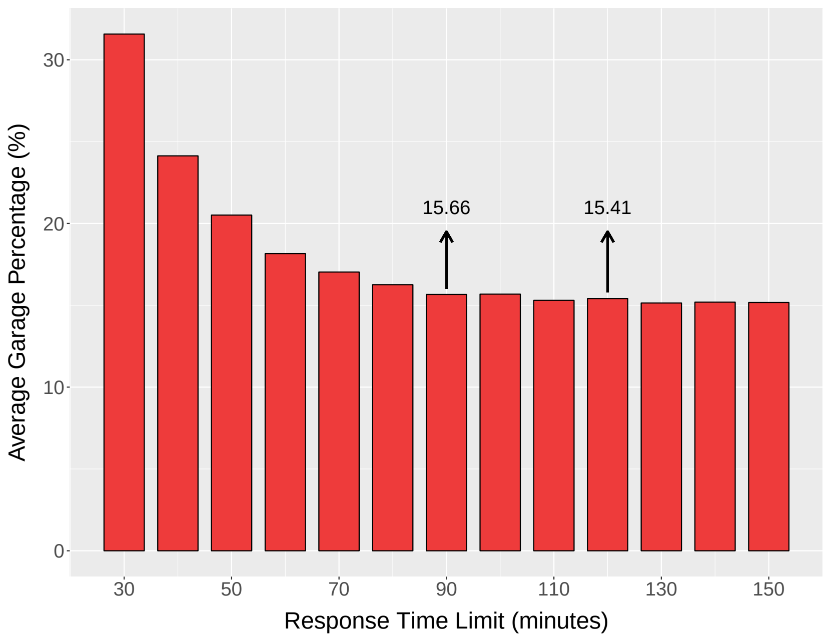


Figure 3.26: Average garage percentage vs. response time limit

Figure 3.26 presents the average garage percentage produced by the simulation model when different values of T^{Max} applied. When T^{Max} is set to 30 minutes, the model has to reject a large proportion of requests because the internal resources are not able to reach these requests within the required time limit. With a higher garage percentage, the model is able to achieve a smaller average response time (as shown in Figure 3.27). The graph also shows that the average garage percentage decreases as the value of T^{Max} increases from 30 minutes to 90 minutes, then it remains briefly on the same level, even T^{Max} increases up to 150 minutes. This can be explained by the fact that the garage decision is not only determined by T^{Max} but also the service distance constraint ($D^{Max} = 60$km). Using the time function provided by the company, the driving time required for a driving distance of 60km is 80.54 minutes. Thus, even T^{Max} is allowed to take large values, the driving time of technicians must be less than 80.54 minutes. This actually significantly restricts the possibility of accepting requests that have large expected response times.

Figure 3.27 shows that the average response time consistently increases as the value of T^{Max} increases. This indicate that the average response time is more sensitive than the garage percentage to the changes of T^{Max}. The current value of T^{Max} used by the real

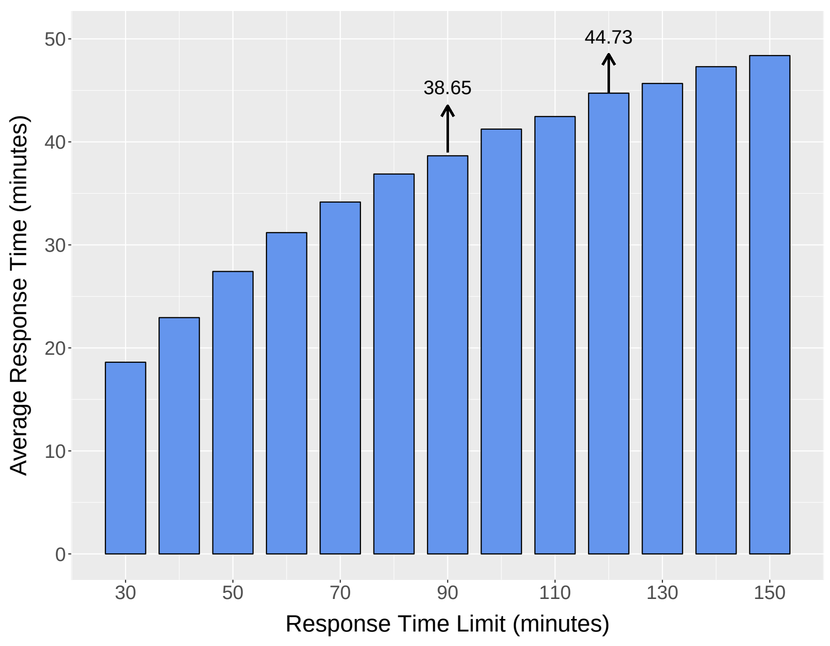


Figure 3.27: Average response time vs. response time limit

system is 120 minutes, which gives an average response time of 44.73 minutes. When the value of T^{Max} is decreased to 90 minutes, the average response time is improved by 13.59%, which is 38.65 minutes. On the other hand, the corresponding increase on the garage percentage is only about 0.25%. This suggests that under the current conditions of the real system, using $T^{Max} = 90$ can provide a better performance than using $T^{Max} = 120$, since it is able to achieve around 10% improvement on the average response time with the garage percentage increases less than 1%.

3.6.5 Comparison of the QR and SP Policies

In addition to the QR policy, we have developed a SP policy, described in Section 3.5.6. The performance of the SP policy is compared with QR policy using the simulation model. The results are discussed as follows.

Dispatching Policy	Mean Response Time	Garage Percentage (%)	Mean Dispatching Distance (km)
SP	63.75	16.94	9.18
QR	43.68	15.08	13.03

Table 3.9: The comparison of the QR and SP policy

As shown in Table 3.9, the average response time achieved by the QR police is 43.68 minutes, which is 31.48% lower than that of the SP policy. This indicates that the closest vehicle may not be the one giving the quickest response to the customer. Instead, the strategy of always assigning the closest vehicle actually leads to the situation that some customers have to wait a significant amount of time before the assigned vehicles become available for them. The average garage percentage produced by the QR policy is 16.39%, and it is about 2% lower than that of the SP policy. In addition, we compute the average dispatching distance of each request, and the results produced by the QR and SP policies are 13.03km and 9.18km respectively. The SP policy improves the driving distance by nearly 30% than the QR policy. This suggests that potential cost savings may be achieved by applying a dispatching policy that considers both the response time and driving distance.

3.7 Conclusion

This chapter presents a discrete-event simulation model of real-time emergency vehicle dispatching and routing, developed based on a case study of a British company providing emergency road services. The developed model was used to identify key characteristics of the real system, as illustrated below:

1. Under the current system setting, the average response time is around 45 minutes, the garage rate is about 15% and the average dispatching distance is about 13 km;
2. With different demand levels, the average response time varies between 20 minutes and 110 minutes;
3. Under the current staffing level, the system's capacity is around 12,000 requests per day. For any demands beyond this limit, the system has to fully relayed on the garage option.

In addition, the simulation model was used to evaluate the effect of the response time limit T^{Max}. We found that compared to strategy of using $T^{Max} = 120$ (default setting of the real system), using $T^{Max} = 90$ can lead to around 10% improvement on the average response time with the garage rate increases less than 1%. Finally, we assessed the performance of two dispatching policies: quickest response (QR) and shortest path (SP). The former reflects the dispatching policy used by the company, while the latter is a naive greedy algorithm which always dispatches the closest vehicle. The experimental results show that these two policies can lead to quite different solutions. Compared to the QR policy, the SP policy can produce nearly 30% improvement on the average dispatching distance, however, its average response time is 30% greater than that of the QR policy. This indicates that a better policy that considers both the response time and driving distance may be possible.

Chapter 4

Iterated Local Search for Workforce Scheduling and Routing Problem

The integration of scheduling workers to perform tasks with the traditional vehicle routing problem gives rise to the Workforce Scheduling and Routing Problems (WSRP). In the WSRP, a number of service technicians with different skills, and tasks at different locations with pre-defined time windows and skill requirements are given. It is required to find an assignment and ordering of technicians to tasks, where each task is performed within its time window by a technician with the required skill, for which the total cost of the routing is minimized. This chapter describes an iterated local search (ILS) algorithm for the WSRP. The performance of the proposed algorithm is evaluated on benchmark instances against an off-the-shelf optimizer and an existing adaptive large neighbourhood search algorithm. The proposed ILS algorithm is also applied to solve the skill vehicle routing problem, which can be viewed as a special case of the WSRP.

4.1 Introduction

The workforce scheduling and routing problem (WSRP) and its variants are commonly faced by many service providers, and have applications of home health care, field technician scheduling, security personnel routing and manpower allocation.

The term WSRP is coined by Castillo-Salazar et al. (2012), and refers to a class of optimization problems where service personnel are required to carry out tasks at different locations. For example, nurses visiting patients at their homes, and technicians performing maintenance jobs in different companies can each be modelled as a WSRP. As service personnel need to travel between different locations, minimizing their distances and times for travel is usually considered as one of the objectives when making operational decisions. This results in a routing problem of finding a set of least cost routes for a given workforce, where each route consists of a sequence of locations. Sometimes, tasks have associated time windows, within which service must start. This type of problem can be modelled as an extension of the vehicle routing problem with time windows (VRPTW), which is a well-known variant of the classical vehicle routing problem (VRP).

Service personnel often specialize in different skill domains, and possess skills at different levels. The tasks themselves have different skill requirements. For example, in the telecommunications industry, tasks may include maintenance, installation, construction and repair jobs, and technicians are trained in skills that allow them to only be able to service a subset of these tasks. Thus, skill compatibility must be taken into account to ensure that tasks are performed only by qualified personnel. The associated scheduling problem involves the assignment of tasks to service personnel. In some applications, tasks can be outsourced to a third party, albeit at the expense of additional cost, if appropriate resources are not available to provide the required service, or better operational performance can be achieved. The version of the WSRP that we consider allows for outsourcing.

Due to its complexity, most of the existing research on the WSRP has aimed at developing efficient heuristic solution algorithms. However, most of them are sophisticated and highly problem specific. In this paper, a simple heuristic algorithm based on iterated local search (ILS) is proposed to solve the WSRP. ILS is one of the most conceptually simple and robust algorithms (Burke et al., 2010). The essential idea of ILS is that when the local search is trapped at a local optimum, the ILS perturbs the previously visited local optimum instead of generating a new initial solution, and then restarts the local search from this modified solution (Lourenço et al., 2003). Although the ILS has a very simple framework, it has been successfully applied to a wide variety of optimization problems including the graph colouring problem (Chiarandini and Stützle, 2002), the job shop scheduling problem (Lourenço, 1995) and the vehicle routing problem (Hashimoto et al., 2008; Chen et al., 2010; Walker et al., 2012; Penna et al., 2013; Michallet et al., 2014) . However, no study has been reported on the application of the ILS to the WSRP, which is the aim of this paper. The contribution of the paper is a fast and simple algorithm for the WSRP with the objective of minimizing the total travel cost and outsourcing cost. The proposed algorithm is also applied to solve the skill vehicle routing problem (Skill VRP). To the best of our knowledge, this is also the first ILS approach for the Skill VRP.

The remainder of the paper is organized as follows. Section 2 reviews the related literature on the WSRP. A formal definition of the problem is presented in Section 3. Section 4 gives a description of the proposed ILS. Computational results for benchmark instances are presented in Section 5. The paper ends with some concluding remarks in Section 6.

4.2 Related Works

Recent studies on the WSRP include the work of Kovacs et al. (2012). They present an adaptive large neighbourhood search (ALNS) algorithm to solve the service technician routing and scheduling problem (STRSP). In this problem, tasks are associated with time windows and skill requirements, outsourcing tasks is allowed, and team building may be required in order to fulfil skill requirements of difficult tasks. The objective is to minimize the total operational cost comprising the routing and outsourcing cost. The scheduling aspect of this problem is adapted from the study of Cordeau et al. (2010), which considers a technician and task scheduling problem arising in a large telecommunications company. Cordeau et al. (2010) focus on the construction of teams and the assignment of tasks to teams without considering routing costs between tasks. Their problem is solved by using a construction heuristic and an ALNS algorithm. Pillac et al. (2013b) extend the study of Kovacs et al. (2012) by taking tools and spare parts into account, where each task must be carried out by a technician with the required skills, tools, and spare parts, and within the prescribed time window. The problem is solved by a matheuristic consisting of a parallel version of ALNS algorithm and a mathematical programming based post-optimization procedure.

Xu and Chiu (2001) also consider a field technician scheduling problem arising in the telecommunications industry. The objective is to maximize the number of jobs scheduled to technicians, while accounting for each job's priority and skill constraints. Three different heuristic approaches, namely, a greedy heuristic, a local search algorithm, and a greedy randomized adaptive search procedure (GRASP) are proposed to solve the problem. Castillo-Salazar et al. (2015) describe a greedy heuristic to address the WSRP with five types of time-dependent constraints, which model the relationship between tasks, e.g. one task needs to start after the completion of another task.

A variant of the WSRP is the skill vehicle routing problem (Skill VRP), which is introduced by Cappanera et al. (2011). The Skill VRP differs from other problems reviewed above in two aspects: (1) tasks do not have associated time windows, and (2) the routing costs depend both on the travelling distance and the technician in such a way that increasing the skill level of the technician causes an increase in costs. The use of technician-dependent routing costs is motivated by practical applications, since high-skilled employees usually have higher salaries than those with only basic skills. The Skill VRP is also studied by Schwarze and Voß (2012), but their study incorporates load balancing and resource utilization when constructing tours for service vehicles. Their motivation for proposing this model is their finding that many Skill VRP solutions usually use only a subset of vehicles, and a considerable number of tasks are assigned to vehicles that have higher skills than necessary.

Some studies have considered stochastic elements in the WSRP. For example, Weintraub et al. (1999) study a scheduling and routing problem for service vehicles belonging to an electric utility company in Chile, where service requests are stochastic. Pillac et al. (2012) also consider a technician routing and scheduling problem with stochastic service requests, which is solved by a parallel adaptive large neighbourhood search (pALNS) and a multiple plan approach. Binart et al. (2016) solve a field service routing problem with stochastic travel and service times using a two-stage stochastic programming model. Finally, Chen et al. (2015) describe a technician routing problem with experience-based service times, where technicians learn over time, which results in service times being reduced as experience increases.

Other problems closely related to the WSRP are the site-dependent vehicle routing problem with time windows (Cordeau and Laporte, 2001; Cordeau et al., 2004), the home health care scheduling problem (Blais et al., 2003; Bertels and Fahle, 2006; Akjiratikarl et al., 2007) and the manpower allocation problem (Dohn et al., 2009).

4.3 Problem Definition

In this section, we first provide a formal description of the WSRP problem that we address. We then formulate a mixed integer programming (MIP) model for our problem.

The WSRP is defined on a complete graph $G = (V, A)$, where $V = \{0, 1, ..., n + 1\}$ is a set of vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is a set of arcs. The vertex 0 denotes the depot and vertex $n + 1$ is a copy of the depot, and $C = V \setminus \{0, n + 1\}$ represents the set of vertices that each has a unique task. Depending on the context, we refer to a task i or a vertex i for any $i \in C$. A set K of technicians are available to perform the tasks. Each technician is specialized in a number of skill domains at different proficiency levels. Each task $i \in C$ has an associated service duration d_i, a time window $[e_i, l_i]$ within which service should commence, and a skill requirement. The depot and its copy also have time windows, which define the earliest departure time e_0 and the latest return time l_{n+1} of any technician. Also, the route duration of each technician must not exceed a given time D. Each arc $(i, j) \in A$ has an associated cost c_{ij} and travel time t_{ij}.

In the studies of Cordeau et al. (2010) and Kovacs et al. (2012), each technician's skills and each task's skill requirements are described by skill matrices, which are used to determine if a single technician or a team of technicians would be able to perform a given task. In this paper, we do not consider the possibility of building a team of technicians, and thus simply define a binary parameter q_i^k, where $q_i^k = 1$ if technician $k \in K$ is qualified to perform task $i \in C$, and $q_i^k = 0$ otherwise. The values of q_i^k can be easily computed based on technicians' skills and tasks' skill requirements. Finally, any task $i \in C$ can be outsourced by incurring a cost o_i, in the event that resources are insufficient or too expensive to undertake all of the tasks.

The WSRP can be formulated as a mixed integer programming model that contains the following binary variables:

$$x_{ij}^k = \begin{cases} 1 & \text{if arc } (i,j) \text{ is traversed by technician } k, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall (i,j) \in A,\ k \in K;$$

$$y_i = \begin{cases} 1 & \text{if task } i \text{ is outsourced,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V;$$

and the continuous variable b_i^k, $\forall i \in V,\ k \in K$, that lies within the interval $[e_i, l_i]$ if technician k does not perform task i; otherwise, it is the time at which service of task i commences, or the leaving time and returning time of technician k from and to the depot when $i = 0$ and $n+1$ respectively.

The mathematical model is presented as follows:

$$\text{minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{i \in C} o_i y_i \tag{4.1}$$

subject to:

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k + y_i = 1 \qquad \forall i \in C \tag{4.2}$$

$$\sum_{j \in V} x_{ij}^k \leq q_i^k \qquad \forall k \in K,\ \forall i \in C \tag{4.3}$$

$$\sum_{j \in V} x_{0,j}^k = 1 \qquad \forall k \in K \tag{4.4}$$

$$\sum_{i \in V} x_{i,n+1}^k = 1 \qquad \forall k \in K \tag{4.5}$$

$$\sum_{i \in V} x_{ih}^k - \sum_{j \in V} x_{hj}^k = 0 \qquad \forall k \in K,\ \forall h \in C \tag{4.6}$$

$$b_i^k + (d_i + t_{ij}) x_{ij}^k \leq b_j^k + l_i (1 - x_{ij}^k) \qquad \forall k \in K,\ \forall (i,j) \in A \tag{4.7}$$

$$e_i \leq b_i^k \leq l_i \qquad \forall k \in K,\ \forall i \in V \tag{4.8}$$

$$b_{n+1}^k - b_0^k \leq D \qquad \forall k \in K \tag{4.9}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K,\ \forall (i,j) \in A \tag{4.10}$$

$$y_i \in \{0,1\} \qquad \forall i \in C \tag{4.11}$$

$$b_i^k \geq 0 \qquad \forall k \in K,\ \forall i \in V. \tag{4.12}$$

The objective function (1) minimises the total operation cost comprising routing and outsourcing cost. Constraints (2) ensure that each task is either visited exactly once or outsourced, while constraints (3) guarantee that the tasks can only be performed by technicians satisfying the skill requirements. Constraints (4) and (5) ensure that each technician departs from the depot and returns to the copy of the depot after completing their service. Constraints (6) are the typical flow conservation equations. Constraints (7) set the time variables b_i^k, while constraints (8) enforce the time window restrictions. Constraints (9) guarantee that the route duration for each technician is no more than the maximum time allowed. Constraints (10) and (11) represent the binary restrictions on variables x_{ij}^k and y_i, and (12) are the non-negativity constraints on the variables b_i^k.

4.4 Iterated Local Search

This section describes our proposed iterated local search (ILS) algorithm for solving the WSRP. The ILS consists of three main components: initial solution construction, local search procedure and perturbation mechanism. They are combined into a multi-start framework as given in Algorithm 2. At each iteration of the main loop between lines 3 to 18, an initial feasible solution s is constructed for the ILS loop (lines 6 to 14). At each ILS iteration, the local search procedure takes as input the solution s, and returns an improved solution s', which is accepted as the new best current solution if it is feasible and has a value $f(s')$ that is strictly smaller than that of the incumbent solution $\bar{s}$, denoted by $f(\bar{s})$. Then a new starting solution s for the local search procedure is generated by perturbing on the incumbent solution $\bar{s}$ (line 12). The ILS loop repeats until the maximum number of iterations without improvement $MaxIt_{NI}$ is met. Then the incumbent solution $\bar{s}$ replaces the global best solution s^* if $f(\bar{s}) < f(s^*)$. This procedure repeats until a predefined number *MaxIt* of iterations have been executed.

Algorithm 2 Iterated Local Search

```
1: procedure ILS
2:     $It \leftarrow 1$, $f(s^*) \leftarrow +\infty$
3:     for $It \leftarrow 1$ to $MaxIt$ do
4:         generate initial solution $s$
5:         set $\bar{s} \leftarrow s$, $It_{NI} \leftarrow 0$
6:         while ( $It_{NI} < MaxIt_{NI}$) do
7:             $s' \leftarrow$ Local Search $(s)$
8:             if (($s'$ is feasible ) and ($f(s') < f(\bar{s})$)) then
9:                 $\bar{s} \leftarrow s'$
10:                $It_{NI} \leftarrow 0$
11:            end if
12:            $s \leftarrow$ Perturb($\bar{s}$)
13:            $It_{NI} \leftarrow It_{NI} + 1$
14:        end while
15:        if ($f(\bar{s}) < f(s^*)$) then
16:            $s^* \leftarrow \bar{s}$
17:        end if
18:    end for
19:    Return $s^*$
20: end procedure
```

4.4.1 Search Space

A number of studies have shown that an efficient exploration of infeasible solutions can contribute significantly to the performance of a heuristic (Cordeau et al., 1997; Glover and Hao, 2011; Cordeau et al., 2001; Vidal et al., 2012, 2013b). We follow the same line of thought here and allow the ILS to search infeasible, as well as feasible solutions, where the constraint violations in the former relate to the route duration and time window constraints. However, the skill requirement constraint is always respected, since it is concerned with the scheduling aspect of the WSRP and its relaxation would enlarge the search space dramatically. A solution s is therefore evaluated by an augmented cost function, which is defined by

$$f(s) = c(s) + \alpha d(s) + \beta w(s), \tag{4.13}$$

where $c(s)$ is the total operation cost as defined in (1), and $d(s)$ and $w(s)$ are the total violations of duration and time window constraints, which are weighted by parameters α and β, respectively.

The time window violation is measured based on a method proposed by Nagata et al. (2010). If there is a late arrival to a customer $i \in C$ at time $a_i > l_i$, then it is assumed that there is a penalty for the delay $a_i - l_i$, and that service starts at time l_i. In case of an early arrival at time $a_i < e_i$, then the technician has to wait until time e_i, but the waiting time is not penalized. The same method is used by Vidal et al. (2013b), who refer to the penalty as 'time warp'. Figure 4.1 illustrates the waiting time and time warp of a route with visits involving five vertices $v_1, \ldots, v_5$. The horizontal axis corresponds to time, while the vertical axis presents the sequence of visits. The dots on each line show the start time of each visit, and the brackets on each line indicate the time window of the corresponding task. As seen in Figure 1, there are no penalties associated with tasks v_1, v_3, and v_5 as the visits are made within the respective time windows. The bold line displays a possible schedule having a waiting time period at vertex v_2 and a time warp at vertex v_4.

4.4.2 Move Evaluation

Most local search heuristics spend the largest part of the overall computational effort on move evaluation (Vidal et al., 2014). Efficient move evaluation techniques are therefore crucial for improving algorithm performance, particularly when the search space involves infeasible solutions.

The operation cost $c(s)$ consists of the outsourcing cost and the total travelling distance which can be computed in amortized $O(1)$ time (Kindervater and Savelsbergh, 1997). However, it takes $O(n)$ to compute the penalties $d(s)$ and $w(s)$ in (4.13).

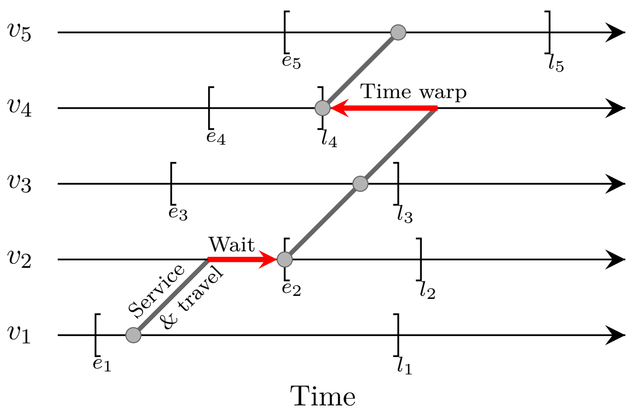


Figure 4.1: Illustration of waiting time and time warp

Nagata et al. (2010) propose an evaluation technique to compute the violation of time window constraints in amortized $O(1)$ time for most traditional neighbourhood operators including 2-opt, inter-route swaps, and inter-route inserts. Vidal et al. (2013b) extend this technique to allow the evaluation of both duration and time windows violations not only for inter-route but also for intra-route operators. A preprocessing phase is required to develop relevant data for their evaluation techniques, and the data must be updated once the route under consideration has been modified. Our ILS incorporates the technique (see Appendix B.1) proposed by Vidal et al. (2013b) to compute the violation of infeasible solutions.

4.4.3 Initial Solution Construction

Our procedure for constructing a feasible solution includes the following steps. The existence of a feasible solution is guaranteed due to the possibility of outsourcing. First, a task list L_1 is created as follows. The first task in the list is selected at random. The remaining entries in the list are constructed by sorting the remaining tasks of C in non-decreasing order of the angle they make with a line drawn from the depot to the randomly selected first task on L_1. Then, a technician list L_2 is constructed by sorting the technicians of K in non-increasing order of the number of tasks they are qualified to perform. We then randomly select a task $i \in C$ from the list L_1 and insert it into the cheapest feasible position of the route of the first technician on list L_2. If the insertion violates feasibility, we insert i into the following technician's route. In the case where no feasible route can be constructed that incorporates task i, we set i to be outsourced. The procedure repeats by inserting tasks sequentially into technicians' routes following the above steps, yielding a feasible solution that consists of technicians' routes and a list of outsourced tasks.

4.4.4 Local Search Procedure

Our local search procedure consists of an inter-route search operator, an intra-route search operator, and an update mechanism of the weight parameters α and β using in (4.13).

The inter-route search uses a single neighbourhood structure called Swap & Relocate that removes two paths, each containing at most two tasks from two different routes, and then exchanges them. One of these paths may contain zero tasks, which results in the path from the other route being relocated. Figure 4.2 gives an example of this operator which removes two successive vertices v_2 and v_3 from route r_1 and one vertex v_6 from route r_2, and then exchanges them. This neighbourhood structure is extended to allow an outsourced task to be swapped or relocated into the route of one of the technicians. When considering new routes created by this operator, the skill requirement constraints must be always respected, but any violations of duration and time window constraints are allowed.

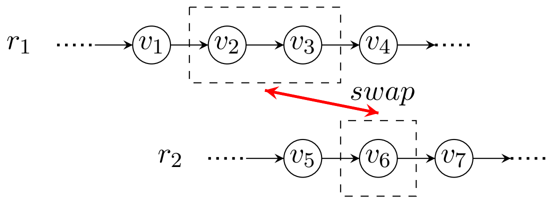


Figure 4.2: Example of the Swap & Relocate operator

The intra-route search consists of three neighbourhood structures, namely, opt1, opt2 and 2-opt (Croes, 1958), that operate on a single route. Operator opt1 removes one task and inserts it into another position on the same route, while operator opt2 is similar but removes and inserts two adjacent customers on a route. Operator 2-opt reverses the order of visits on this path, and then inserts the path back into the same position to form a new route r'. The cost of r' is evaluated by the method described in Section 4.4.2.

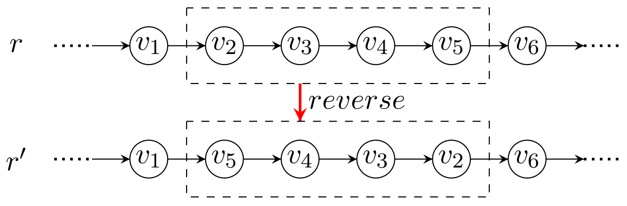


Figure 4.3: Example of the 2-opt operator

The inter-route search and the intra-route search can be combined in different ways within the local search procedure. To test the effect of the search strategy on the performance of the algorithm, we investigate the three following strategies:

1. Execute only the inter-route search operator;
2. Execute both the inter-route and intra-route search operators at each iteration of the local search procedure;
3. Apply the intra-route search as a post-optimization procedure on the locally optimal solution returned by the inter-route search.

After each iteration of the local search, the weight parameters α and β are adjusted according to the duration violation $d(s)$ and the time window violation $w(s)$ of the incumbent solution s as follows. If $d(s) = 0$, then the parameter α is divided by a factor $1 + \delta$; otherwise, it is multiplied by $1 + \delta$, where $\delta > 0$ is a parameter that controls the strength of adjustment. The same rule applies to the parameter β with respect to $w(s)$. The initial values of α and β are both set to 1, as suggested by a number of studies that have similar cost functions and weight parameters (Cordeau et al., 2001, 1997; Ibaraki et al., 2008; Nagata et al., 2010).

The structure of the local search procedure is illustrated in Algorithm 3. The current best solution s' is set to the incumbent solution s. Then s is taken as input by the *SearchStrategy* function, which applies inter-route or intra-route search depending on the search strategy selected and returns an improved solution $\hat{s}$ if such a solution exists. If $f(\hat{s}) < f(s')$, then $\hat{s}$ replaces s' as the current best solution. Then, the duration violation $d(\hat{s})$ and time window violation $w(\hat{s})$ are computed, and parameters α and β are adjusted accordingly by the control mechanism described above. The pre-processed data for the routes that have been modified at the current iteration are updated. This procedure repeats until the local search becomes trapped at a local optimal solution.

4.4.5 Perturbation Mechanism

The perturbation mechanism uses a random cross exchange operator, which removes two paths from two randomly selected routes and exchanges them. Figure 4.4 gives an example of the perturbation operator which removes a path of four successive visits from route r_1 and a path of two successive visits from route $r_2 \neq r_1$, and then exchanges them. Violations of duration and time window constraints are allowed, but the skill requirement constraint must be respected. The perturbation procedure is always carried out on the best solution found thus far, and applies the random cross exchange operator p times, where p is a positive integer denoting the perturbation strength.

Algorithm 3 Local Search Procedure

```
 1: procedure LocalSearch
 2:     input solution $s$
 3:     set $\alpha = 1$ and $\beta = 1$
 4:     set $s' = s$
 5:     set $LocalOptimumFound$ = false
 6:     while ($LocalOptimumFound$ = false) do
 7:         $\hat{s} \leftarrow SearchStrategy(s)$
 8:         if $(f(\hat{s}) < f(s'))$ then
 9:             $s' \leftarrow \hat{s}$, $s \leftarrow \hat{s}$
10:             Compute $d(\hat{s})$ and $w(\hat{s})$, and update $\alpha$ and $\beta$
11:             Update PreprocessData
12:         else
13:             set $LocalOptimumFound$ = true
14:         end if
15:     end while
16:     return $s'$
17: end procedure
```

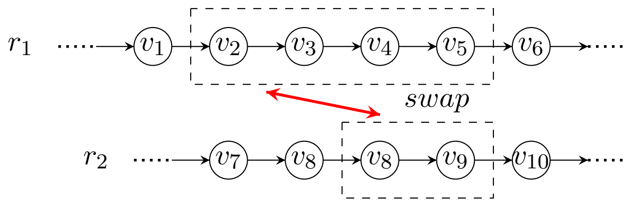


Figure 4.4: Example of the random cross exchange

The perturbation strength p is a crucial parameter of the ILS. If p is too small, the local search may not be able to escape from a locally optimal solution. If p is too large, the ILS may behave similar to a random restart algorithm, making it difficult to discover better quality solutions (Lourenço et al., 2003). In order to determine the most appropriate value of p, we developed an adaptive mechanism, which adjusts p according to the number of consecutive iterations without improvement, denoting by It_{NI}. Let γ be a trigger for the adjustment of p. More precisely, whenever It_{NI} has increased by γ, the value of p will be increased by 1 until it reaches the upper bound $\bar{p}$, where $\bar{p}$ is used to prevent excessively large values of p to be chosen. For example, if $\gamma = 10$ and $\bar{p} = 5$, then p starts from 1 and increases by 1 when $It_{NI} \in \{10, 20, 30, 40\}$.

4.4.6 Reducing Outsourcing Cost

As the cost of outsourcing a task is usually higher than that of serving it by internal resources, reducing the outsourcing cost is considered as an objective within the algorithm. This is achieved by a simple mechanism embedded in the perturbation procedure of the proposed ILS. At the beginning of the perturbation procedure, we check the list of outsourced tasks. If it is not empty, we randomly select a task and insert it to the cheapest position of the current solution, and then proceed with the perturbation procedure; otherwise, we only apply the random cross exchange operator. The insertion of outsourced tasks and the perturbation procedure is likely to produce an infeasible solution, which will be improved by the local search procedure. Infeasible solutions are evaluated by a cost function defined in (4.13), and weight parameters α and β are dynamically adjusted based on the rule described in Section 4.4.4. If the local search procedure cannot repair the infeasibility during the first few iterations, the weight parameters will be adjusted to large values, such that the cost of scheduling a task to a technician becomes greater than the cost of outsourcing it. As a consequence, the local search tends to repair the infeasibility by simply outsourcing the relevant tasks. In order to avoid the overuse of the outsourcing option, we force the local search procedure to always select improved solutions with lower outsourcing costs, even if solutions with higher outsourcing costs but lower overall costs exist.

4.5 Computational Results

This section presents results of our computational tests conducted to assess the performance of the proposed ILS. The ILS algorithm is coded in C++, and run on a personal computer with Intel Core i5-3570 3.40 GHz processor and 4GB Memory (RAM). The MIP model is implemented on the same machine, and solved by the commercial solver CPLEX 12.6. Our ILS results are compared with existing solutions of an ALNS algorithm (Kovacs et al., 2012), where the reported ALNS results are based on the average of five runs of the algorithm. To maintain consistency and provide a fair comparison, we also perform five random runs of the ILS for each instance tested and report the obtained results.

4.5.1 Test Instances

The experiments are conducted using the technician routing and scheduling problem (TSRP) instances introduced by Kovacs et al. (2012). These instances are adapted from the Solomon's benchmark instances (Solomon, 1987) for the VRPTW and the test instances provided for the ROADEF 2007 challenge. They are available online at: `http://prolog.univie.ac.at/research/STRSP/`.

The set of instances of Kovacs et al. (2012) are generated using 12 instances of Solomon (1987), namely, R101, R103, R201, R203, C101, C103, C201, C203, RC101, RC103, RC201, RC203, where R, C and RC represent the random, clustered and a mix of random and clustered geographical setting, respectively. Instance sets with prefixes R1, C1 and RC1 have a short scheduling horizon, while those with prefixes R2, C2 and RC2 have a long scheduling horizon. The final two digits in the name of the instance indicate the time window density. In the 01 instances, all customers are associated with time windows, while in the 03 sets, only 50% of customers have time windows. In terms of the skill requirements, Kovacs et al. (2012) generate three types of skill requirement matrices shown by 5×4, 6×6, and 7×4 based on the ROADEF data, where the rows of the matrices correspond to skill domains, and the columns correspond to skill levels under each skill domain. The customer data of Solomon's instances are randomly paired with the skill data, which results in a total of 36 test instances. All instances have 100 customers and a single depot. For each instance, Kovacs et al. (2012) define a 'team' and a 'no team' version. As our study does not consider the possibility of team building, we only use the 'no team' version of instances in our experiments. For each instance, there are two sets of technician data: one has a sufficient number of technicians that feasibility can be achieved without outsourcing, while the other has limited technicians such that it is impossible to service all tasks without the use of the outsourcing option. The outsourcing cost of a task i is defined as $o_i = 200 + \mu_i^{1.5}$, where μ_i measures the difficulty of task i, and is calculated as the sum of the skill requirement for i in the skill matrix. The outsourcing cost is always higher than the cost of assigning a task to a technician.

4.5.2 Parameter Setting

The ILS requires five input parameters as follows: *MaxIt*; $MaxIt_{NI}$; δ, which is the factor used to adjust weight parameters of duration and time window violations; $\bar{p}$, which is the upper bound that is used in the perturbation mechanism; and γ is the adjustment factor of the perturbation strength. The value of $MaxIt_{NI}$ is defined by Penna et al. (2013) as

$$MaxIt_{NI} = |C| + \lambda|K|, \tag{4.14}$$

where $|C|$ is the number of customers, $|K|$ is the number of technicians, and λ is a weight parameter determining the influence of $|K|$ on the value of $MaxIt_{NI}$. Thus, instead of finding the most appropriate value for $MaxIt_{NI}$, the value of λ is examined. To find the set of parameters that produces the best performance, we have carried out extensive parameter tuning experiments using a holdout set of instances containing different numbers of customers and technicians, as well as different skill settings. The results suggest that a parameter setting shown in Table 4.1 performs well.

Parameter	*MaxIt*	δ	$\bar{p}$	γ	λ
Value	5	0.5	5	20	10

Table 4.1: Parameter setting for the proposed ILS

4.5.3 Performance Measurement

The proposed ILS is evaluated against the MIP model and the ALNS (Kovacs et al., 2012) using benchmark instances. To compare the ILS and ALNS solutions, we compute the relative percentage difference defined as

$$\mathrm{Imp}^{\mathrm{S}}_{\mathrm{A/I}} = \frac{v^{\mathrm{S}}(\mathrm{ALNS}) - v^{\mathrm{S}}(\mathrm{ILS})}{v^{\mathrm{S}}(\mathrm{ALNS})} \times 100. \tag{4.15}$$

where $v^{\mathrm{S}}(\mathrm{ALNS})$ and $v^{\mathrm{S}}(\mathrm{ILS})$ represent values of the ALNS and ILS solutions respectively, and $\mathrm{S} = \{-, *, +\}$ denotes the minimum, mean, and maximum values over five random runs of the algorithm. For example, $\mathrm{Imp}^{-}_{\mathrm{A/I}}$ represents the relative percentage difference between the values of the best solutions found by the ALNS and ILS over five random runs. A positive value of $\mathrm{Imp}^{\mathrm{S}}_{\mathrm{A/I}}$ indicates an improvement of the ILS over ALNS; otherwise, the cost of the ILS solution is greater or equal to that of the ALNS solution.

By replacing $v^{\mathrm{S}}(\mathrm{ALNS})$ of the expression (4.15) with $v^{\mathrm{S}}(\mathrm{CPLEX})$, we obtain the relative percentage difference between the values of the ILS solutions and the optimal values produced by CPLEX, denoted by $\mathrm{Imp}^{\mathrm{S}}_{\mathrm{C/I}}$. There is no difference between $v^{-}(\mathrm{CPLEX})$, $v^{*}(\mathrm{CPLEX})$ and $v^{+}(\mathrm{CPLEX})$, as they all refer to the value of the optimal solution found by CPLEX.

In addition to the comparison of solution values, we compare the computational times required by our ILS and the ALNS. Kovacs et al. (2012) run their ALNS on a Pentium D computer with two 3.2 GHz CPUs and 4 GB memory (the algorithm only uses one CPU), which is different from our machine that is used to implement the ILS and MIP model. In order to provide a fair comparison of computational speed, we scale the reported CPU times according to the speed factors provided in the report of Dongarra (2014). The report does not cover the two computers considered in our experiments. Thus, we use a slower but similar computer (Pentium IV with 3.0 GHz) available in Dongarra (2014) instead of the computer used by Kovacs et al. (2012), and use a speed factor of 1573 Mflop/s (millions of floating-point operations per second). As there is no suitable substitute available in Dongarra (2014), we apply the same software used by Dongarra (2014) to record the speed factor of our computer, which yields 2462 Mflop/s. Based on the speed factors, the reported CPU times of the ALNS are adjusted by multiplying a factor of (1573/2462), when comparing with the ILS times.

4.5.4 Evaluation of Search Strategies

Results on comparing the three search strategies described in Section 4.4.4 are shown in Table 4.2. Columns headed Avg. show the average solution values produced by the ALNS and the ILS using three different strategies over five random runs. The corresponding relative percentage difference between the values of the ALNS solutions and ILS solutions are reported in columns titled $\text{Imp}^*_{\text{A/I}}$.

Comparing Strategy 1 with Strategy 2 and Strategy 3, it can be seen that by applying the intra-route search, the solution quality improves significantly from -0.21% to 0.54% and 0.51%, with the average computational time increasing accordingly. The intra-route search is seen to be especially useful on the R2, C2 and RC2 types of instances, which are characterized by a long scheduling horizon and a low number of technicians, where each route contains a relatively high number of tasks. Comparing Strategy 2 with Strategy 3, the difference between the average $\text{Imp}^*_{\text{A/I}}$ values is only 0.03%. However, the average computational time of Strategy 2 is about 13% higher than that of Strategy 3. Therefore, Strategy 3, which applies the intra-route search as a post-optimization procedure on the local optimum returned by the inter-route search, is recommended based on efficiency and effectiveness, and is used in the remainder of our tests.

			ALNS		Strategy 1			Strategy 2			Strategy 3		
Instances	$\|C\|$	$\|K\|$	Avg.	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU
C101_5x4	100	17	1111.08	66.50	1106.01	0.46	28.67	1100.36	0.96	29.38	1108.14	0.26	30.96
C103_5x4	100	17	1037.33	84.61	1028.28	0.87	36.96	1035.82	0.15	37.44	1032.09	0.50	40.46
C201_5x4	100	8	1180.93	54.29	1157.58	1.98	10.30	1157.86	1.95	16.19	1157.58	1.98	12.62
C203_5x4	100	8	1049.30	89.78	1070.44	−2.01	23.86	1055.75	−0.62	43.34	1057.83	−0.81	26.43
R101_5x4	100	25	1685.85	84.25	1671.28	0.86	45.40	1663.13	1.35	51.85	1670.08	0.94	53.90
R103_5x4	100	25	1249.91	101.77	1240.41	0.76	64.01	1240.81	0.73	59.36	1237.72	0.98	62.53
R201_5x4	100	7	1448.93	57.22	1461.82	−0.89	25.47	1436.38	0.87	39.05	1443.02	0.41	25.57
R203_5x4	100	7	1106.12	84.18	1127.99	−1.98	31.12	1097.38	0.79	49.92	1098.64	0.68	30.84
RC101_5x4	100	22	1716.07	78.25	1692.06	1.40	39.29	1694.78	1.24	47.47	1690.19	1.51	48.71
RC103_5x4	100	22	1354.11	91.41	1365.55	−0.84	52.43	1346.18	0.59	48.28	1348.31	0.43	50.39
RC201_5x4	100	9	1607.25	58.71	1619.35	−0.75	28.55	1605.45	0.11	36.47	1602.75	0.28	26.03
RC203_5x4	100	9	1166.50	87.01	1177.42	−0.94	29.65	1165.32	0.10	46.27	1167.93	−0.12	34.12
C101_6x6	100	16	1004.82	69.76	998.15	0.66	40.97	983.91	2.08	45.82	993.71	1.11	36.34
C103_6x6	100	16	897.86	88.17	920.12	−2.48	52.71	911.33	−1.50	53.24	910.20	−1.37	58.97
C201_6x6	100	7	821.55	55.65	832.92	−1.38	13.95	830.36	−1.07	29.01	832.69	−1.36	20.70
C203_6x6	100	7	703.10	99.75	727.77	−3.51	39.21	703.28	−0.02	47.66	708.82	−0.81	36.18
R101_6x6	100	26	1667.43	96.78	1655.83	0.70	60.47	1658.88	0.51	72.55	1658.82	0.52	74.52
R103_6x6	100	29	1231.49	116.00	1225.62	0.48	74.87	1224.92	0.53	79.70	1222.90	0.70	79.47
R201_6x6	100	7	1270.26	62.02	1288.10	−1.40	51.63	1290.01	−1.55	54.62	1278.29	−0.63	47.75
R203_6x6	100	7	951.84	99.46	948.33	0.37	50.72	929.96	2.30	101.70	933.32	1.95	55.20
RC101_6x6	100	24	1683.96	90.42	1676.27	0.46	62.59	1672.12	0.70	51.39	1671.86	0.72	55.13
RC103_6x6	100	24	1310.95	103.68	1327.05	−1.23	69.85	1301.77	0.70	61.91	1315.62	−0.36	64.58
RC201_6x6	100	8	1406.95	61.32	1392.39	1.04	43.80	1372.90	2.42	52.14	1377.50	2.09	49.11
RC203_6x6	100	8	1016.71	91.39	1028.09	−1.12	56.16	1013.42	0.32	78.29	1016.73	0.00	56.64
C101_7x4	100	17	1398.95	58.65	1374.40	1.76	17.32	1382.87	1.15	24.74	1381.59	1.24	23.25
C103_7x4	100	17	1239.22	75.26	1232.41	0.55	26.91	1232.61	0.53	27.12	1233.67	0.45	25.22
C201_7x4	100	8	1282.18	49.54	1265.58	1.29	8.72	1269.68	0.97	17.83	1262.94	1.50	10.71
C203_7x4	100	8	1151.27	76.83	1164.43	−1.14	20.34	1150.27	0.09	32.04	1142.22	0.79	22.92
R101_7x4	100	28	1793.95	88.52	1785.47	0.47	40.15	1787.10	0.38	50.00	1797.41	−0.19	43.76
R103_7x4	100	28	1375.09	102.08	1352.08	1.67	41.53	1368.30	0.49	35.89	1346.73	2.06	43.04
R201_7x4	100	10	1410.90	59.31	1414.88	−0.28	18.74	1403.89	0.50	29.16	1403.94	0.49	18.48
R203_7x4	100	10	1166.94	80.29	1197.09	−2.58	21.24	1176.64	−0.83	36.54	1166.52	0.04	28.34
RC101_7x4	100	23	1844.37	75.55	1845.04	−0.04	29.33	1825.72	1.01	32.97	1818.02	1.43	28.72
RC103_7x4	100	23	1455.33	85.04	1441.02	0.98	38.42	1442.00	0.92	35.12	1446.39	0.61	39.64
RC201_7x4	100	9	1701.25	52.78	1729.81	−1.68	21.52	1703.28	−0.12	20.11	1704.20	−0.17	16.21
RC203_7x4	100	9	1241.65	73.82	1243.55	−0.15	20.93	1234.72	0.56	37.47	1235.11	0.53	22.78
Average			1298.37	79.17	1299.57	−0.21	37.16	1290.81	0.54	44.78	1290.93	0.51	38.89

Table 4.2: Evaluation of local search strategies

4.5.5 Comparison of Performance

This section presents the results of evaluating our ILS against the MIP model and the ALNS using benchmark instances containing 25, 50, and 100 tasks. In the tables presented hereafter, the first group of columns shows the instance identifier, the number of tasks $|C|$, and the maximum number of technicians $|K|$. Columns Opt. and Avg. show, for each instance, the optimal solution value found by CPLEX, and the average solution values found by the ALNS and ILS over five random runs. Columns $\text{Imp}^*_{\text{C/I}}$ and $\text{Imp}^*_{\text{A/I}}$ give the relative percentage difference between the values of the ILS solutions and the CPLEX solutions and the ALNS solutions, respectively. The average number of outsourced tasks, the average number of technicians used, and the average CPU time in seconds are reported in the columns headed $|C_o|$, $|K^*|$, and CPU, respectively. Emboldening in the ILS columns is used to highlight values that correspond to an improvement over the corresponding values of the ALNS.

Table 4.3 gives experimental results on small instances containing 25 tasks. Compared to CPLEX, our ILS algorithm consistently finds optimal solutions in all five random runs for 19 out of 23 instances and produces an overall average gap of -0.18% over all instances. Moreover, the average number of outsourced tasks given by the ILS is exactly the same as that for CPLEX. Compared to ALNS, our ILS algorithm gives better solutions for four instances, in particular RC101_5×4 and RC101_6×6, for which the solutions found by the ILS improve the ALNS solutions by 9.32% and 12.56% respectively. The significant improvement on these two instances is achieved by the reduction in the number of outsourced tasks. To test the statistical significance between the performances of ALNS and ILS, we conduct the two-tailed Wilcoxon test on the paired samples between the average solution values obtained by ALNS and ILS. The test is performed at a 95% significance level, where a p-value of less than 0.05 indicates the rejection of the null hypothesis, which says that there is no significant difference between the results of ALNS and ILS. The p-value of the Wilcoxon test for instances containing 25 tasks is 0.24, which suggests that the performances of ALNS and ILS on this set of instances are similar. This can be explained by the fact that both ALNS and ILS can solve a large majority of small instances to optimality. Perhaps the most significant feature of ILS is the speed with which it produces good-quality solutions, and it is significantly faster than the ALNS. With an average CPU time of 0.11 seconds, it only requires 7% of the time used by the ALNS. Although our computer is faster, the effect of the computer speed is negligible compared to the improvement on CPU times.

Table 4.4 presents results of the experiments on instances with 50 tasks. Of the 12 instances, our ILS algorithm discovers optimal solutions for seven and yields an overall average deviation of -0.14% in comparison to CPLEX. The average deviation of the ILS from the ALNS in terms of the solution values is 0.67%, and it finds better solutions for five instances. The p-value of the Wilcoxon test for this set of instances is 0.06, which is very close to the margin of significance. This suggests that when the problem size increases to 50, our ILS tends to perform better than the ALNS. Using the computer speeds, the average computation time of the ALNS is adjusted to 4.77 seconds, which is still considerably greater than the 1.89 seconds for ILS.

For instances with 100 tasks, a time limit of 7200 seconds is imposed on CPLEX. Tables 4.5 and 4.6 report computational results on instances with limited and unlimited technicians, respectively. The third to fifth columns of each table are associated to the results of the MIP model solved by CPLEX, where the columns Best and Gap present, for each instance, the value of the optimal or best solution found by CPLEX within the time limit, and the percentage gap of the LP bound with respect to the best solution value. In addition, we report the minimum and maximum solution values found by the ALNS and ILS over five random runs in columns titled Min. and Max., and the corresponding percentage differences between the values of ALNS and ILS solutions are presented in columns $\text{Imp}^{-}_{\text{A/I}}$ and $\text{Imp}^{+}_{\text{A/I}}$ respectively. Proven optimal solutions are underlined.

Of the 36 instances with unlimited technicians, CPLEX is only able to find optimal solutions for 9, and for the 36 instances with limited technicians, the model finds optimal solutions for 5 instances within the required time limit. This indicates that instances with limited technicians tend to be more difficult to solve than those with unlimited technicians, as the former problem considers the additional set of decisions concerning the selection of tasks to be outsourced.

A comparison of ILS and ALNS on instances with 100 tasks and limited technicians is given in Table 4.5. Of the 36 instances, our ILS algorithm outperforms ALNS in 17. In particular, for instances R101_5×4, RC101_6×6 and RC101_7×4, the solutions found by the ILS are between 5% and 8% better in cost than those for ALNS. The significant improvement on these instances can be explained by the reduced use of the outsourcing option by the ILS. The average number of outsourced tasks of the ILS solutions is 9.76, which is about 3% less than the value of the ALNS solutions. To determine the statistical significance between the numbers of outsourced tasks produced by the ILS and ALNS on this set of instances, we conduct a two-tailed Wilcoxon test and a p-value of 0.004 is obtained. This confirms that our ILS uses significantly less outsourcing option than the ALNS, and also implies that the proposed mechanism of reducing outsourcing cost (described in Section 4.4.6) is effective. The average percentage difference between the ILS and ALNS solution values is 0.82%. Comparing the worst solutions found during five random runs, the ILS improves the ALNS solutions by 1.24%, which indicates that our ILS is more stable than the ALNS when performing multiple runs. The average computational time required by ALNS is 52.87 seconds, which is equivalent to 33.78 seconds after applying the conversion factor, and is 16.59% higher than that of ILS.

Table 4.6 provides a comparison of ILS and ALNS on large instances with unlimited technicians. The average number of outsourced tasks is not reported in this table, as these instances have enough technicians to avoid outsourcing. The ILS algorithm outperforms ALNS in 30 out of 36 instances, and improves the best solutions for 24 instances. Of the 9 instances that are solved to optimality by CPLEX, our ILS algorithm finds optimal solutions for 5 of them. The average percentage difference between the ILS and ALNS solution values is 0.64%. Moreover, the ILS solutions tend to have smaller deviations within five random runs since the overall average values of $\mathrm{Imp}^{-}_{\mathrm{A/I}}$ and $\mathrm{Imp}^{+}_{\mathrm{A/I}}$ are both greater than 0. In terms of speed, ALNS requires an average solution time of 79.17 seconds, which is equivalent to 50.58 seconds under the adjustment of computer speeds, but is still 20% higher than the average CPU time required by ILS. Lastly, we conduct a two-tailed Wilcoxon test on the solutions values of all large instances containing 100 tasks and a p-value of 0.02 is obtained. This indicates that our ILS has a significantly better performance than the ALNS on the set of large instances since the p-value is less than the chosen significance level 0.05.

			CPLEX				ALNS			ILS					
Instance	$\|C\|$	$\|K\|$	Opt.	$\|C_o\|$	$\|K^*\|$	CPU	Avg.	$\|C_o\|$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\text{Imp}^*_{\text{A/I}}$	$\|C_o\|$	$\|K^*\|$	CPU
C101_5x4	25	4	271.70	0.00	4.00	0.16	271.70	0.00	1.63	272.96	−0.46	−0.46	0.00	4.00	0.11
C201_5x4	25	2	863.08	3.00	2.00	0.09	863.08	3.00	2.45	863.08	0.00	0.00	3.00	2.00	0.09
C203_5x4	25	2	835.83	3.00	1.00	214.45	835.83	3.00	0.00	835.83	0.00	0.00	3.00	1.00	0.15
R101_5x4	25	4	2195.04	9.00	4.00	0.34	2195.04	9.00	1.26	2195.04	0.00	0.00	9.00	4.00	0.22
R201_5x4	25	2	1091.07	3.00	2.00	0.86	1092.41	3.00	1.45	**1091.07**	0.00	0.12	3.00	2.00	0.03
RC101_5x4	25	4	862.21	2.00	4.00	13.51	950.81	2.40	1.41	**862.21**	0.00	9.32	2.00	4.00	0.37
RC201_5x4	25	3	465.25	0.00	3.00	1.12	465.25	0.00	1.71	465.31	−0.01	−0.01	0.00	3.00	0.06
C101_6x6	25	4	927.35	3.00	3.00	0.14	927.35	3.00	1.77	927.35	0.00	0.00	3.00	3.00	0.09
C201_6x6	25	2	1217.10	4.00	1.00	0.13	1217.10	4.00	4.56	1217.10	0.00	0.00	4.00	1.00	0.01
C203_6x6	25	2	930.60	3.00	1.00	5.73	930.60	3.00	1.86	930.60	0.00	0.00	3.00	1.00	0.03
R101_6x6	25	4	2857.05	12.00	4.00	0.17	2977.63	12.60	1.28	**2868.19**	−0.39	3.68	12.00	4.00	0.30
R201_6x6	25	2	1377.42	4.00	1.00	0.53	1377.42	4.00	1.76	1422.57	−3.28	−3.28	4.00	2.00	0.05
RC101_6x6	25	4	1361.80	4.00	4.00	1.95	1557.44	5.00	1.47	**1361.80**	0.00	12.56	4.00	4.00	0.23
RC201_6x6	25	3	1228.89	3.00	2.00	22.79	1228.89	3.00	1.40	1228.89	0.00	0.00	3.00	2.00	0.14
C101_7x4	25	4	789.08	2.00	4.00	0.16	789.08	2.00	1.86	789.08	0.00	0.00	2.00	4.00	0.06
C103_7x4	25	4	671.06	2.00	3.00	3993.12	671.06	2.00	2.03	671.06	0.00	0.00	2.00	3.00	0.11
C201_7x4	25	2	738.35	2.00	2.00	0.05	738.35	2.00	1.60	738.35	0.00	0.00	2.00	2.00	0.02
C203_7x4	25	2	684.98	2.00	2.00	190.13	684.98	2.00	1.86	684.98	0.00	0.00	2.00	2.00	0.03
R101_7x4	25	4	2447.74	10.00	4.00	0.09	2447.74	10.00	1.27	2447.74	0.00	0.00	10.00	4.00	0.12
R201_7x4	25	2	959.51	2.00	2.00	0.19	959.51	2.00	1.49	959.51	0.00	0.00	2.00	2.00	0.07
R203_7x4	25	2	849.47	2.00	2.00	496.32	849.47	2.00	1.88	849.47	0.00	0.00	2.00	2.00	0.03
RC101_7x4	25	4	1669.63	6.00	4.00	0.50	1669.63	6.00	1.36	1669.63	0.00	0.00	6.00	4.00	0.09
RC201_7x4	25	3	967.60	2.00	3.00	5.46	967.60	2.00	2.00	967.60	0.00	0.00	2.00	3.00	0.08
Average			1141.82	3.61	2.70	215.13	1159.48	3.70	1.71	1144.32	−0.18	0.95	3.61	2.74	0.11

Table 4.3: Comparison of exact, ALNS and ILS solutions on small instances with 25 tasks

			CPLEX				ALNS			ILS					
Instance	$\lvert C\rvert$	$\lvert K\rvert$	Opt.	$\lvert C_o\rvert$	$\lvert K^*\rvert$	CPU	Avg.	$\lvert C_o\rvert$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\text{Imp}^*_{\text{A/I}}$	$\lvert C_o\rvert$	$\lvert K^*\rvert$	CPU
C101_5x4	50	6	830.00	1.00	6.00	2.61	838.11	1.00	6.98	**830.00**	0.00	0.97	1.00	6.00	1.09
C201_5x4	50	4	859.54	1.00	4.00	0.59	859.54	1.00	7.60	859.54	0.00	0.00	1.00	4.00	0.78
R101_5x4	50	6	4507.87	19.00	6.00	1.20	4540.34	19.20	7.22	**4511.36**	−0.08	0.64	19.00	6.00	6.84
R201_5x4	50	4	1107.51	1.00	4.00	107.52	1107.51	1.00	7.49	1112.25	−0.43	−0.43	1.00	4.00	2.13
C101_6x6	50	6	1154.84	3.00	5.00	251.94	1181.31	3.00	7.68	**1154.84**	0.00	2.24	3.00	5.00	1.95
C201_6x6	50	4	1203.93	3.00	3.00	0.42	1203.93	3.00	8.15	1203.93	0.00	0.00	3.00	3.00	0.66
R101_6x6	50	6	5190.35	22.00	6.00	1.22	5362.77	23.00	6.13	**5190.32**	0.00	3.22	22.00	6.00	2.62
R201_6x6	50	4	1647.70	3.00	3.00	794.06	1647.70	3.00	8.27	1649.95	−0.17	−0.18	3.00	3.00	1.99
C101_7x4	50	6	1356.54	3.00	6.00	7.61	1367.75	3.00	7.71	1367.75	−0.83	0.00	3.00	6.00	1.26
C201_7x4	50	4	1312.21	3.00	3.00	0.42	1312.21	3.00	7.87	1312.21	0.00	0.00	3.00	3.00	0.37
R101_7x4	50	6	4463.80	18.00	6.00	1.15	4540.34	18.60	7.22	**4469.31**	−0.12	1.56	18.00	6.00	2.01
R201_7x4	50	4	1553.23	3.00	4.00	129.01	1553.23	3.00	7.19	1553.23	0.00	0.00	3.00	4.00	0.93
Average			2098.91	6.67	4.67	108.15	2126.17	6.82	7.46	2101.22	−0.14	0.67	6.67	4.67	1.89

Table 4.4: Comparison of exact, ALNS and ILS solutions on medium instances with 50 tasks

		CPLEX			ALNS						ILS								
Instance	$\|K\|$	Best	Gap	CPU	Avg.	Min.	Max.	$\|C_o\|$	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	Min.	$\text{Imp}^-_{\text{A/I}}$	Max.	$\text{Imp}^+_{\text{A/I}}$	$\|C_o\|$	$\|K^*\|$	CPU
C101_5x4	8	5857.35	17.54	7200.00	5733.75	5656.63	5806.55	23.40	8.00	34.90	**5691.28**	0.74	**5589.76**	1.18	**5780.75**	0.44	22.80	8.00	33.65
C103_5x4	8	7874.13	92.02	7200.00	2782.20	2644.65	2869.64	7.60	8.00	49.49	2830.77	−1.75	2650.69	−0.23	2921.15	−1.79	7.80	8.00	38.05
C201_5x4	4	2755.52	0.00	14.40	2755.52	2755.52	2755.52	6.00	4.00	35.84	2781.37	−0.94	2755.52	0.00	2800.21	−1.62	6.00	4.00	11.40
C203_5x4	4	3630.92	46.67	7200.00	2392.50	2389.37	2393.62	6.00	4.00	70.46	2393.88	−0.06	**2383.01**	0.27	2407.41	−0.58	6.00	4.00	15.89
R101_5x4	12	5446.89	0.01	4229.83	5895.38	5582.58	6181.52	22.60	12.00	54.30	**5572.16**	5.48	**5561.83**	0.37	**5590.55**	9.56	21.00	12.00	62.46
R103_5x4	12	4856.22	81.38	7200.00	1845.25	1710.25	2020.48	2.00	12.00	60.09	**1765.16**	4.34	**1658.23**	3.04	**1943.30**	3.82	1.60	12.00	63.82
R201_5x4	4	3295.67	23.10	7200.00	2854.30	2838.50	2865.75	6.00	4.00	39.27	2866.39	−0.42	2839.54	−0.04	2887.93	−0.77	6.00	4.00	14.14
R203_5x4	4	4042.66	50.47	7200.00	2332.23	2332.23	2332.23	6.00	4.00	68.40	2349.24	−0.73	2335.76	−0.15	2360.57	−1.22	6.00	4.00	12.86
RC101_5x4	11	5441.33	61.97	7200.00	5164.84	5127.79	5262.36	18.20	11.00	50.49	**4983.98**	3.50	**4909.89**	4.25	**5093.47**	3.21	17.40	11.00	46.74
RC103_5x4	11	6142.71	87.10	7200.00	2348.06	2170.57	2490.12	4.20	11.00	57.80	2421.65	−3.13	2301.86	−6.05	2499.04	−0.36	4.60	11.00	39.24
RC201_5x4	5	3579.44	25.17	7200.00	3091.67	3088.23	3093.56	6.00	5.00	40.73	**3090.03**	0.05	**3083.49**	0.15	3100.00	-0.21	6.00	5.00	15.79
RC203_5x4	5	3579.20	43.86	7200.00	2540.35	2516.16	2550.62	6.00	5.00	67.95	**2530.63**	0.38	**2512.64**	0.14	2568.56	-0.70	6.00	4.80	12.54
C101_6x6	8	7660.86	9.61	7200.00	7762.94	7731.07	7791.08	32.00	8.00	36.72	**7695.83**	0.86	**7660.86**	0.91	**7783.33**	0.10	31.80	8.00	30.17
C103_6x6	8	9050.71	91.86	7200.00	5028.83	4980.70	5136.21	18.20	8.00	55.31	5066.61	−0.75	**4971.66**	0.18	5195.96	−1.16	18.40	8.00	37.08
C201_6x6	4	3315.30	6.02	7200.00	3299.56	3278.07	3328.01	9.60	3.40	35.35	3313.45	−0.42	3298.68	−0.63	3331.26	−0.10	9.20	3.80	21.10
C203_6x6	4	5443.79	63.09	7200.00	2465.90	2460.17	2468.71	6.00	3.00	67.12	2479.44	−0.55	2468.53	−0.34	2484.72	−0.65	6.00	3.00	23.56
R101_6x6	13	5944.91	0.01	2038.29	6152.29	5955.17	6322.82	23.00	13.00	62.19	**6005.32**	2.39	**5948.71**	0.11	**6083.98**	3.78	22.20	13.00	56.05
R103_6x6	13	8799.51	85.42	7200.00	2329.25	2251.64	2404.57	4.40	12.00	72.51	**2290.01**	1.68	**2225.67**	1.15	**2383.03**	0.90	4.20	12.20	61.65
R201_6x6	4	5169.21	48.79	7200.00	3536.70	3503.40	3574.97	8.60	4.00	38.66	3574.46	−1.07	3510.36	−0.20	3633.35	−1.63	8.80	4.00	32.37
R203_6x6	4	5595.34	62.06	7200.00	2446.18	2437.28	2481.77	6.00	3.00	71.59	2462.68	−0.67	2443.97	−0.27	2504.36	−0.91	6.00	3.00	23.00
RC101_6x6	12	7621.03	70.53	7200.00	5466.44	5276.34	5771.99	18.60	12.00	57.54	**5029.94**	7.99	**4975.34**	5.70	**5142.08**	10.91	16.40	12.00	43.80
RC103_6x6	12	6878.60	85.93	7200.00	2349.57	2263.83	2522.71	3.20	12.00	64.98	**2257.78**	3.91	**2113.03**	6.66	**2337.45**	7.34	2.80	12.00	45.74
RC201_6x6	4	7017.96	60.69	7200.00	4519.95	4422.86	4656.79	12.60	4.00	33.34	4550.99	−0.69	4490.33	−1.53	**4608.61**	1.03	12.60	4.00	31.23
RC203_6x6	4	5683.37	62.73	7200.00	2673.72	2649.51	2730.78	6.00	3.00	62.53	2686.83	−0.49	2671.23	−0.82	**2719.03**	0.43	6.00	3.00	19.58
C101_7x4	9	5208.99	9.11	7200.00	5257.90	5208.30	5307.12	19.00	9.00	35.89	5284.48	−0.51	5246.13	−0.73	5360.98	−1.01	19.00	9.00	19.88
C103_7x4	9	6657.08	86.65	7200.00	2117.44	2020.40	2173.38	2.60	9.00	49.85	**2059.98**	2.71	**2009.86**	0.52	**2163.05**	0.48	2.00	9.00	24.70
C201_7x4	4	2773.41	0.00	97.79	2779.37	2773.41	2803.21	5.00	4.00	32.51	2808.29	−1.04	2781.07	−0.28	2830.03	−0.96	5.00	4.00	8.11
C203_7x4	4	3048.56	41.76	7200.00	2282.15	2261.37	2301.73	5.00	4.00	61.64	2297.16	−0.66	2262.00	−0.03	2366.11	−2.80	5.00	4.00	9.98
R101_7x4	14	5079.67	0.00	1798.64	5381.35	5239.81	5437.66	18.80	14.00	55.20	**5238.11**	2.66	**5127.29**	2.15	**5283.80**	2.83	17.80	14.00	33.24
R103_7x4	14	5603.74	82.34	7200.00	2215.84	2104.92	2314.30	3.40	14.00	63.37	2222.76	−0.31	2139.77	−1.66	2333.64	−0.84	3.40	14.00	33.70
R201_7x4	5	2790.76	13.57	7200.00	2679.38	2672.96	2682.23	5.00	5.00	41.29	**2678.32**	0.04	**2664.93**	0.30	2693.43	−0.42	5.00	5.00	9.51
R203_7x4	5	3209.88	43.55	7200.00	2209.80	2199.10	2229.67	5.00	5.00	66.75	2223.96	−0.64	2209.32	−0.46	2242.78	−0.59	5.00	5.00	10.09
RC101_7x4	12	5692.20	36.43	7200.00	5799.77	5531.06	6367.47	20.40	12.00	52.57	**5440.59**	6.19	**5373.05**	2.86	**5556.76**	12.73	18.40	12.00	29.21
RC103_7x4	12	7660.38	88.19	7200.00	2674.54	2586.03	2820.48	5.00	12.00	57.89	**2615.16**	2.22	2591.39	−0.21	**2653.73**	5.91	5.00	12.00	24.52
RC201_7x4	5	3278.57	25.12	7200.00	2936.28	2919.83	2945.46	5.00	5.00	40.06	**2934.44**	0.06	**2910.73**	0.31	2948.77	−0.11	5.00	5.00	9.44
RC203_7x4	5	3739.80	51.34	7200.00	2285.17	2277.62	2301.26	5.00	5.00	58.82	2308.80	−1.03	2305.62	−1.23	2314.51	−0.58	5.00	4.40	10.14
Average		5261.82	45.95	6427.19	3510.73	3439.37	3597.12	10.04	7.54	52.87	3466.72	0.82	3416.16	0.43	3525.21	1.24	9.76	7.53	28.18

Table 4.5: Comparison of the ILS and the ALNS on benchmark instances with 100 tasks and limited technicians

		CPLEX			ALNS					ILS							
Instance	$\|K\|$	Best	Gap	CPU	Avg.	Min.	Max.	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	Min.	$\text{Imp}^-_{\text{A/I}}$	Max.	$\text{Imp}^+_{\text{A/I}}$	$\|K^*\|$	CPU
C101_5x4	17	1096.85	0.00	2640.69	1111.08	1098.71	1128.02	13.00	66.50	**1107.76**	0.30	**1097.67**	0.09	**1117.56**	0.93	12.60	33.70
C103_5x4	17	2604.28	81.10	7200.00	1037.33	1018.61	1049.41	11.60	84.61	**1026.51**	1.04	**1012.86**	0.56	**1045.80**	0.34	11.00	38.63
C201_5x4	8	1157.56	0.00	9.44	1180.93	1158.97	1228.88	7.40	54.29	**1157.56**	1.98	**1157.56**	0.12	**1157.56**	5.80	7.00	9.21
C203_5x4	8	2449.89	72.07	7200.00	1049.30	1046.93	1052.83	5.00	89.78	1059.58	−0.98	1054.21	−0.70	1068.72	−1.51	5.20	21.17
R101_5x4	25	1652.13	0.00	6093.49	1685.85	1687.68	1697.20	20.00	84.25	**1668.00**	1.06	**1658.93**	1.70	**1676.64**	1.21	20.40	43.66
R103_5x4	25	2604.28	81.10	7200.00	1249.91	1238.67	1282.28	14.80	101.77	**1243.54**	0.51	**1235.05**	0.29	**1253.46**	2.25	15.20	58.66
R201_5x4	7	1569.42	18.54	7200.00	1448.93	1440.30	1462.62	7.00	57.22	**1436.37**	0.87	**1431.16**	0.63	**1447.84**	1.01	6.60	27.61
R203_5x4	7	2477.25	68.27	7200.00	1106.12	1098.00	1123.08	5.80	84.18	**1100.75**	0.49	**1097.55**	0.04	**1105.09**	1.60	6.00	23.91
RC101_5x4	22	2503.98	42.91	7200.00	1716.07	1702.51	1729.75	16.40	78.25	**1695.67**	1.19	**1673.94**	1.68	**1710.32**	1.12	16.80	51.32
RC103_5x4	22	6810.66	89.74	7200.00	1354.11	1337.99	1388.13	12.40	91.41	1355.40	−0.09	**1321.66**	1.22	**1383.61**	0.33	12.20	59.19
RC201_5x4	9	1849.86	24.94	7200.00	1607.25	1601.89	1610.75	8.60	58.71	**1606.08**	0.07	**1589.24**	0.79	1620.59	−0.61	8.00	27.02
RC203_5x4	9	2724.35	72.45	7200.00	1166.50	1161.53	1178.64	6.00	87.01	**1165.81**	0.06	1162.95	−0.12	**1169.24**	0.80	6.00	24.84
C101_6x6	16	972.89	0.00	2044.98	1004.82	989.21	1028.72	11.40	69.76	**988.43**	1.63	**972.89**	1.65	**1001.62**	2.63	11.40	45.79
C103_6x6	16	3897.18	88.27	7200.00	897.86	893.94	907.65	10.40	88.17	911.62	−1.53	900.82	−0.77	933.90	−2.89	10.40	53.15
C201_6x6	7	821.55	0.00	664.21	821.55	821.55	821.55	4.00	55.65	826.42	−0.59	**821.55**	0.00	832.56	−1.34	4.60	42.76
C203_6x6	7	2164.58	74.26	7200.00	703.10	689.60	750.12	4.00	99.75	**693.50**	1.36	**689.60**	0.00	**699.19**	6.79	4.00	44.15
R101_6x6	26	1648.27	0.00	7100.00	1667.43	1658.27	1672.57	19.80	96.78	**1660.15**	0.44	**1657.55**	0.04	**1664.28**	0.50	19.80	67.27
R103_6x6	26	15070.30	94.47	7200.00	1231.49	1223.63	1243.49	14.00	116.00	**1225.80**	0.46	**1216.08**	0.62	**1238.01**	0.44	14.40	73.23
R201_6x6	7	1462.90	23.69	7200.00	1270.26	1261.94	1279.81	6.00	62.02	**1270.25**	0.00	1265.56	−0.29	**1278.94**	0.07	5.80	55.72
R203_6x6	7	3329.47	79.29	7200.00	951.84	932.35	964.54	5.40	99.46	**933.41**	1.94	**930.74**	0.17	**940.98**	2.44	4.00	56.79
RC101_6x6	24	2317.68	40.98	7200.00	1683.96	1679.13	1690.06	15.60	90.42	**1674.61**	0.56	**1663.30**	0.94	**1682.62**	0.44	16.00	61.94
RC103_6x6	24	5614.42	87.81	7200.00	1310.95	1281.55	1331.41	11.80	103.68	**1309.18**	0.14	1297.09	−1.21	**1331.28**	0.01	12.00	86.38
RC201_6x6	8	1849.86	24.94	7200.00	1406.95	1395.40	1411.48	6.40	61.32	**1380.38**	1.89	**1367.89**	1.97	**1394.90**	1.17	6.40	52.16
RC203_6x6	8	2337.76	72.19	7200.00	1016.71	1001.04	1030.15	5.40	91.39	**1014.51**	0.22	1003.81	−0.28	**1024.80**	0.52	5.40	49.44
C101_7x4	17	1357.05	0.00	281.50	1398.95	1357.05	1462.16	15.20	58.65	**1370.78**	2.01	**1357.05**	0.00	**1381.59**	5.51	14.80	22.31
C103_7x4	17	3506.12	82.13	7200.00	1239.22	1215.70	1264.17	13.20	75.26	**1233.60**	0.45	1220.19	−0.37	**1256.67**	0.59	13.20	27.96
C201_7x4	8	1256.30	0.00	2.78	1282.18	1256.30	1302.56	8.00	49.54	**1263.00**	1.50	**1256.30**	0.00	**1289.68**	0.99	8.00	10.56
C203_7x4	8	2401.88	67.78	7200.00	1151.27	1150.85	1152.94	7.80	76.83	**1145.36**	0.51	**1137.07**	1.20	**1150.85**	0.18	7.60	17.89
R101_7x4	28	1764.78	0.00	2875.72	1793.95	1776.46	1813.53	21.40	88.52	**1787.22**	0.38	1781.13	−0.26	**1796.48**	0.94	22.00	39.28
R103_7x4	28	7412.24	87.71	7200.00	1375.09	1346.80	1399.95	16.20	102.08	**1349.32**	1.87	**1337.92**	0.66	**1373.27**	1.91	16.40	43.54
R201_7x4	10	1411.36	5.83	7200.00	1410.90	1398.14	1427.95	9.20	59.31	**1406.55**	0.31	1401.68	−0.25	**1412.46**	1.08	9.40	19.36
R203_7x4	10	2476.19	67.43	7200.00	1166.94	1164.90	1169.27	8.00	80.29	**1164.89**	0.18	**1160.51**	0.38	1170.12	−0.07	8.00	18.45
RC101_7x4	23	1902.94	17.32	7200.00	1844.37	1821.90	1859.17	17.80	75.55	**1822.30**	1.20	**1805.39**	0.91	**1837.41**	1.17	17.60	36.51
RC103_7x4	23	3367.93	77.17	7200.00	1455.33	1435.63	1477.84	13.40	85.04	**1436.23**	1.31	**1427.40**	0.57	**1450.97**	1.82	13.60	38.30
RC201_7x4	9	1823.30	17.64	7200.00	1701.25	1697.82	1705.48	9.00	52.78	1706.24	−0.29	**1697.82**	0.00	1727.00	−1.26	9.00	15.34
RC203_7x4	9	2070.17	60.02	7200.00	1241.65	1239.45	1249.72	7.20	73.82	**1235.76**	0.47	**1230.63**	0.71	**1238.74**	0.88	8.00	16.81
Avgerage		2826.05	45.00	6003.13	1298.37	1285.57	1315.22	10.79	79.17	1289.79	0.64	1280.35	0.35	1301.80	1.05	10.80	39.28

Table 4.6: Comparison of exact, ALNS and ILS solutions on benchmark instances with 100 tasks and unlimited technicians

4.5.6 Skill VRP Instances

The proposed ILS algorithm is also applied to solve a set of Skill VRP instances, which are generated based on the benchmark instances of Solomon (1987) and the skill pattern introduced by Cappanera et al. (2011). As the Skill VRP does not involve time window and capacity constraints, we use only the geographical information of Solomon's instances to generate three types of geographical data for Skill VRP instances, namely, R, C and RC, which represent the random, clustered and a mixed of random and clustered geographical setting, respectively. Similar to Cappanera et al. (2011) and Schwarze and Voß (2012), we consider a skill set with three levels 1, 2 and 3, where skill 1 denotes the lowest level, and skill 3 the highest. Each task $i \in C$ is associated with a skill requirement $s_i \in \{1, 2, 3\}$, which must be fulfilled by a technician $k \in K$ having a skill level $\hat{s}_k \geq s_i$, where $\hat{s}_k \in \{1, 2, 3\}$. The skill data is randomly generated according to the four patterns introduced by Cappanera et al. (2011), as given in the Table 4.7, where each row of values represent a pattern that indicates the distribution of skill requirements over tasks. For example, the first pattern $\{50, 10, 40\}$ indicates that a task i has a skill requirement $s_i = 1$ with probability 0.5, $s_i = 2$ with probability 0.1 and $s_i = 3$ with probability 0.4. For each combination of skill pattern and geographical data, we generated three random instances, which results in a total of 36 instances. All the instances have two sizes, where one has 20 tasks and the other has 30 tasks. Each instance has a set of three technicians $K = \{1, 2, 3\}$, where each technician $k \in K$ is specialised at a different skill level $\hat{s}_k \in S$; for example, $\hat{s}_1 = 1, \hat{s}_2 = 2$ and $\hat{s}_3 = 3$.

	Skill		
Pattern	1 (%)	2 (%)	3 (%)
1	50	10	40
2	50	20	30
3	40	40	20
4	30	30	40

Table 4.7: Distribution of skill requirements over tasks

In the Skill VRP, the routing costs depend on both the traveling distance and the technician, such that the increasing skill level of the technician causes increasing costs. Thus, for each arc $(i, j) \in A$ and each technician $k \in K$, we follow the approach of Schwarze and Voß (2012) by defining a skill dependent routing cost c_{ij}^k by

$$c_{ij}^k = c_{ij} \theta \hat{s}_k, \tag{4.16}$$

where c_{ij} is the traveling distance of arc $(i, j) \in A$, and θ is a weight parameter of the skill level $\hat{s}_k$ of the technician $k \in K$. Following the suggestion of Schwarze and Voß (2012), we set $\theta = 1$ in our experiments.

4.5.7 Results for Skill VRP Instances

The above Skill VRP instances are solved by using our ILS, and the results are compared with the solutions obtained from a basic MIP model of Cappanera et al. (2011) that is solved by using CPLEX 12.6. A time limit of 7200 seconds is imposed on CPLEX, and for instances not solved to optimality, we report the best values of the solutions found within this time limit.

Table 4.8 presents results of the experiments for instances with 20 tasks. Of the 36 instances tested, CPLEX finds optimal solutions for 27 and exceeds the time limit for 9 instances. The solutions produced by our ILS algorithm are exactly the same as the optimal or best solutions found by CPLEX for all instances. The average computational time of our ILS is 0.08 seconds, which is negligible compared to the time used by CPLEX.

Table 4.9 shows results of the experiments on instances with 30 tasks. For this size of instances, CPLEX is only able to find optimal solutions for 10 out of 36 instances. Among these 10 instances, our ILS can produce optimal solutions for 9, with the exception being instance R_4_1 for which our ILS found slightly worse solutions that have an average gap of -0.66% to that of CPLEX. Of the remaining instances that are not solved to optimality by CPLEX, our ILS produces better solutions for 6 and equal cost solutions for 20 compared to the best solutions found by CPLEX within the time limit. The average percentage difference between the values of our ILS solutions and CPLEX solutions is 0.74%. The average computational time required by our ILS is 0.48 seconds, which is also negligible compared to the time used by CPLEX.

4.6 Conclusion

This paper presents an iterated local search (ILS) algorithm for solving the workforce scheduling and routing problem (WSRP). We have examined different combinations of neighbourhood structures, and results show that the strategy of apply the intra-route search as a post-optimization procedure for the inter-route search provides effective and efficient performance. The proposed ILS is evaluated against a mixed integer programming (MIP) model and an adaptive larger neighbourhood search (ALNS) algorithm (Kovacs et al., 2012) on benchmark instances with up to 100 tasks. Computational experiments indicate that the proposed algorithm can produce solutions that are within an average gap of 1% to the optimal values in at most 40 seconds on average for all instances tested here. Compared to other heuristic approaches (Kovacs et al., 2012; Castillo-Salazar et al., 2015) for the similar problems, the proposed ILS has a relatively simple structure and a small number of parameters.

	CPLEX				ILS			
Instance	Best	Gap	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\|K^*\|$	CPU
C_1_1	370.93	0.00	1.00	122.82	370.93	0.00	1.00	0.07
C_1_2	367.55	4.79	2.00	7200.00	367.55	0.00	2.00	0.12
C_1_3	367.55	8.85	2.00	7200.00	367.55	0.00	2.00	0.09
C_2_1	370.93	0.00	1.00	353.27	370.93	0.00	1.00	0.06
C_2_2	367.55	11.36	2.00	7200.00	367.55	0.00	2.00	0.17
C_2_3	367.55	9.63	2.00	7200.00	367.55	0.00	2.00	0.08
C_3_1	370.93	0.00	1.00	239.05	370.93	0.00	1.00	0.06
C_3_2	367.55	4.38	2.00	7200.00	367.55	0.00	2.00	0.15
C_3_3	367.55	0.00	2.00	2742.05	367.55	0.00	2.00	0.17
C_4_1	370.93	0.00	1.00	28.27	370.93	0.00	1.00	0.09
C_4_2	367.55	0.00	2.00	1307.39	367.55	0.00	2.00	0.13
C_4_3	370.93	0.00	1.00	1648.17	370.93	0.00	1.00	0.06
R_1_1	781.09	0.00	2.00	8.78	781.09	0.00	2.00	0.07
R_1_2	772.50	0.00	2.00	73.35	772.50	0.00	2.00	0.10
R_1_3	710.16	0.00	2.00	18.66	710.16	0.00	2.00	0.04
R_2_1	781.09	0.00	2.00	33.40	781.09	0.00	2.00	0.11
R_2_2	729.12	0.00	2.00	76.47	729.12	0.00	2.00	0.06
R_2_3	710.16	0.00	2.00	68.49	710.16	0.00	2.00	0.09
R_3_1	777.40	0.00	3.00	191.36	777.40	0.00	3.00	0.04
R_3_2	702.26	0.00	2.00	143.98	702.26	0.00	2.00	0.09
R_3_3	748.90	0.00	2.00	323.30	748.90	0.00	2.00	0.14
R_4_1	787.01	0.00	1.00	6.76	787.01	0.00	1.00	0.08
R_4_2	787.01	0.00	1.00	15.46	787.01	0.00	1.00	0.09
R_4_3	755.20	0.00	2.00	56.62	755.20	0.00	2.00	0.07
RC_1_1	658.21	0.00	1.00	27.38	658.21	0.00	1.00	0.05
RC_1_2	658.21	1.89	1.00	7200.00	658.21	0.00	1.00	0.03
RC_1_3	570.00	0.00	2.00	2034.46	570.00	0.00	2.00	0.03
RC_2_1	658.21	0.00	1.00	83.90	658.21	0.00	1.00	0.03
RC_2_2	658.21	8.77	1.00	7200.00	658.21	0.00	1.00	0.04
RC_2_3	570.00	4.21	2.00	7200.00	570.00	0.00	2.00	0.03
RC_3_1	658.21	0.00	1.00	786.79	658.21	0.00	1.00	0.06
RC_3_2	658.21	13.29	1.00	7200.00	658.21	0.00	1.00	0.04
RC_3_3	570.00	0.00	2.00	893.34	570.00	0.00	2.00	0.03
RC_4_1	658.21	0.00	1.00	26.89	658.21	0.00	1.00	0.05
RC_4_2	658.21	0.00	1.00	3180.79	658.21	0.00	1.00	0.03
RC_4_3	658.21	0.00	1.00	3042.84	658.21	0.00	1.00	0.03
Average	586.20	1.87	1.58	2287.06	586.20	0.00	1.58	0.08

Table 4.8: The comparison of exact and ILS solutions on Skill VRP instances with 20 tasks

	CPLEX				ILS			
Instance	Best	Gap	$\lvert K^*\rvert$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\lvert K^*\rvert$	CPU
C_1_1	439.45	2.87	1.00	7200.00	439.45	0.00	1.00	0.33
C_1_2	432.86	25.93	1.00	7200.00	432.86	0.00	2.00	0.11
C_1_3	429.52	31.50	2.00	7200.00	429.52	0.00	2.00	0.37
C_2_1	439.45	11.92	1.00	7200.00	439.45	0.00	1.00	0.40
C_2_2	432.86	26.22	2.00	7200.00	432.86	0.00	2.00	0.12
C_2_3	429.52	32.25	2.00	7200.00	429.52	0.00	2.00	0.39
C_3_1	439.45	23.09	2.00	7200.00	439.45	0.00	1.00	0.41
C_3_2	440.30	28.08	2.00	7200.00	439.45	0.19	1.00	0.42
C_3_3	449.16	32.15	2.00	7200.00	439.45	2.16	1.00	0.49
C_4_1	439.45	0.00	1.00	530.36	439.45	0.00	1.00	0.23
C_4_2	475.39	28.19	2.00	7200.00	439.45	7.56	1.00	0.51
C_4_3	505.33	32.30	2.00	7200.00	439.46	13.03	1.00	0.56
R_1_1	956.66	0.00	2.00	122.79	956.66	0.00	2.00	0.57
R_1_2	912.68	0.00	2.00	1319.63	912.68	0.00	2.00	0.60
R_1_3	812.97	0.00	2.00	797.14	812.97	0.00	2.00	0.56
R_2_1	956.66	0.00	2.00	766.26	956.66	0.00	2.00	0.67
R_2_2	921.17	7.18	3.00	7200.00	912.68	0.92	2.00	0.71
R_2_3	812.97	0.00	2.00	6507.53	812.97	0.00	2.00	0.57
R_3_1	964.61	0.28	3.00	7200.00	964.61	0.00	3.00	0.82
R_3_2	896.06	4.19	2.00	7200.00	896.06	0.00	2.00	0.71
R_3_3	890.27	4.01	2.00	7200.00	890.27	0.00	2.00	0.71
R_4_1	973.72	0.00	2.00	25.01	980.11	−0.66	2.00	0.74
R_4_2	981.64	0.00	1.00	3549.01	981.64	0.00	1.00	0.43
R_4_3	919.09	0.00	2.00	7036.77	919.09	0.00	2.00	0.73
RC_1_1	928.31	6.57	1.00	7200.00	928.31	0.00	1.00	0.40
RC_1_2	928.31	37.44	1.00	7200.00	928.31	0.00	1.00	0.34
RC_1_3	773.66	30.98	2.00	7200.00	773.66	0.00	2.00	0.47
RC_2_1	928.31	23.40	1.00	7200.00	928.31	0.00	1.00	0.48
RC_2_2	928.31	21.32	2.00	7200.00	928.31	0.00	1.00	0.34
RC_2_3	773.66	34.10	2.00	7200.00	773.66	0.00	2.00	0.46
RC_3_1	920.74	25.82	2.00	7200.00	920.74	0.00	2.00	0.50
RC_3_2	928.31	39.12	1.00	7200.00	928.31	0.00	1.00	0.37
RC_3_3	842.32	33.62	2.00	7200.00	842.32	0.00	2.00	0.52
RC_4_1	928.31	0.00	1.00	1429.17	928.31	0.00	1.00	0.43
RC_4_2	928.31	17.15	1.00	7200.00	928.31	0.00	1.00	0.33
RC_4_3	953.07	26.58	2.00	7200.00	920.74	3.39	2.00	0.48
Average	753.14	16.29	1.75	5813.44	749.06	0.74	1.58	0.48

Table 4.9: The comparison of exact and ILS solutions on Skill VRP instances with 30 tasks

The proposed ILS algorithm is also applied to solve a set of Skill VRP instances, and results show that our algorithm is able to find optimal or near-optimal solutions in less than 0.5 seconds on average for all instances tested. Although the proposed algorithm is designed for solving the workforce scheduling and routing problem, it can be easily adapted to tackle other types of scheduling and routing problems.

Chapter 5

Incorporating Future Requests in the Dynamic Workforce Scheduling and Routing Problems

This chapter considers a dynamic workforce scheduling and routing problem (WSRP), where service requests arrive continuously and randomly over time, and operational decisions are made in an ongoing fashion. To address this problem, we describes a sampling-based model that incorporates stochastic knowledge about future requests. The proposed model uses a two-stage set-partitioning framework, where the first-stage is concerned with finding a set of feasible technician routes covering known requests, while the second-stage estimates the effect of the same routes with respect to future requests. The performance of the proposed model is evaluated against a deterministic model and a naive greedy heuristic within a simulation framework, and tested on realistic instances generated using probability distributions derived from historical data.

5.1 Introduction

The workforce scheduling and routing problem (WSRP) is commonly faced by many service providers, and arises in applications of home health care, field technician scheduling, security personnel routing and manpower allocation. In the classical setting of the WSRP, all parameters are assumed to be known before plans are made and that the designed routing plans are fixed during the execution stage. However, this is not necessarily the case in real-world applications where all or part of the input data are often incomplete or unknown during the design phase. In such situations, the challenge is to make operational decisions under uncertainties, which has given rise to stochastic or dynamic scheduling and routing problems. Existing studies in this area usually focus on either the stochastic version or the dynamic version exclusively (Bent and Van Hentenryck, 2004). Recent advances in computer and telecommunications technologies have enabled researchers to develop more effective solution tools to simultaneously address the stochastic and dynamic aspects of the problem. This paper aims to exploit stochastic information about future requests and integrate this part of information into the solution methods for the dynamic WSRP.

In this paper, we consider a dynamic WSRP, where service requests arise continuously and randomly over time, and operational decisions are made in an ongoing fashion as new requests arrive. This work is motivated by a British company providing emergency motoring breakdown services to its customers in the UK. A general description of the problem is given as follows. When customers have their vehicles break down, they send service requests to the call centre. The call handlers collect information from customers, and determine whether or not to accept the requests. If a request is accepted, it must be assigned to a technician with required skills and tools, and a service time window is assigned, within which a technician is expected to perform roadside assistance at the customer location. A late service is allowed at the expense of a penalty cost proportional to the amount of time delay. Since a technician may have been assigned several tasks, the system needs to determine the sequence of tasks to be carried out by each technician. If no sufficient resources are available to provide the required service within the time window or if better operational performance can be achieved, tasks can be outsourced (rejected) to a third party, albeit at the expense of additional costs. Our objective is to develop an algorithm that can effectively and efficiently address the task acceptance/rejection decisions as well as scheduling and routing decisions in real-time, while minimising the total response times (the time interval between the arrival time of a request and the time at which service begins), the penalty cost related to late services, and the cost of rejections.

The main contributions of this paper can be summarised as follows. First, we present a deterministic set-partitioning (DSP) model for the offline WSRP. Based on the structure of the problem considered, we introduce two dominance rules and a set of route size constraints to enhance the performance of the DSP model. Second, we integrate the stochastic information about future requests into the DSP model, which results in a stochastic set-partitioning (SSP) model. The proposed SSP model uses a two-stage framework, where the first-stage considers finding a set of feasible routes covering known tasks, while the second-stage evaluates the performance of the first-stage decisions with respect to future requests. The stochastic information on future requests is exploited in the form of sampled scenarios by the SSP model. Similar scenario-based approaches have been proposed by a number of studies for dynamic and stochastic vehicle routing problems (VRPs), which are closely related to the WSRP. However, due to the computational complexity of the sampling-based models, the existing studies have focused on developing heuristic solution algorithms. In this paper, we show that the proposed

SSP model can handle reasonable size instances in short computation times by exact methods. Lastly, we use a simulation framework to assess the performance of the DSP model, the SSP model and a naive greedy heuristic.

The rest of the paper is organised as follows. Section 5.2 provides a brief review of the stochastic and dynamic scheduling and routing problems, with a particular emphasis on the sampling-based algorithms. Section 5.3 formally defines the problem considered in this paper and presents a deterministic formulation for the offline version of the problem. Section 5.4 describes a sampling-based model that incorporates knowledge about future requests. Computational results are presented in Sections 5.5 and 5.6. The chapter ends with concluding remarks in Section 5.7.

5.2 Relevant Body of Literature

The WSRP is closely related to the vehicle routing problem (VRP). Since this paper considers the stochastic and dynamic version of the problem, our review predominately focuses on the papers published in the area of dynamic VRPs. In contrast to the static version, the dynamic VRPs, also referred as real-time or on-line VRPs, assume that part or all input data are revealed dynamically during the execution stage. In this context, scheduling and routing plans are designed in an ongoing fashion. The most common source of dynamism is the arrival of customer requests, but demands, travel times and service times are also possible dynamic elements. According to the classification of Pillac et al. (2013a), dynamic VRPs can be divided into two classes: deterministic and stochastic. The difference between two lies in the exploitation of probabilistic knowledge about stochastic parameters. In the stochastic and dynamic VRPs, the stochastic information on dynamically revealed parameters is investigated and used to support the construction of scheduling and routing plans, while in the deterministic version there is no stochastic element. For surveys on the deterministic and dynamic VRPs, interested readers are advised to Pillac et al. (2013a) and Bektaş et al. (2014). The following review discusses the models and algorithms proposed for addressing stochastic and dynamic VRPs. The solution methods developed in this area can be divided into three groups: anticipatory strategies, stochastic modelling approaches and sampling-based algorithms, which are described below.

5.2.1 Anticipatory Strategies

The most common anticipatory strategies proposed for dynamic VRPs are repositioning and waiting. The former relocates idle vehicles to strategic locations, where new requests are likely to appear. This type of strategies have been recognised as important elements for dynamic VRPs, especially for emergency vehicle dispatching problems (e.g., deployment of police, fire and ambulance services). Some of recent studies in this area include

Bent and Van Hentenryck (2007); Maxwell et al. (2010); Naoum-Sawaya and Elhedhli (2013) and Van Barneveld et al. (2016). Unlike repositioning strategies, which only consider idle vehicles, waiting strategies allow a vehicle to wait at a strategic location for a predefined amount of time before travelling to the destination already assigned to it. Prior studies, e.g., Ichoua et al. (2006); Branke et al. (2005); Thomas (2007); Bent and Van Hentenryck (2007) and Branchini et al. (2009), showed that waiting strategies are very useful for dynamic VRPs, especially for the problems with time windows, where time lags exist between requests.

5.2.2 Stochastic Modelling Approaches

Stochastic modelling approaches, like Markov Decision Process (MDP), have been applied by a number of studies to address dynamic VRPs. Thomas and White III (2004) consider a dynamic pick-up and delivery problem (DPDP), where a single vehicle with unlimited capacity travels from a known origin to a known destination and responds to potential requests at known locations along the route. The problem is modelled as a finite-horizon MDP, and an optimal routing policy is derived based on the structural results. The same problem is also studied by Thomas (2007), where waiting strategies regarding where and how long to wait are exploited in more detail. The problem is also modelled as a finite-horizon MDP. Based on structural results, a real-time heuristic is developed, and its effectiveness is demonstrated by comparing with five waiting heuristics. In Powell (1996), a hybrid model is built for a dynamic assignment problem, where vehicles are assigned to loads that arise randomly over time. The developed model replays on a hybrid network that combines known and forecasted demands. Experimental results confirm that taking forecasted demands into account is advantageous for making dynamic decisions. Yang et al. (2004) consider and compare five rolling horizon strategies for a DPDP. Three of the strategies are based on simple heuristic approaches, while the other two use a repeated re-optimisation procedure on mathematical models of the corresponding offline problem. One of the offline models incorporates some probabilistic knowledge about incoming requests, and this model shows the best performance in their computational experiments. Other studies in this area include Novoa and Storer (2009); Secomandi and Margot (2009) and Goodson et al. (2013).

5.2.3 Sampling-Based Algorithms

A sampling algorithm generates a number of scenarios at each decision epoch that describe possible realisations of future events, which are then evaluated in order to determine the best action. For example, Bent and Van Hentenryck (2004) use a multiple scenario approach (MSA) to solve a partially dynamic vehicle routing problem with time windows (VRPTW), where some customers are known in advance while others are dynamic. The objective is to maximize the number of serviced customers. The MSA populates and maintains a pool of routing plans, which are constructed for scenarios containing known and predicted customers. Once a route plan is obtained for a scenario, the predicted customers are removed. At each decision epoch, a plan is selected from the solution pool using a consensus function, in which new customers can be accommodated, since extra room has been reserved during the design phase. A similar problem is considered by Hvattum et al. (2006), where the objective is to minimize the number of vehicles used and the total travelling time. They also apply a scenario-based approach, namely a dynamic stochastic hedging heuristic (DSHH). Similar to the MSA, the DSHH incorporates stochastic knowledge by generating scenarios containing known and randomly sampled future customers. Each scenario is then solved as a static VRPTW to produce a routing plan. The main difference between the DSHH and the MSA is that, instead of choosing an executed plan from the solution pool, the DSHH combines common features of scenario solutions to build a unique solution. Based on the framework of DSHH, Hvattum et al. (2007) propose a branch-and-regret heuristic for a dynamic and stochastic VRP, where the goal is to minimize the number of unserviced customers, the number of vehicles used and the total distance travelled. Ghiani et al. (2012) compare the performance of a sampling-based algorithm and an anticipatory insertion heuristic proposed for a dynamic and stochastic travelling salesman problem, where a single, uncapacitated vehicle responds to a mix of advance and late customers at known locations. Their results indicate that the anticipatory insertion heuristic can match the solution quality of the sampling-based algorithm and use less computational effort, particularly when the degree of dynamism is low (e.g., low percentage of late customers). However, the sampling-based algorithm tends to perform better for instances with a high degree of dynamism.

Instead of solving scenarios to construct a routing plan, some studies apply scenario-based methods to estimate the influence of current decisions on future customers. Ghiani et al. (2009), for example, propose anticipatory insertion and local search algorithms to solve a dynamic VRP with pick-up and delivery (PDP), where the objective is to minimize the total expected customer inconvenience. To take future requests into account, they define a penalty function that estimates the expected cost of accommodating future request into a candidate solution. The expected value of the penalty function is approximated based on a set of scenarios containing only future requests. The candidate solution that gives the best objective value consisting of the cost of serving known requests and the penalty cost regarding future requests is selected as the executed plan. In Azi et al. (2012), a scenario-based approach is applied to assess the opportunity value of new requests for a dynamic VRP with multiple delivery routes to dynamically decide whether to accept new incoming requests. They define the opportunity value of a new request by the sum of the differences on solution values over a set of scenario solutions with and without the insertion of the new request. Each scenario solution is constructed for known requests and a scenario containing future requests by using an adaptive large neighbourhood search (ALNS) algorithm. Only the new request with positive opportunity value is accepted in the solution. Sarasola et al. (2016) develop a sampling-based variable neighbourhood search (VNS) to solve a VRP with stochastic demand and dynamic requests, where the goal is to minimize the total travelling cost and the delay penalty related to route duration constraints. The proposed heuristic applies a sampling-based method to evaluate and compare solutions. Tirado and Hvattum (2016) also apply sampling-based methods in the solution evaluation mechanisms of three local search heuristics developed for a dynamic and stochastic PDP. Their results show that the performance of three existing heuristics presented in Tirado et al. (2013), Bent and Van Hentenryck (2004) and Hvattum et al. (2007) can be improved by incorporating the proposed local search heuristics. Table 5.1 presents an overview of sampling-based algorithms proposed for dynamic and stochastic VRPs.

Author	Request Type	Time Windows	Route Duration	Objectives	Method Type
Bent and Van Hentenryck (2004)	Mixed	Hard	-	Max. number of serviced customers	Heuristic
Hvattum et al. (2006)	Mixed	Hard	-	Min. number of vehicles used & travel time	Heuristic
Hvattum et al. (2007)	Mixed	-	-	Min. number of unserviced customers & vehicles used & travel distance	Heuristic
Ghiani et al. (2009)	Dynamic	Soft	-	Min. expected customer inconvenience	Heuristic
Ghiani et al. (2012)	Mixed	-	-	Max. number of serviced customers	Heuristic
Azi et al. (2012)	Dynamic	-	Hard	Max. total profit (revenue minute travel distance)	Heuristic
Sarasola et al. (2016)	Mixed	-	Soft	Min. travel distance & time delay penalty	Heuristic
Tirado and Hvattum (2016)	Mixed	Hard	-	Min. travel cost & other operational cost	Heuristic

Table 5.1: Overview of sampling-based algorithms for dynamic and stochastic VRPs

The above overview shows that the majority of existing studies consider partially dynamic problems, where some requests are known in advance and others are dynamic (indicated as 'Mixed' in Table 5.1). Moreover, the objective functions are usually concerned with the number of serviced or unserviced customers, the number of vehicle used and various travelling related costs. Unlike these studies, the problem considered in our study has a high degree of dynamism, where only dynamic requests are considered and the service at customers must start as soon as possible. Our objective function minimizes the total response times, delay penalties and rejection costs. Although sampling-based algorithms are able to provide high quality solutions, the computational effort increases with the number of scenarios. Previous studies have focused on the integration of sampling-based approaches into heuristic algorithms. In contrast, we present a sampling-based model that can handle reasonable size instances of a real-life problem in short computational times without the need for using heuristic approaches.

5.3 The Dynamic WSRP: Definition and Formulation

The dynamic WSRP is defined on a planning horizon, within which service requests arrive randomly, and operational decisions are made whenever a new request arrives. We refer to the set of time points at which decisions are made by decision epochs, which correspond to the arrival times of requests. Each request represents a task i requiring a technician to carry out emergency assistance at a customer location, and it is characterized by a 6-tuple $(e_i, l_i, d_i, s_i, v_i, o_i)$ with the following definitions, all revealed at the time of receiving the request.

- $[e_i, l_i]$ is the time window within which the service at task i is expected to commence. As illustrated in Figure 5.1, the earliest time e_i is simply defined as the arrival time of request i and the latest time l_i is defined as $e_i + T^{Tar}$, where T^{Tar} is the target response time. A late service after l_i is allowed, but at a penalty cost proportional to the amount of delay. To ensure that all customers are served within reasonable amount of waiting time, we define U^{Max} as the maximum amount of time delay allowed at every task. As a results, each task effectively has a soft time window $[e_i, l_i]$ and a hard time window $[e_i, l_i + U^{Max}]$. The service at a given task must start within its hard time window and preferably within its soft time window. Similar definitions have been used by Fagerholt (2001) and Calvete et al. (2007);

- d_i and s_i are the service duration and the skill requirement respectively;

- v_i is the location of task i, defined as a point on a Euclidean plane, as are the vehicle positions, where the distance between two points is the Euclidean distance;

- o_i is the penalty cost of rejecting task i.

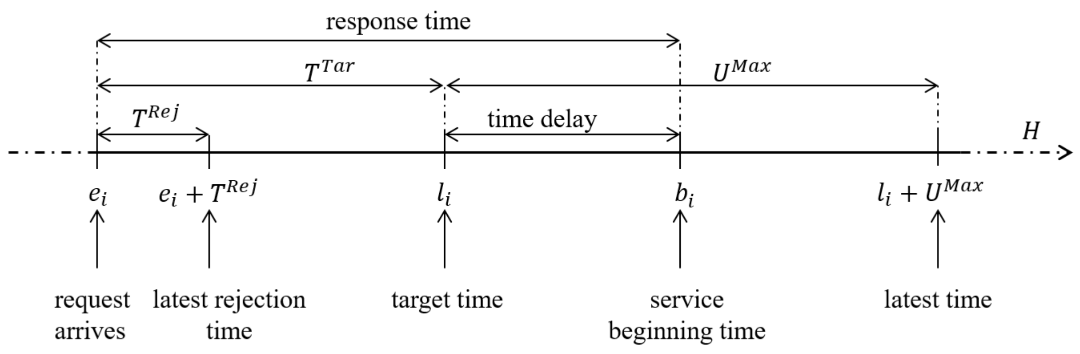


Figure 5.1: Timeline of a service request

Service requests can be either accepted or rejected, but the decisions must be made within T^{Rej} units of time. As shown in Figure 5.1, the epoch $e_i + T^{Rej}$ represents the deadline of making the final acceptance/rejection decision for task i. Before this deadline, task i can be either temporarily accepted or rejected. If a task is accepted, it will be inserted to the job plan of a technician. Once a technician starts travelling to a task, they cannot serve another task until the current one is completed. In this case, the task is defined as locked, which means that it cannot be reassigned to another technician. Technicians are allowed to swap unlocked tasks in their job plans. We use a parameter I_i^t to represent the state of a known task i at time t, defined as follows.

- $I_i^t = 0$: task i arrives and no decision has been made;
- $I_i^t = 1$: task i is temporarily rejected and the deadline for rejection has not passed;
- $I_i^t = 2$: task i is accepted but not locked, and the deadline for rejection has not passed;
- $I_i^t = 3$: task i is permanently rejected, and the decision cannot be changed;
- $I_i^t = 4$: task i is accepted but not locked, and the deadline for rejection has passed;
- $I_i^t = 5$: task i is accepted and locked;
- $I_i^t = 6$: task i is accepted and the service is completed by a technician.

At each time point, every known task is in one of the seven states defined above. Figure 5.2 shows the states and the state transitions that can occur from the time a task arrives until it is completed by a technician (state 6) or rejected (state 3). Tasks can be temporarily rejected if they are in state 0 or 2. For tasks that are in state 2 or 4, they are allowed to be swapped between technicians. Once a task is in state 5, the scheduling plan for this task is fixed, which must be carried out by the technician assigned.

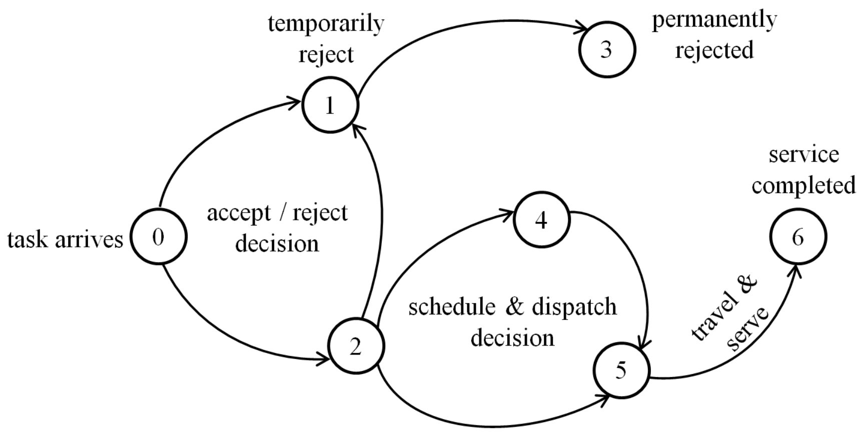


Figure 5.2: State diagram of a task

Our objective is to find a scheduling strategy that minimizes the total solution costs over the planning horizon, where the solution costs consist of (1) response times, (2) penalty for late service and (3) cost of rejecting requests. As illustrated in Figure 5.1, the response time to a task i is computed as $b_i - e_i$, where $b_i > e_i$ is a continuous variable that denotes the time at which service commences at task i. The amount of time delay is computed as $b_i - l_i$, if $b_i > l_i$, and 0 otherwise. Each unit of time delay is penalized by a factor β.

The dynamic WSRP defined above can be solved in a rolling horizon framework. At each decision epoch, a scheduling plan is generated based on the information up to that epoch. The designed plan is then carried out until the next decision epoch, at which the current plan is interrupted and a new plan is made. To generate a scheduling plan, we solve an offline WSRP, as defined in the following section.

5.3.1 Deterministic Formulation

The offline WSRP aims to find a scheduling and routing solution for a set K of technicians, to carry out a set of tasks at different locations, defined at each decision epoch of the dynamic WSRP. Let C be a set of tasks that are in state 0, 1, 2 or 4 at the epoch. Let C' be a subset of C, where each task $i \in C'$ is in state 4, indicating that the subset $C \setminus C'$ is the set of tasks that can be rejected. At a decision epoch, each technician $k \in K$ can be idle, travelling to a task, or performing service. We define e_k and v_k for each $k \in K$ to denote the earliest time and location at which the technician becomes available for the next task, respectively. The objective is to find an assignment and ordering of tasks to technicians, such that each task $i \in C$ is carried out by a technician with the required skills or rejected if task i is in the subset $C \setminus C'$; the amount of time delay must not exceed U^{Max} for each scheduled tasks; each technician $k \in K$ starts at the current location v_k at a time not earlier than e_k, visits a subset of tasks, and ends at the location of the last task; and the total cost consisting of response times, penalty cost of late service, and cost of task rejections is minimized.

The above problem is formulated as a deterministic set-partitioning (DSP) model with following parameters and variables. Let $V = K \cup C = \{1, ..., m, m+1, ..., m+n\}$ be the set of vertices, where $K = \{1, 2, ..., m\}$ and $C = \{m+1, ..., m+n\}$ correspond to m technician and n tasks, respectively. Let $t_{i,j}$, $\forall i, j \in V$ be the travel time between vertices i and j. Let R be the set of all feasible technician routes, where each feasible route $r \in R$ is defined by starting at a technician vertex, visiting a sequence of task vertices and ending at the location of the last task assigned, with the relevant time and skill constraints satisfied. For example, a route $r = (1, m+1, m+3, m+5)$ represents technician 1 departing from the current location v_1, visiting tasks $1, 3$ and 5, and ending at the location of task 5. For each route $r = (k, i_1, i_2, ..., i_Q)$ assigned to technician k to serve Q tasks, let w_r and u_r be the total response and delay times over the route, calculated as follows:

$$w_r = \sum_{q=1}^{Q} (b_{i_q} - e_{i_q}) \tag{5.1}$$

$$u_r = \sum_{q=1}^{Q} \max\{b_{i_q} - l_{i_q},\, 0\}, \tag{5.2}$$

where

$$b_{i_1} = \max\{e_k + t_{k,i_1},\, e_{i_1}\} \tag{5.3}$$

$$b_{i_q} = \max\{b_{i_{q-1}} + d_{i_{q-1}} + t_{i_{q-1},i_q},\, e_{i_q}\} \qquad \forall q = 2, ..., Q. \tag{5.4}$$

We introduce two sets of binary parameters:

$$\gamma_{kr} = \begin{cases} 1 & \text{if technician } k \text{ is used by route } r, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall k \in K,\ r \in R;$$

$$\delta_{ir} = \begin{cases} 1 & \text{if task } i \text{ is covered by route } r, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in C,\ r \in R;$$

and two sets of binary decision variables:

$$x_r = \begin{cases} 1 & \text{route } r \text{ is selected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall r \in R;$$

$$z_i = \begin{cases} 1 & \text{if task } i \text{ is rejected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in C.$$

The DSP model is presented as follows:

$$\text{minimize} \quad \sum_{r \in R} (w_r + \beta u_r) x_r + \sum_{i \in C} o_i z_i \tag{5.5}$$

$$\text{subject to:} \quad \sum_{r \in R} \gamma_{kr} x_r \leq 1 \qquad \forall k \in K \tag{5.6}$$

$$\sum_{r \in R} \delta_{ir} x_r + z_i = 1 \qquad \forall i \in C \tag{5.7}$$

$$z_i = 0 \qquad \forall i \in C' \tag{5.8}$$

$$x_r \in \{0, 1\} \qquad \forall r \in R \tag{5.9}$$

$$z_i \in \{0, 1\} \qquad \forall i \in C. \tag{5.10}$$

Constraints (5.6) ensure that each technician is assigned to at most one route. Constraints (5.7) guarantee that each task is either covered by a route or rejected, while constraint (5.8) impose that tasks in the subset C' cannot be rejected. Constraints (5.9) and (5.10) are the binary restrictions on the decision variables.

In addition to the DSP model defined above, we have proposed a deterministic vehicle-flow (DVF) formulation for the off-line WSRP. The details of the DVF model and the computational results can be found in Appendix C.1. As the computational comparison of the two models indicate that the DVF model is not as efficient as the DSP model, we choose to use the latter in what follows.

5.3.2 Route Size Constraints

Compared to the vehicle flow formulation, the set-partitioning formulation generally has stronger linear relaxations, but this comes at the expense of an exponential number of binary variables (Baldacci et al., 2004). Sophisticated techniques such as column generation are required to solve such formulations. However, the nature of the problem we address here is such that customers request immediate service and we are required to assign resource to provide emergency services as soon as possible. In such situation, constructing long routes is neither necessary nor practical, as it leads to long waiting times for some of customers on the route. For this reason, we introduce a set of route size constraints, which enforce that each route is constrained to at most α tasks. With this set of constraints, the number of binary variables of the DSP model is reduced from $O(m\,n!)$ to $O(m\frac{n!}{(n-\alpha)!})$. Such a practical consideration also reduces the computational complexity of the DSP model without losing optimal solutions. Section 5.3.4 describes the way in which we determine the value of α.

5.3.3 Dominated Routes

To further improve the performance of the DSP model, we introduce two dominance rules that describe the dominance relationship between candidate routes. Any dominated route can then be eliminated, because they will never be optimal, as they can be replaced by the corresponding non-dominated routes. The two dominance rules are defined as follows.

Rule 1 Suppose r_1 and r_2 ($r_1, r_2 \in R$) are two routes that are carried out by the same technician to visit same sets of tasks, but have a different ordering of tasks. If inequalities (5.11) and (5.12) are both satisfied with at least one as inequality, then route r_2 is said to be dominated by r_1 (or r_1 dominates r_2).

$$w_{r_1} \leq w_{r_2} \tag{5.11}$$

$$u_{r_1} \leq u_{r_2}. \tag{5.12}$$

Under the objective function (5.5), the above dominance rule is equivalent to the inequality below:

$$w_{r_1} + \beta u_{r_1} < w_{r_2} + \beta u_{r_2}. \tag{5.13}$$

Rule 1 can be viewed as solving a shortest path problem for a given set of tasks and a technician, where the objective function consisting of the response time and the amount of time delay is minimised.

Rule 2 Suppose r_1 and r_2 ($r_1, r_2 \in R$) are two different routes that are carried out by the same technician, but r_1 only includes a subset C_{r_1} of the set C_{r_2} of tasks covered by r_2. Then route r_2 is said to be dominated by r_1 (or r_1 dominates r_2), if the inequality below is satisfied:

$$w_{r_1} + \beta u_{r_1} + \sum_{i \in C_{r_2} \setminus C_{r_1}} o_i < w_{r_2} + \beta u_{r_2}. \tag{5.14}$$

When $C_{r_1} = C_{r_2}$, inequality (5.14) is equivalent to (5.13), and thus the two dominance rules are equivalent. In the extreme case $C_{r_1} = \varnothing$, route r_1 only includes the technician vertex, and the above dominance relationship suggests that the solution of rejecting all tasks in C_{r_2} and allowing the technician to stay idle is better than selecting route r_2.

5.3.4 Optimality Condition

The impact of route size constraints on the number of feasible routes is controlled by the parameter α. With the increase in the value of α, the route size constraints tend to be weaker. In the extreme case $\alpha = |C|$, the set of route size constraints becomes redundant, as all possible and feasible routes are considered by the DSP model. On the other hand, if α takes small values, the route size constraints will filter out all possible routes that contain more than α tasks, and the optimal solution may be excluded from the searching space. In order to find as small as possible a value of α, such that the optimal solution is not discarded, we define the following optimality condition:

Rule 3 For a given task set C, let R_1 and R_2 be two sets of all feasible routes, where $r_1 \in R_1$ and $r_2 \in R_2$ contain α and $\alpha+1$ tasks, respectively. If any $r_2 \in R_2$ is dominated by a route $r_1 \in R_1$ by the Rule 2, then R_2 is said to be a completely dominated route set and any route containing more than $\alpha + 1$ tasks is also a dominated route. Thus, the optimal solution only includes routes which contain up to α tasks.

Therefore, to determine the minimum value of α without loss of optimality, we can start with a small enough value of α (e.g. $\alpha = 1$) and increment the value by 1 until it is no longer possible to construct feasible routes or a completely dominated route set is obtained

5.4 Exploiting Stochastic Information about Future Requests

To exploit the stochastic information on future requests, we develop a new strategy that allows the above deterministic model to consider potential requests that may arise in the near future. More precisely, we extend the set R of routes of the DSP model to include a set C^+ of random tasks assumed to arrive up to a near future epoch. To keep the resulting model tractable, we limit the size of C^+ to the number of technicians $|K|$, and impose that each technician route can be appended at most one future task. With this restriction, we only need to deal with an assignment problem of future tasks to technician routes, where the information about service times of future tasks will not be required. This strategy actually allows us to evaluate the impact of current routing decisions on the incoming requests. An example of this strategy is illustrated in Figure 5.3, where white circles represent four known tasks, and red circles show two future tasks, each appended to a route. According to the solution shown, technicians 1 and 2 will perform future tasks 6 and 5 after they complete their assigned tasks.

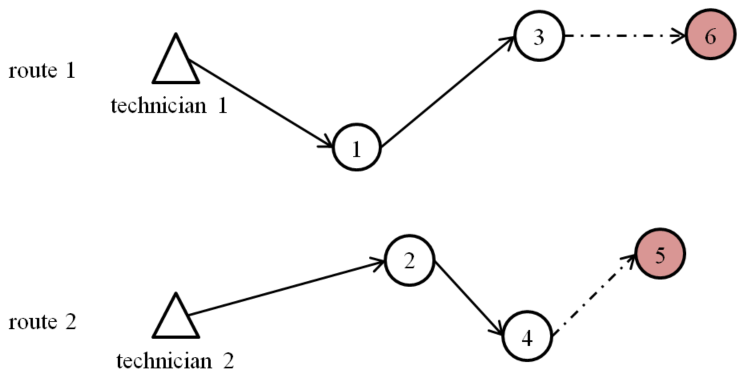


Figure 5.3: Example of the proposed strategy with future tasks

The proposed strategy aims to find a set of routes which cover the set of currently known tasks and also can provide prompt response to future customers. The set C^+ of future tasks is randomly generated based on the probability distribution derived from the historical data. As different random sets of C^+ will yield different solution, which may be arbitrarily poor if C^+ is not a good representation of the true realization. We apply a scenario-based approach to implement the strategy described above. Let S be the set of scenarios, where each scenario $s \in S$ corresponds to a set C_s^+ of future tasks with an associated probability p_s. Let $C^+ = C_1^+ \cup C_2^+ \cup ... \cup C_{|S|}^+$ be the set of all future tasks. Each task $j \in C^+$ is also associated with a time window (e_j, l_j) and a penalty cost of rejection o'_j. For a late service at task j, each unit of tardiness is penalized by a factor β'. Lastly, we define $t'_{r,j}$, $\forall r \in R$, $\forall j \in C^+$ to be the travel time from the destination of route r to the future task j. The above problem is formulated as a two-stage model, where the first stage considers finding a set of feasible routes for the known tasks, while the second stage handles the assignment of future tasks to the routes found in the first stage. We have examined two formulations for the above two-stage problem. The first one is straightforward, which combines the DSP model and an assignment model to deal with the first-stage and second-stage problems, respectively. The second one is based on a single set-partitioning framework, but the variables need to be defined in a novel way. The computational results (see Appendix C.3.1) indicate that the second model performs better than the first one. The details of the first model can be found in Appendix C.3, while the second model is presented in the following section.

5.4.1 Stochastic Set-Partitioning Model

Recall that R to is the set of all feasible routes that cover known tasks in C. As we need to maintain the structure of the two-stage model, we cannot simply extend routes in R to include future tasks. Thus, we introduce the concept of future routes, defined by starting at the destination of a current route $r \in R$, visiting a future task $j \in C^+$

and ending there. Let $r^+ = (r, j)$, $r \in R, j \in C^+$ represent a feasible future route. It must satisfy the following constraints: (1) the start time of r^+ must be not earlier than the completion time c_r of the current route r and the arrival time of future task j, (2) the amount the time delay at j must not exceed U^{Max}, and (3) the technician of r has the required skills for serving j. We define w_{r^+} and u_{r^+} to be the response time and the amount of time delay of r^+, computed as:

$$\begin{aligned} w_{r^+} &= \max\{c_r + t'_{r,j} - e_j, t'_{r,j}\} \\ u_{r^+} &= \max\{c_r + t'_{r,j} - l_j, e_j + t'_{r,j} - l_j, 0\}. \end{aligned}$$

For each scenario $s \in S$, we define R_s^+ to be a set of feasible future routes. Then $R^+ = R_1^+ \cup R_2^+ \cup ... \cup R_{|S|}^+$ is the set of all feasible future routes. We define a binary parameter δ_{jr}^+, $\forall j \in C^+$, $\forall r \in R^+$, which equals to 1 if future task j is covered by future route r, and 0 otherwise; and a binary parameter $\rho_{r_1 r_2}$, $\forall r_1 \in R$, $\forall r_2 \in R^+$ which equals to 1 if current route r_1 is linked with future route r_2, and 0 otherwise. Lastly, we introduce the following binary variables:

$$y_r^s = \begin{cases} 1 & \text{if future route } r \text{ of scenario } s \text{ is selected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall r \in R_s^+,\ \forall s \in S;$$

$$z_i^s = \begin{cases} 1 & \text{if future task } i \text{ of scenario } s \text{ is rejected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in C_s^+.\ \ \forall s \in S.$$

The proposed stochastic set-partitioning (SSP) model is presented below:

$$\text{minimize} \quad \sum_{r \in R} (w_r + \beta u_r) x_r + \sum_{i \in C} o_i z_i + \lambda \sum_{s \in S} p_s \Big(\sum_{r \in R_s^+} w_{r^+} y_r^s + \beta' \sum_{r \in R_s^+} u_{r^+} y_r^s + \sum_{j \in C_s^+} o'_j z_j^s \Big) \tag{5.15}$$

subject to: (5.6)–(5.10)

$$\sum_{r \in R_s^+} \delta_{ir}^+ y_r^s + z_i^s = 1 \qquad \forall i \in C_s^+, \forall s \in S \tag{5.16}$$

$$\sum_{r_2 \in R_s^+} \rho_{r_1 r_2} y_{r_2}^s \le x_{r_1} \qquad \forall r_1 \in R, \forall s \in S \tag{5.17}$$

$$z_i^s \in \{0, 1\} \qquad \forall i \in C_s^+, \forall s \in S \tag{5.18}$$

$$y_r^s \in \{0, 1\} \qquad \forall r \in R_s^+, \forall s \in S. \tag{5.19}$$

Constraints (5.16) ensure that each future task is either covered by a future route or rejected. Constraints (5.17) enforce that a future route can be selected only if the associated current route is selected in the first stage.

5.4.2 Evaluation of Stochastic Solutions

To evaluate the performance of the stochastic solutions, we carry out the following experiment. We first solve the SSP model to obtain a set R^* of technician routes. We assume that the routing plan is fixed such that R^* is carried out by the technicians assigned. Once technicians complete their scheduled routes, they become available again for new tasks. Let $Q(R^*)$ be the cost of executing R^*. We then randomly generate a set ξ of realisations, where $\xi_i \in \xi$ is the true realisation in the near future and contains a set $C^+_{\xi_i}$ of new tasks, where $|C^+_{\xi_i}| = |K|$. To investigate how the current routing plan R^* performs in responding to new tasks, we solve $|\xi|$ assignment problems in terms of R^* and each of the realisations, $\xi_i \in \xi$. The assignment problem aims to find the optimal plan between R^* and $C^+_{\xi_i}$, such that either a technician $k \in K$ can carry out a new task $i \in C^+_{\xi_i}$ upon completion of the assigned route $r_k \in R^*$ or the task is rejected, and where the total costs of serving $C^+_{\xi_i}$ is minimised. Let $Q(R^*, \xi_i)$ be the optimal solution value of such an assignment problem. We define $\overline{Q}(R^*, \xi)$ to be the average solution value in terms of R^* and the entire set ξ of realisations, and compute it as

$$\overline{Q}(R^*, \xi) = \frac{1}{|\xi|} \sum_{i}^{|\xi|} Q(R^*, \xi_i),$$

used to indicate the performance of a routing plan R^* with respect to the future realisation set ξ. We define $Q^*(R^*) = Q(R^*) + \overline{Q}(R^*, \xi)$ to be the overall performance indicator of R^*, which takes both known tasks and future tasks into account. A smaller value of $Q^*(R^*)$ indicates better performance of the routing plan R^*.

We also apply two commonly used measurements, the *expected value of perfect information* (EVPI) and the *value of the stochastic solution* (VSS) (Birge and Louveaux, 2011), to assess the importance of using stochastic information. The EVPI estimates the difference between the solution values of SP and WS, where SP corresponds to the solution produced by our stochastic model, and WS refers to a wait-and-see solution (Madansky, 1960), obtained under the assumption that decisions are made after waiting and observing the realisation of the random variables. To compute the WS solution, we first separate our stochastic model by scenarios, which results in $|S|$ independent deterministic problems, each is associated with a particular scenario $s \in S$. Let min $F(x_r, z_i, s)$ be such an optimisation problem. Then the WS solution can be defined as the average objective value shown as $\mathbb{E}_{s \in S}[\min_{\ x} F(x_r, z_i, s)]$. The EVPI is the percentage reduction in the solution value if perfect information about future realisation is available, computed as $100 \times (\text{SP} - \text{WS})/\text{SP}$.

The VSS compares the SP solution with the solution of a much simpler expected value (EV) problem, in which the random variables are replaced by their expected values (Birge and Louveaux, 2011). The expected value of using the EV solution is defined by

EEV, which measures the performance of the second-stage solution when the first-stage decisions are determined by the EV problem. The VSS is equal to the difference between the solution values of EEV and SP. For the problems considering stochastic demands, stochastic travel times and stochastic service times, the corresponding EV problems can be obtained by simply replacing stochastic parameters with their mean values. However, we cannot use this approach to construct our EV problem, as our models consider future requests as the random parameters. Thus, we simply use the DSP model to represent the EV problem, where the stochastic information of future tasks are not utilised. Once the EV solution is obtained, we evaluate how the solution performs in terms of responding to future tasks described by the scenario set S. The procedure of computing the EEV is actually similar to the one defined above for calculating $Q^*(R^*)$, where the scenario set S is considered as the realisation set ξ and the routing plan R^* is generated without utilizing stochastic information. We defined VSS as the percentage difference between the solution values of SP and EEV, and it is calculated as $100 \times (\text{EEV} - \text{SP})/\text{EEV}$.

The following sections presents the results of the computational experiments conducted to assess the performance of the proposed models. The models are implemented on a personal computer with Intel Core i5-3570 3.40 GHz processor and 4GB Memory (RAM), and solved by CPLEX 12.6. We first describe the results of off-line experiments, and then present the results of on-line experiments within a simulation framework.

5.5 Off-line Experiments

The off-line experiments aim to compare the performance of the proposed models in terms of the solution quality and the computational speed, and also investigate the influence of different parameter settings. We first describe the test instances used in the experiments. Then we investigate the effect of dominance rules and route size constrains. Lastly, we assess the performance of our stochastic solutions using the performance indicators described above.

5.5.1 Generation of Test Instances

The test instances are generated based on historical data containing service requests received within the area of Southampton in the UK. The entire area is split into 95 squares (as shown in Figure 5.4), where each square has a side length of 5 km and represents a demand zone. We assume that the entire region is in a two-dimensional Euclidean space, and the location of each demand zone $j \in \{1, 2, ..., 95\}$ is represented by the coordinates of its bottom-left corner.

Each instance contains a set of service requests, generated using the arrival distribution of customers and the service time distribution, defined as follows.

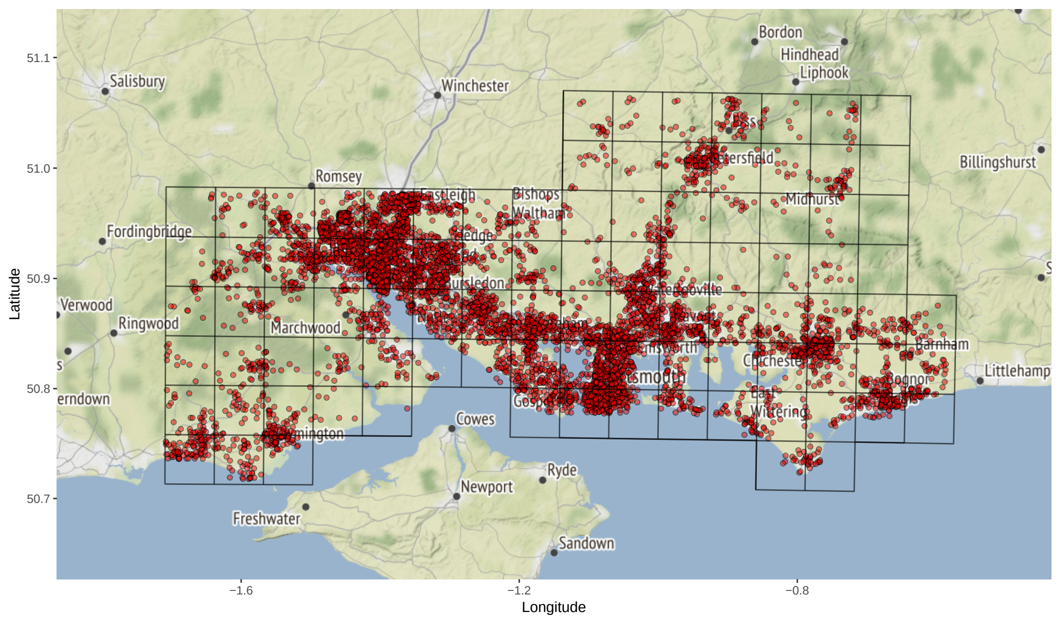
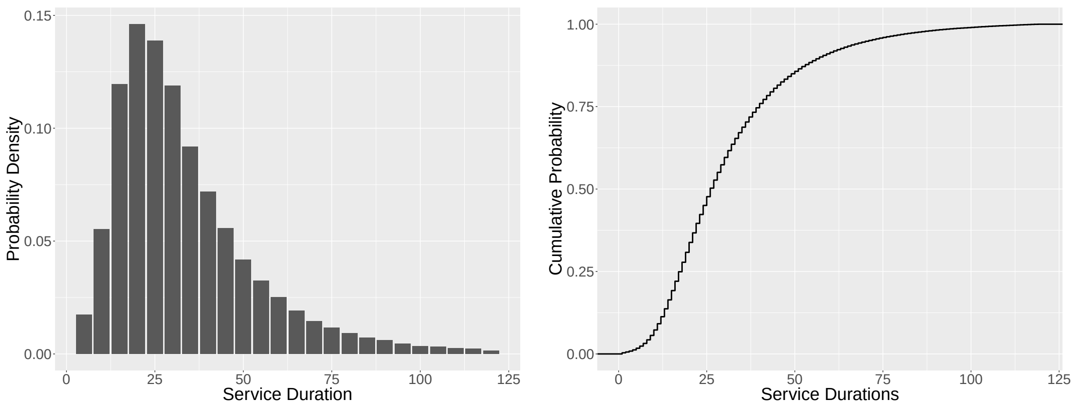


Figure 5.4: Illustration of the demand areas of the Southampton area

1. The arrival distribution is modelled by a Poisson process, where the arrival rate of each demand zone is defined by the average number a_j of requests received in zone $j \in \{1, 2..., 95\}$. According to the superposition property of the Poisson process, the arrival of service requests in the entire area also follows a Poisson process with rate of $\bar{a} = \sum_{j=1}^{95} a_j$. To capture the geographical distribution of service requests, we define a density factor D_j for each demand zone j, and computed as $D_j = a_j/\bar{a}$.

2. Service durations: the analysis of historical data suggests that the service duration does not follow any specific parametric distributions, for which reason we construct an empirical distribution with mean service duration equal to 31.25 minutes. The distribution of service durations is shown in Figure 5.5.



(a) probability density plot

(b) cumulative probability plot

Figure 5.5: Probability distribution of service durations

Service requests are generated independently by demand zones. The location of a service request within a demand zone is determined according to uniform distributions along the x and y axes. Once a task i is generated, an associated time window (e_i, l_i) is created, where e_i is set to the arrival time of the task and l_i is computed as $e_i + T^{Tar}$. The values of T^{Tar} and U^{MAX} (the maximum amount of time delay allowed) are both set to 60 minutes according to the policy of the company. Thus, the service at a task must start within two hours of receiving the associated request; otherwise, it is outsourced.

Each instance corresponds to a decision epoch corresponding to the arrival time of the latest task of the instance and contains a set of N tasks that have been received but not yet locked or permanently rejected. We set the total arrival rate $\bar{a}$ to N, and then compute the individual arrival rate of each demand zone based on the corresponding density factor defined above. By adjusting $\bar{a}$ to N, we assume that each instance only consists of tasks that are received within approximately an one-hour time interval. This assumption ensures that all tasks can be reached by technicians within reasonable amount of response times if accepted. Next, we generate random tasks and store them in a task list denoted by LS. This is implemented by a loop structure, where at each iteration, we generate tasks that will arise in the next one-hour interval, denoted by T^{Limit}. Tasks are generated independently by demand zones. Once the generation has been completed for all demand zones, we check the size of LS. If $|LS| < N$, we repeat the above procedure to generate tasks for another one-hour period; otherwise, the loop terminates and produces a task list LS containing at least N tasks. Finally, we sort the task list LS in ascending order of the arrival times of tasks and select the first N tasks to form a test instance. The pseudo-code of the above procedure can be found in Appendix C.4.

Each instance also includes a set of technicians, whose locations are distributed randomly across the entire demand region, but following a similar geographical distribution of requests to avoid placing technicians to unrealistic locations. The earliest available times of technicians are set to the decision epoch of the corresponding instance. We assume that all technicians are initially idle and can start service as soon as tasks have been scheduled to them. This assumption allows to better assess the influence of different staffing levels on the scheduling and routing solutions. For simplicity reasons, technicians are assumed to be identical and have the required skills to provide services for all tasks. Moreover, we assume that all technicians travel at the same constant speed of 40 km/h determined by using the average driving speed on local roads in England according to the UK Department for Transport (Department for Transport (DfT), 2016). To estimate the travel distance between two locations within the demand region, we first calculate the Euclidean distance, and then adjust it by multiplying a factor estimated as 1.40 for England (Ballou et al., 2002) in order to account for the circuitous nature of actual road network. Lastly, we introduce a parameter called resource ratio, denoted by θ, which describes the staffing level. For an instance containing $|C|$ tasks, the number of technicians is determined by $\theta|C|$.

When solving each of the instances defined above, the proposed stochastic models use a set of random scenarios. Each scenario corresponds to a set of future tasks, which are generated using an approach similar to the one used for generating known tasks. The initial time of the generation procedure is set to the current decision epoch of the instance. The generation terminates until the number of future tasks generated is equal to the number of technicians, since our models only allow each of the technician routes to accommodate at most one future task.

Based on preliminary experiments, the parameters α and β are set to 5 and 1 respectively. The penalty cost of rejecting a task is set to a relatively high value equal to 300, making it preferable to schedule a task rather than to reject it, since the cost of serving a task at its latest time is 180 (including 120 minutes of response time and 60 minutes of allowable time delay), which is much less than the rejection cost. However, if tasks are difficult and require long service times, we may achieve cost savings by rejecting them. Lastly, a time limit of 3,600 seconds is imposed on the CPLEX. For instances not solved to optimality, we report the values of the best solutions found within the time limit and the final optimality gaps.

5.5.2 The Effect of the Dominance Rules

To assess the effectiveness of the dominance rules defined in Section 5.3.3, we conduct computational experiments based on 15 instances containing up to 50 tasks. For each instance, we test three resource ratios, for low ($\theta = 0.2$) , medium ($\theta = 0.5$) and high ($\theta = 0.8$) staffing levels. Table 5.2 presents the computational results. The first group of columns show the instance identifier, the number of tasks $|C|$, and the resource ratio θ. The columns titled Var and T(s) show, for each instance, the number of variables and the CPU solution time in seconds, respectively. The solution values and the number of constraints are not reported in this table, because the dominance rules only affect the number of variables and computational times, and the model with or without dominance rules produces exactly the same optimal solutions on all instances tested. In addition to the two dominance rules defined previously, the table includes a third called Dominance Rule Three, where rules one and two are used together. To compare the performance of dominance rules, we define the percentage decrease in the number of variables as

$$\text{Gap}_{i/j}^{\text{Var}} = \frac{v^{\text{Var}}(i) - v^{\text{Var}}(j)}{v^{\text{Var}}(i)} \times 100, \tag{5.20}$$

where $v^{\text{Var}}(i)$ and $v^{\text{Var}}(j)$ represent the number of variables of using policies i and j respectively, and $i, j \in \{0, 1, 2, 3\}$. Policy 0 refers to the model without using dominance rules, and policies 1, 2 and 3 correspond to the model running with dominance rule one, two and three, respectively.

			No Dominance		Dominance Rule One			Dominance Rule Two			Dominance Rule Three		
Instance	$\|C\|$	θ	Var	T(s)	Var	$\text{Gap}^{\text{Var}}_{0/1}$	T(s)	Var	$\text{Gap}^{\text{Var}}_{0/2}$	T(s)	Var	$\text{Gap}^{\text{Var}}_{0/3}$	T(s)
1	10	0.2	55	0.00	50	9.09	0.02	55	0.00	0.00	50	9.09	0.00
2	10	0.5	125	0.00	112	10.40	0.02	124	0.80	0.00	111	11.20	0.02
3	10	0.8	208	0.00	178	14.42	0.02	207	0.48	0.00	177	14.90	0.02
4	20	0.2	829	0.02	611	26.30	0.02	806	2.77	0.02	597	27.99	0.02
5	20	0.5	3669	0.05	2377	35.21	0.05	3569	2.73	0.05	2328	36.55	0.05
6	20	0.8	5843	0.03	3783	35.26	0.03	5682	2.76	0.03	3702	36.64	0.03
7	30	0.2	3329	0.08	2246	32.53	0.09	3305	0.72	0.08	2246	32.53	0.08
8	30	0.5	13914	0.47	8149	41.43	0.45	13758	1.12	0.48	8104	41.76	0.45
9	30	0.8	20142	0.11	11979	40.53	0.08	19875	1.33	0.11	11912	40.86	0.09
10	40	0.2	4669	0.06	3204	31.38	0.06	4648	0.45	0.06	3200	31.46	0.08
11	40	0.5	11849	0.25	8047	32.09	0.23	11786	0.53	0.38	8037	32.17	0.33
12	40	0.8	19289	0.11	13004	32.58	0.09	19181	0.56	0.11	12987	32.67	0.09
13	50	0.2	49616	0.83	22689	54.27	0.77	47785	3.69	0.73	22450	54.75	0.73
14	50	0.5	117956	1.15	53370	54.75	0.95	113488	3.79	1.22	52640	55.37	0.92
15	50	0.8	212368	1.23	93343	56.05	0.83	203967	3.96	1.19	92120	56.62	0.83
Average			30924.07	0.29	14876.13	33.75	0.25	29882.40	1.71	0.30	14710.73	34.30	0.25

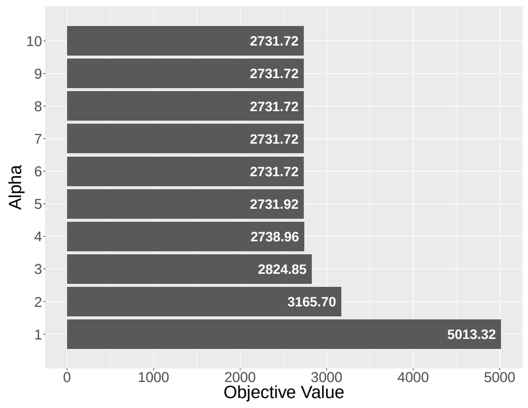
Table 5.2: The comparison of the dominance rules

Rule one achieves an average reduction of 33.75% on the number of variables. For large instances containing 50 tasks, more than 50% of the feasible routes are identified as dominated by rule one. Rule two, however, can only reduce the number of variables by less than 5%. This can be explained by the fact that the task rejection cost is set to a relatively high value, which is 300, where tasks are rejected mostly due to the time window constraints rather than solution costs. Dominance rule three reduces the number of variables by 34.30% on average. The reduction on the number of variables results in an improvement in computational speed, where the model is able find optimal solutions for instances containing up to 50 tasks within a second.

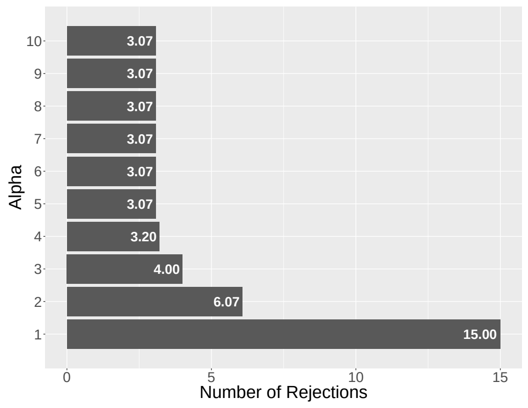
5.5.3 Evaluation of Route Size Constraints

To investigate the effect of different values of parameter α, we conduct computational experiments based on the same set of instances used above, and ten values of $\alpha \in \{1, 2, ..., 10\}$ are tested. Figure 5.6 displays the average solution values of the instances solved by the DSP model with different values of α.

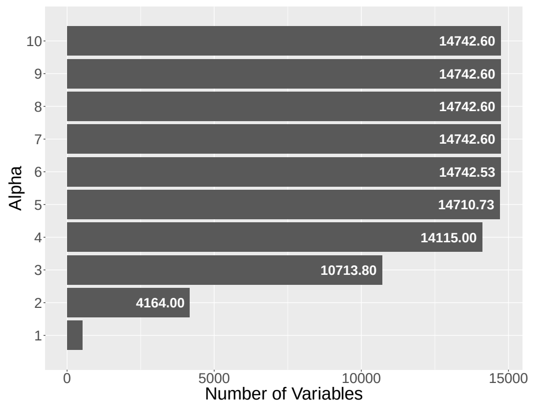
From Figure 5.6a, it can be observed that the average objective value decreases when the value of α increases from 1 to 6, and then remains the same even if α increases up to 10. This implies that even if longer routes are allowed, near-optimal solution can be obtained with $\alpha = 4$ or 5, where $\alpha \geq 6$ yields optimal values. Comparing results of $\alpha = 4$ with $\alpha = 5$, the percentage difference between the corresponding objective values is only 0.25%. When $\alpha \leq 2$, the average objective values are considerably worse, as a proportion of tasks are rejected due to the limitation on the route size and the lack of technicians. This is also shown in Figure 5.6b, where the average numbers of rejections for $\alpha = 1$ and $\alpha = 2$ are much higher than those of other groups. By increasing α from 5 to 10, the number of rejected tasks remain the same, which suggests that the rejections



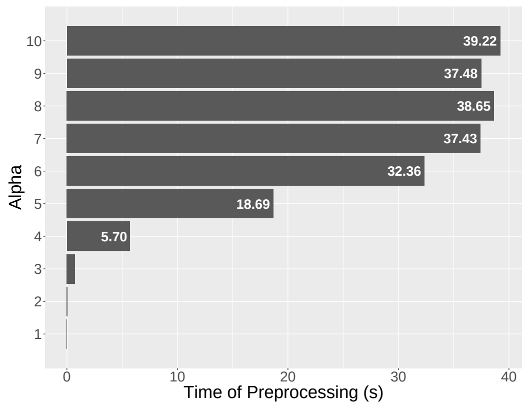
(a) Objective value vs. alpha



(b) Number of rejections vs. alpha



(c) Number of variables vs. alpha



(d) Time of preprocessing vs. alpha

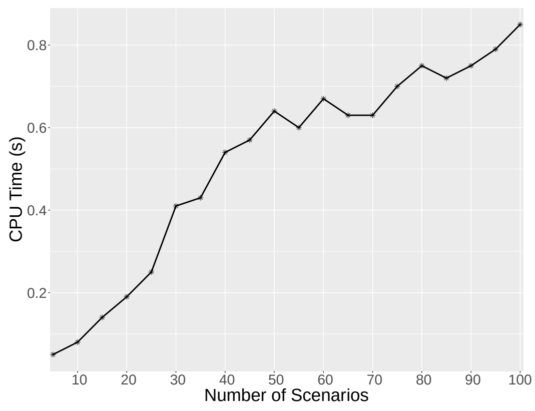
Figure 5.6: The solution values of the SP model running with different values of alpha

are due to the restriction of time window constraints and possibly the consideration of routing costs rather than the route size constraints.

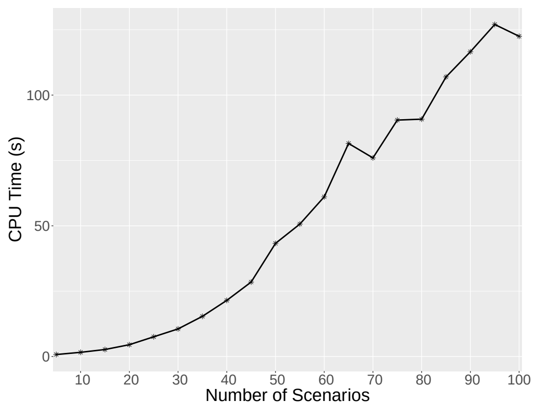
Figure 5.6c displays that the average number of variable increases until $\alpha = 7$, even though the change from $\alpha = 5$ or 6 is insignificant. The set of all possible routes containing seven tasks is a completely dominated set according to the definition in Section 5.3.4. Therefore, it is actually not necessary to test $\alpha > 7$, as any route containing more than 7 tasks is an infeasible or dominated route and it will not be selected in the optimal solution. Increasing the value of α does not significantly increase the computational times required, because the number of variables is strongly affected by the dominance rules. However, increasing α significantly increase the amount of time required for the data preprocessing step (see Figure 5.6d), which involves the evaluation of all possible routes and selection of non-dominated routes. Based on the efficiency and effectiveness, we choose $\alpha = 4$ in the remainder of the tests.

5.5.4 Number of Scenarios vs. Computation Times

To assess the effect of the number $|S|$ of scenarios on the computational times required by the SSP model, we examine twenty values of $|S|$, starting with 5 until 100 using a step size of 5. Each value of $|S|$ is tested on two sets of instances, where one set contains ten instances of 10 known tasks and the other contains ten instances of 20 known tasks. Each scenario $s \in S$ has the same probability $p_s = 1/|S|$. The weight parameters λ and β' of the objective function (5.15) are both set to 1. The penalty cost of rejecting a future task is set equal to the cost of rejecting a known task ($o'_j = o_i = 300$, $\forall j \in C^+, \forall i \in C$). To keep the computational experiments manageable, we fix the resource ratio to 0.5, which corresponds to a medium staffing level. The results are illustrated in Figure 5.7, where the graphs display the changes of the average CPU times against the value of $|S|$. Even the number of scenarios is increased to 100, the SSP model can solve all instances to optimality. With the increase of the number of scenarios, the computational time increases accordingly, and the graphs suggest that there exists a roughly linear relationship between them.



(a) Instances with 10 known tasks



(b) Instances with 20 known tasks)

Figure 5.7: The relationship between the number of scenarios used and the computational times required

5.5.5 Number of Scenarios vs. Performance of Stochastic Solutions

As defined in Section 5.4.2, we use $Q^*(R^*)$ to measure the performance of stochastic solutions. In addition to the two sets of instances used previously, we introduce two sets, which contains 50 and 100 realisations, respectively. Each instance is paired with a set of realisations, which results in a total of 40 combinations. The resource ratio is fixed to 0.5. To compute $Q^*(R^*)$, we first use the SSP model to solve an instance to generate a routing plan R^* and compute the corresponding solution cost $Q(R^*)$. Then we evaluate R^* on a set ξ of realisations to get $\overline{Q}(R^*, \xi)$. Lastly we sum the values of $Q(R^*)$ and $\overline{Q}(R^*, \xi)$ to obtain $Q^*(R^*)$. To investigate the sensitivity of $Q^*(R^*)$ to the number of scenarios, we examine 11 values of $|S|$ given by $\{1, 5, 10, ..., 50\}$. Each combination of instance and set of realisations is solved ten times using the SSP model with randomly generated scenarios. We report the average solution values and the standard deviation obtained over these ten runs.

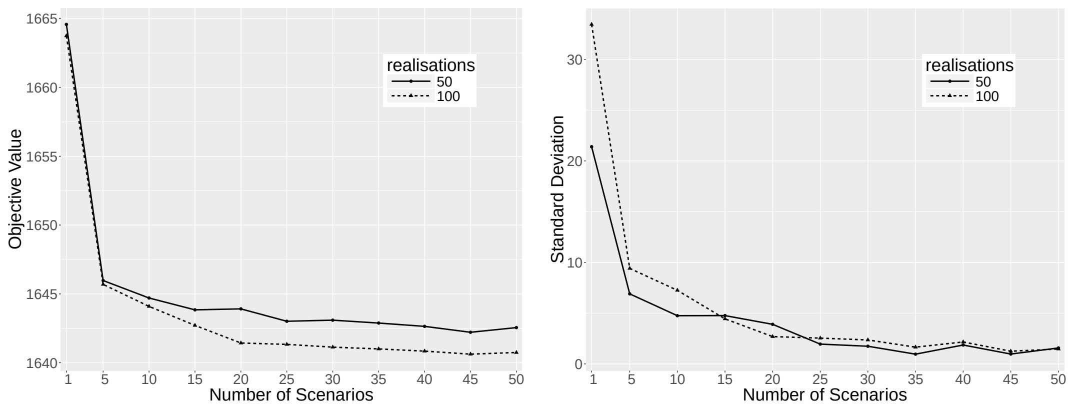


Figure 5.8: Sensitivity analysis of stochastic solutions to the number of scenarios on instances with 10 known tasks

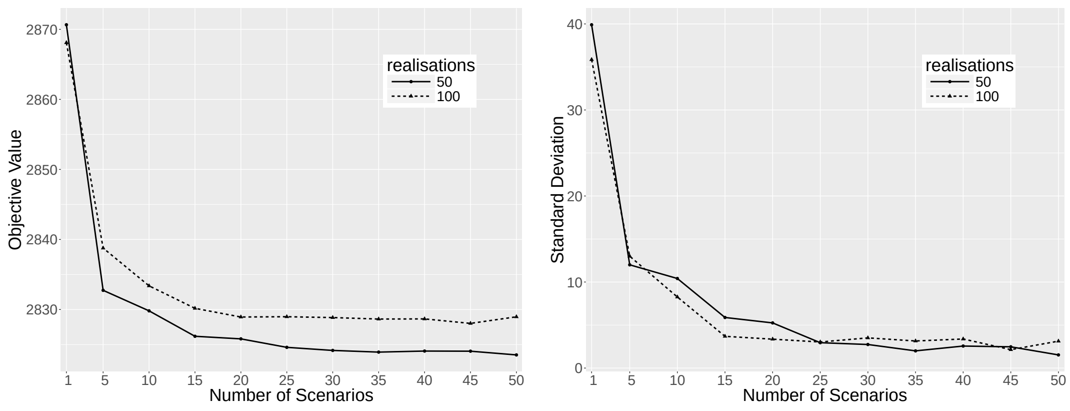


Figure 5.9: Sensitivity analysis of stochastic solutions to the number of scenarios on instances with 20 known tasks

Figure 5.8 and Figure 5.9 present the changes of the average solution values and standard deviations when the number of scenarios used varies between 1 and 50, where the solid lines and the dotted lines display the results of using 50 and 100 realisations, respectively. It can be seen that the average solution values of $|S| = 1$ are significantly higher than those of other groups. This indicates that the SSP model occasionally produces bad solutions when only one scenario is used. The average solutions values decrease as the number of scenarios used increases from 1 to 20, and then remain briefly stable. The standard deviation also shows a similar pattern. There is no significant difference between the results of using 50 realisations and 100 realisations. Based on the above results, we choose to use 20 scenarios for the SSP model.

To assess the performance of the stochastic solutions, we conduct the same experiments defined above, but this time by using the DSP model. Since no random scenario is used, each pair of instance and realisation set is solved only once using the DSP model. The obtained output is compared with the stochastic solutions obtained previously using the SSP model with a set of 20 scenarios applied. The average solution values for each instance set are displayed in Table 5.3. The columns titled Obj^1 and Obj^2 show the average solution values with respect to known tasks and future realisations respectively, which also correspond to $Q(R^*)$ and $\overline{Q}(R^*, \xi)$ defined previously. The sum of Obj^1 and Obj^2 is displayed in column Obj^3, which corresponds to $Q^*(R^*)$. The corresponding numbers of rejections are shown in columns named Rej^1, Rej^2 and Rej^3.

			DSP Model						SSP Model Using 20 Scenarios					
Set	$\|C\|$	$\|\xi\|$	Obj^1	Rej^1	Obj^2	Rej^2	Obj^3	Rej^3	Obj^1	Rej^1	Obj^2	Rej^2	Obj^3	Rej^3
1	10	50	882.47	0.30	915.87	1.73	1798.34	2.03	1046.28	1.86	597.63	0.47	1643.91	2.33
2	10	100	882.47	0.30	914.64	1.73	1797.10	2.03	1051.91	1.90	589.52	0.45	1641.43	2.35
3	20	50	1450.48	0.10	1579.34	2.18	3029.82	2.28	1742.83	2.14	1082.98	0.68	2825.81	2.82
4	20	100	1450.48	0.10	1586.27	2.20	3036.75	2.30	1736.42	2.10	1092.52	0.70	2828.93	2.80
Avg			1166.47	0.20	1249.03	1.96	2415.50	2.16	1394.36	2.00	840.66	0.57	2235.02	2.57

Table 5.3: Comparison of deterministic and stochastic solutions

It can be seen that the deterministic solution has better performance on known tasks than the stochastic solution. However, stochastic solutions have much better performance on future realisations. Considering both known and future tasks, the stochastic solution outperforms the deterministic solution by 7.42% on average. The improvement achieved by the stochastic solution can be explained by the increase on the number of rejections. The stochastic model tends to reject some of known tasks in order to achieve a better resource management and provide quick response to future tasks.

5.5.6 VSS and EVPI

To compute the values of VSS and EVPI, we follow the method described in Section 5.4.2. The experiments are based on two sets of instances used above, and each instance is associated with a set of 20 scenarios. Moreover, we examine seven values of the resource ratio θ ranging from 0.2 to 0.8 in increments of 0.1. Table 5.4 presents the average solution values for each instance set. In addition to the columns defined previously, we use columns $\text{Gap}_{1/2}^{\text{Rej}}$ and $\text{Gap}_{2/3}^{\text{Rej}}$ to display the percentage differences between the rejection numbers of EEV and SP, and SP and WS solutions, respectively, and they are computed using a equation similar to (5.20).

Over all instances and resource ratios tested, the average value of VSS is 4.52% that shows the value of the SSP model using stochastic information. When $\theta = 0.4$, the average values of VSS for the 10 tasks set and the 20 tasks set are 9.88% and 9.25%, respectively. This indicates that when a median staffing level is used, exploiting stochastic information is particularly important as it allows the SSP model to produce a better resource management for both known and future tasks. However, the average rejection number of SP solutions is about 7% greater than that of EEV solution, which implies

			EEV		SP		WS					
Set	$\lvert C\rvert$	θ	Obj	Rej	Obj	Rej	Obj	Rej	VSS	$\text{Gap}_{1/2}^{\text{Rej}}$	EVPI	$\text{Gap}_{2/3}^{\text{Rej}}$
1	10	0.2	2436.49	6.32	2408.10	6.22	2366.93	6.00	1.17	1.58	1.74	3.62
2	10	0.3	2169.87	4.55	2067.85	4.51	2030.99	4.16	4.70	0.88	1.82	7.66
3	10	0.4	1993.11	3.06	1796.22	3.07	1758.21	2.73	9.88	−0.49	2.16	11.07
4	10	0.5	1724.98	1.99	1597.84	2.11	1564.16	1.83	7.37	−6.30	2.15	13.51
5	10	0.6	1460.58	1.14	1436.74	1.28	1406.59	1.14	1.63	−11.84	2.14	10.98
6	10	0.7	1278.75	0.65	1252.87	0.74	1234.20	0.66	2.02	−14.73	1.51	10.81
7	10	0.8	1135.08	0.42	1112.77	0.43	1098.00	0.39	1.97	−3.61	1.34	9.30
8	20	0.2	4499.57	10.77	4356.71	10.54	4293.08	10.07	3.18	2.14	1.48	4.51
9	20	0.3	3929.71	7.02	3684.44	6.83	3624.45	6.37	6.24	2.71	1.66	6.73
10	20	0.4	3506.66	4.41	3182.39	4.40	3125.55	4.00	9.25	0.34	1.82	8.99
11	20	0.5	2923.17	2.27	2749.82	2.53	2701.88	2.20	5.93	−11.45	1.77	13.24
12	20	0.6	2490.94	1.19	2404.36	1.58	2362.09	1.30	3.48	−32.77	1.79	17.72
13	20	0.7	2187.03	0.78	2109.96	0.80	2075.87	0.80	3.52	−3.23	1.64	0.63
14	20	0.8	1963.75	0.47	1905.24	0.59	1877.06	0.53	2.98	−24.47	1.50	10.26
Avg			2407.12	3.22	2290.38	3.26	2251.36	3.01	4.52	−7.23	1.75	9.22

Table 5.4: Comparison of EEV, SP and WS solutions

that the SSP model tends to reject some known requests in order to achieve higher service quality for future requests. On the other hand, when an extremely low or high staffing level ($\theta = 0.2$ or 0.8) is used, the average value of VSS is only around 3%. In the case of a low staffing level, a large proportion of tasks have to be rejected and very limited improvement can be obtained by utilizing stochastic information. In the case of a high staffing level, there is enough resource to service both known and future tasks, and using stochastic information to prepare for the future therefore becomes less important. In contrast to the results of VSS, the EVPI is less sensitive to the value of θ. Although the average value of EVPI over all instances conducted is only 1.75%, the average rejection number of SP solutions is 9.22% higher than that of WS solutions. If the penalty cost of rejecting tasks increases, the value of EVPI is expected to increase accordingly. The results also indicate that better quality solutions with significantly less number of rejections can be identified if the information about future tasks is known in advance.

5.6 On-line Experiments

With the above off-line experiments confirming the advantage of exploiting stochastic information about future requests, this section presents on-line experiments which assess the performance of the SSP model within a simulation framework.

5.6.1 Simulation Setup

In the simulation, the task arrival is modelled by a Poisson process. To reduce the model complexity, we assume that the arrival rate is not dependent on time, and thus the task arrival rate of each demand zone is set to a constant value. The arrival rate is relevant to the peak period of the day (9am to 10 am), which corresponds to 19.26 requests per hour within the entire Southampton area. Tasks are generated independently by demand zones. The number $|K|$ of technicians is set to a constant in each simulation run. The initial locations of the technicians are randomly distributed across the Southampton area, but following a similar geographical distribution of requests. Technicians are not required to return to their home bases, and only travel between task locations within each simulation run. The simulation model is driven by random instance sets, where each set contains $1,000$ tasks generated based on the Poisson process. Each instance set is split into a training set and a test set, where the former contains the first 100 tasks and the latter includes the remaining 900 tasks. The simulation starts with a warm-up period using the training set. Once the warm-up period is finished, the simulation runs with the test set to collect the results. The outputs of the simulation include the average solution value, the average response time, the average amount of time delay and the overall rejection rate.

The simulation is run under a simple policy, which reoptimises the model whenever a new task arrives. In each re-optimisation run, we interrupt the current plan and generate a new plan to be carried out until the next decision epoch, where the plan is generated by solving the corresponding offline problem using the DSP or SSP model. In addition to these two models, we also develop a simple policy based on a naive greedy algorithm (NGA). For a new task i, the NGA calculates the incremental cost of inserting it at the end of the job list of every technician. If all the incremental costs are higher than the rejection cost o_i, task i is rejected; otherwise, it is assigned to the technician that gives the minimum incremental cost. A similar heuristic has been used by Yang et al. (2004).

The DSP and SSP model are solved by CPLEX under a time limit of 20 seconds to keep the overall optimization time manageable. If CPLEX is not able to solve the problem within the time limit, we use the NGA policy to generate the new plan. The new scheduling plan is carried out until the next decision epoch is reached.

5.6.2 Simulation Results

To assess the effect of the number of technicians on the system performance, we examine six values of $|K|$ displayed in Table 5.5. Each value of $|K|$ is tested under ten simulation runs using respectively the DSP, SSP and NGA policies, and report the average solution. Columns titled Obj, Res, Tard, and RejR, display the average objective value, the average response time, the average amount of time delay, and the rejection rate respectively.

Comparing the results of the three policies, the SSP clearly outperforms the DSP and the NGA, while the DSP performs better than the NGA, confirming that using optimal solutions at each decision epoch yields a better performance than a naive heuristic,

		DSP Policy				SSP Policy				NGA Policy			
Set	$\|K\|$	Obj	ResT	DelT	RejRate	Obj	ResT	DelT	RejRate	Obj	ResT	DelT	RejRate
1	10	182.75	93.91	36.52	30.86	137.00	73.11	20.90	20.87	205.47	103.73	43.75	38.02
2	12	158.13	91.12	34.52	18.63	105.89	66.01	16.08	10.92	183.07	100.00	40.07	26.89
3	14	118.57	78.91	25.69	7.16	72.00	54.04	9.06	3.76	157.51	94.19	34.60	16.78
4	16	57.41	49.11	7.59	0.29	43.77	38.95	3.13	0.66	117.45	79.65	23.71	7.17
5	18	31.08	30.27	0.82	0.00	27.77	26.80	0.61	0.13	62.14	51.97	7.96	0.92
6	20	23.10	22.93	0.17	0.00	20.85	20.65	0.17	0.01	29.56	28.71	0.79	0.02
Avg		95.17	61.04	17.55	9.49	67.88	46.59	8.33	6.06	125.87	76.37	25.15	14.97

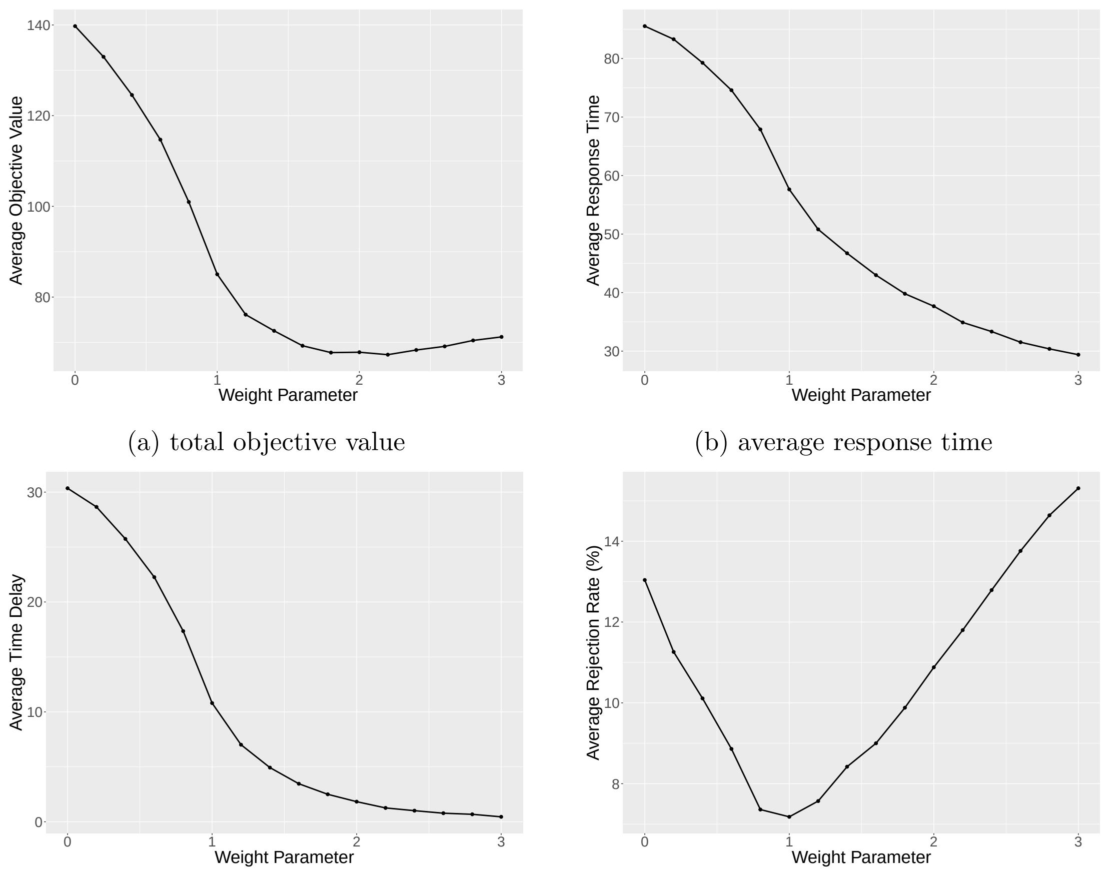
Table 5.5: Simulation results of using different re-optimisation policies

and future improvements are possible by exploiting stochastic information about future request. In the off-line experiments (Table 5.3), the SSP model finds better solution values than the DSP model, but it produces higher rejections numbers. In contrast, the simulation experiments show that the SSP policy improves the solution values found by the DSP, and also yields lower rejection rates. This indicates that the SSP policy has a much better performance on making the acceptance/rejection decision from a long-term perspective. With the increase on the number of technicians, the performance of each policy improves. However, there is still a significant gap between the quality of solutions identified by DSP and SSP policies.

5.6.3 Evaluating Weight Parameters via Simulation

To evaluate the effect of the weight parameter, we examine 16 different values of λ varying from 0 to 2 with a step size of 0.2, each tested under ten simulation runs using the SSP policy. The results are illustrated in the Figure 5.10, where Figure 5.10a shows the changes of the average objective values against the value of λ, and the other three present the values of the three components of the objective function.

The results suggest an inversely proportional relationship between the average response time and λ, and between the average time delay and λ. This indicates that when the objective function applies a large weight on the term concerning future tasks, the SSP model tends to keep more resources for future customers in order to provide quicker response once they materialise. However, this may lead to a high rejection rate of currently known tasks. This can be seen in Figure 5.10d, where the average rejection rate steadily increases as the value of λ is changed from 1 to 3. However, when $\lambda \leq 1$, the average rejection rate drops as the value of λ increases. This confirms that taking future requests into account can lead to a better resource management, lowering the overall rejection rate. Figure 5.10a shows that the average objective value drops quickly as the value of λ increases from 0 to 1, decreases slowly until reaches the minimum value at $\lambda = 2.2$, and then shows an upward trend for $\lambda > 2.2$ since the increase on the cost of rejecting tasks is greater than the savings on the response time and the amount of time delay. The setting $\lambda = 2.2$ provides the best trade-off between the service quality



(a) total objective value

(b) average response time

(c) average amount of time delay

(d) average rejection rate

Figure 5.10: Evaluation of the weight factor with varying values

(response time and time delay) and the rejection rate. However, if the cost of rejecting a task is increased, the optimal value of λ is expected to decrease accordingly.

5.6.4 Evaluating Route Size Constraints via Simulation

The off-line experiments in Section 5.5.3 indicates that the parameter α controls the strength of route size constraints, and therefore affects the solution quality, which deteriorates when α takes small values such as 1 or 2. The results also suggest that it is not necessary to use large values of α, since long routes that contain large number of tasks are usually infeasible or less competitive due to the nature of the problem considered. This section assesses the impact of route size constraints within a real-time environment. We examine five values of α, which are shown in Table 5.6, and perform ten random simulation runs for each value using the SSP policy, where the weight parameter λ is set equal to 2.2. The average solution values are reported in Table 5.6.

The results show that the average rejection rate for instances with $\alpha = 1$ is about 20% higher than those with $\alpha \geq 2$. This can be explained by the fact that when $\alpha = 1$, the model assigns only one task to each technician at each decision epoch, and the rest

α	Obj	ResT	DelT	RejRate
1	69.91	35.49	1.33	12.58
2	65.89	37.42	1.98	10.17
3	64.50	37.07	2.02	9.74
4	65.61	37.48	2.02	10.02
5	65.49	37.44	1.94	10.02
Average	66.28	36.98	1.86	10.51

Table 5.6: Running simulation with different values of α

of tasks that have not been scheduled are temporarily rejected. Once the deadline of acceptance has passed and they have not been assigned to technicians, the tasks have to be rejected permanently. Moreover, the simulation only reoptimises the scheduling plan whenever a new task arrives. When technicians complete all tasks assigned, they stay idle until the next reoptimisation epoch, at which point new tasks are assigned to them. Thus, if each technician is assigned only one task at each decision epoch, the technician is likely to wait a period of time before being assigned a new task again. Such idle waiting can be avoided by performing another reoptimisation whenever a technician completes the current task. Although $\alpha = 3$ yields the best performance, the percentage differences between the average solution value of $\alpha = 3$ and those of $\alpha = 2, 4$ and 5 are only about 2%. To assess the statistical significance between the solution values of these four groups, we conduct an analysis of variance (ANOVA) based on the individual results of the ten simulation runs. The p value of this test is 0.88, which indicates that there is no significant difference between the solution values of $\alpha = 2, 3, 4$ and 5. This is opposed to what we observed in the off-line experiments, where the solutions of $\alpha = 2$ and $\alpha = 3$ are much worse than those of $\alpha \geq 3$. This indicates that, in contrast to what is observed with off-line problems, allowing the model to construct long routes becomes less important since the re-optimisation is frequently performed, especially for instances with high degree of dynamism. Even if a long route is constructed at a decision epoch, this route is likely to be modified at later epochs. Moreover, as our model takes future requests into account, it tends to balance the workload between technicians and avoid constructing long routes which contain large number of tasks.

5.7 Conclusion

This paper has presented ways in which stochastic information about future requests can be taken into account in the dynamic workforce scheduling and routing problem. To this end, a stochastic set-partitioning model is described and integrated with a sampling-based approach to exploit the stochastic information about future requests. Two dominance rules and a set of routing size constraints are proposed to enhance the performance of the proposed model. The numerical tests show that the proposed model can handle realistic instances in short computational times without the need to resort to heuristic approaches. The effect of route size constraints is also investigated within a simulation framework. The results suggest that the effect of using tighter route size constrains is comparable with that of using weaker constraints. This implies that for problems with high degree of dynamism, applying route size constraints can significantly reduce the computational effort without affecting the solution quality too much. The performance of the proposed stochastic model is evaluated against a deterministic set-partitioning model, and a naive greedy heuristic. The results demonstrate that the proposed model provides significant improvement over the approaches that do not utilize stochastic information.

Chapter 6

Concluding Remarks

In this thesis, we have studied both static and dynamic versions of workforce scheduling and routing problems (WSRPs) arising in the emergency response services. The first section of this chapter summarizes the work of the thesis and reports the main research findings and contributions. We then discuss research directions that may be considered in future work.

6.1 Summary of the Work

This section is split into three subsections, each for one of the three objectives of our study.

6.1.1 Objective One

The first objective is to develop a simulation model of real-time emergency vehicle dispatching and routing. The literature review (Chapter 2) suggests that discrete event simulation (DES) is a suitable approach for modelling the problem at hand, for which reason we chose to build a DES model based on a case study of a British company providing emergency road services. The process of model development including data collection, data analysis, distribution fittings, model design as well as model verification and validation were described in detail in Chapter 3. The developed model was used to identify key characteristics of the real system, as illustrated below:

1. Under the current system setting, the average response time is around 45 minutes, the garage rate is about 15% and the average dispatching distance is about 13 km;
2. With different demand levels, the average response time varies between 20 minutes and 110 minutes;

3. Under the current staffing level, the system's capacity is around 12,000 requests per day. For any demands beyond this limit, the system has to fully rely on the garage option.

In addition, the simulation model was used to evaluate the effect of the response time limit T^{Max}. We found that compared to the model performance of using $T^{Max} = 120$ (default setting of the real system), using $T^{Max} = 90$ can lead to around 10% improvement on the average response time with the garage rate increases less than 1%. Finally, we assessed the performance of two dispatching policies: quickest response (QR) and shortest path (SP). The former reflects the dispatching policy used by the company, while the latter is a naive greedy algorithm which always dispatches the closest vehicle. The experimental results show that these two policies can lead to quite different solutions. Compared to the QR policy, the SP policy can produce nearly 30% improvement on the average dispatching distance, however, its average response time is 30% greater than that of the QR policy. This indicates that a better policy that considers both the response time and driving distance may be possible.

6.1.2 Objective Two

The objective two is to design algorithms that are efficient and effective in dealing with the static WSRP. As our literature review reveals that the majority of existing algorithms for WSRP are sophisticated and highly problem specific, we aimed to develop a fast and simple heuristic algorithm based on iterated local search (ILS) framework. The proposed ILS was also applied to solve the skill vehicle routing problem (Skill VRP). To the best of our knowledge, this is the first ILS approach for WSRP and Skill VRP.

The proposed ILS algorithm consists of three main components: initial solution construction, local search procedure and perturbation mechanism, which are combined into a multi-start framework. The initial solution construction uses a greedy heuristic which always inserts a task at the cheapest feasible position. The local search procedure includes an inter-route search operator and an intra-route search operator. We have examined different combinations of these two operators, and results show that the strategy of applying the intra-route search as a post-optimisation procedure on the locally optimal solution returned by the inter-route search gives the best performance in terms of efficiency and effectiveness. Our perturbation mechanism uses a random cross exchange operator, and the perturbation strength is controlled by an adaptive mechanism. Moreover, we proposed a simple mechanism for reducing the outsourcing cost, which was confirmed to be useful and effective by the computational experiments.

The proposed ILS was evaluated against a mixed integer programming (MIP) model and an existing adaptive larger neighbourhood search (ALNS) algorithm (Kovacs et al., 2012) on benchmark instances with up to 100 tasks. Computational experiments indicate that the proposed algorithm can produce solutions that are within an average gap of 1% to the optimal values in at most 40 seconds on average for all instances tested here. Compared to other heuristic approaches (Kovacs et al., 2012; Castillo-Salazar et al., 2015) for the similar problems, the proposed ILS has a relatively simple structure and a small number of parameters.

The proposed ILS algorithm was also applied to solve a set of Skill VRP instances, and results show that our algorithm is able to find optimal or near-optimal solutions in less than 0.5 seconds on average for all instances tested. Although the proposed algorithm is designed for solving the workforce scheduling and routing problem, it can be easily adapted to tackle other types of scheduling and routing problems.

6.1.3 Objective Three

The third objective is to investigate how to exploit stochastic information about future requests in order to improve decision making for dynamic WSRP. To address this problem, we developed a sampling-based model that incorporates stochastic knowledge about future requests. The proposed model uses a two-stage set-partitioning framework, where the first-stage is concerned with finding a set of feasible technician routes covering known requests, while the second-stage estimates the effect of the same routes with respect to future requests. To enhance the performance of the proposed model, we introduced two dominance rules and a set of route size constraints. We have examined alternative formations that are based on the vehicle-flow framework and the assignment model. However, they are not as competitive as the set-partitioning based models as indicated by the computational results.

The review of relevant studies shows that sampling-based algorithms are able to provide high quality solutions, but the computational effort increases dramatically with the number of scenarios, for which reason the existing studies have focused on the integration of sampling-based approaches into heuristic algorithms. In contrast, we showed that our model can handle realistic instances in short computational times without the need to resort to heuristic approaches.

We also investigated the effect of route size constraints within a simulation framework. The results suggest that the effect of using tighter route size constraints is comparable with that of using weaker constraints. This implies that for problems with high degree of dynamism, applying route size constraints can significantly reduce the computational effort without affecting the solution quality too much.

Finally, the performance of the proposed stochastic set-partitioning model was evaluated against a deterministic model and a naive greedy heuristic within a simulation framework, and tested on realistic instances generated using probability distributions derived from historical data. The computational results demonstrate that the proposed model provides significant improvement over the approaches that do not utilize stochastic information.

6.2 Limitations and Future Research

In chapter 3, we have presented a simulation model of real-time emergency vehicle dispatching and routing, where the model is driven by two simple greedy algorithms. Chapter 4 describes a fast and simple ILS algorithm that can find near-optimal solutions for instances with up to 100 tasks within one minute.

A question that naturally arises as to whether the integration of the ILS algorithm into the simulation model can lead to improvement. To address this question, we adapted the ILS algorithm to a dispatching policy and tested it on a scale-down model that covers a small area of the UK, since the ILS was not designed for problems with large sizes. The results show that the ILS algorithm does not offer improvement over the simple greedy heuristics. The reason is that the ILS algorithm was designed for static problems, where routes are typically constructed to accommodate a relatively high number of requests. Therefore, the ILS algorithm applies a variety of neighbourhood structures to explore different request ordering. However, the simulation model deals with a problem characterized by a high degree of dynamism, where each route often contains a small number of requests. In such situations, the ILS algorithm becomes less useful as finding the best request ordering is much easier. Therefore, the development of a more suitable metaheuristic algorithm for the simulation model could be considered in the future.

The sampling-based model described in Chapter 5 was tested within a simplified simulation model, which covers only the Southampton area. The results show that the proposed model is able to handle instances generated from the simplified simulation model in short computational times without the need to resort to heuristic approaches. However, the original simulation model (Chapter 3) considers a very large-scale problem (e.g., 10,000 requests per day and 2,500 vehicles), and it is also highly dynamic, which requires decisions to be made within a short time period. Therefore, it would require a new solution algorithm in order to apply the sampling-based model to the large simulation model.

In this thesis, we have considered static and dynamic aspects of the WSRP and also exploited the stochastic information about future requests. However, there still exist many areas that we have not explored. For example, uncertainties in traffic conditions, service times and skill requirements. Those uncertainties have been recognised as important elements affecting the decision making in emergency response services. Therefore, it would be useful to extend our model to consider the stochastic elements listed above.

Appendix A

A.1 Distribution Fitting for Preparation Duration

This section presents the results of fitting distributions to two samples $\hat{S}_1$ and $\hat{S}_2$. The former refers to the data sample of preparation durations of tasks requiring recovery 1 services, while the latter consists of the duration data of tasks requiring recovery 2 services.

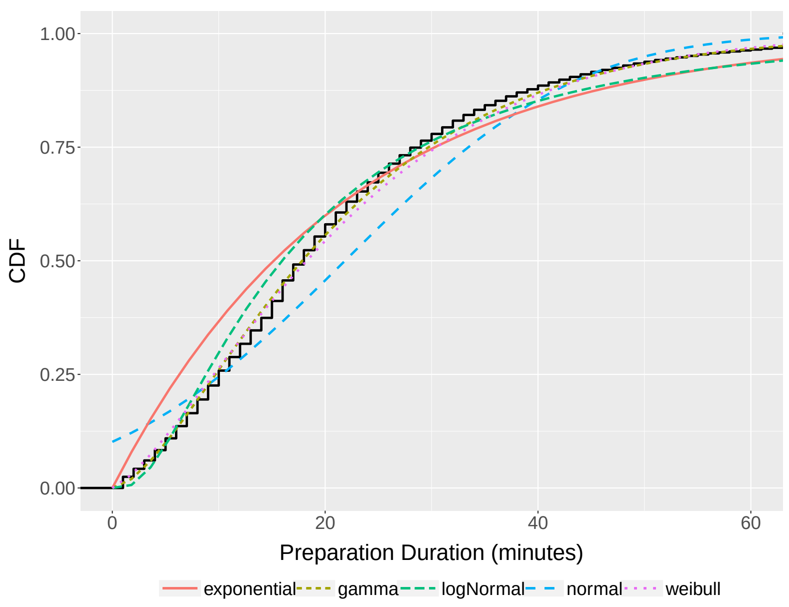


Figure A.1: Empirical CDF of sample $\hat{S}_1$ against theoretical CDFs of estimated distributions

In Figure A.1, the black line shows the empirical cumulative distribution function (CDF) of sample $\hat{S}_1$, while the coloured lines display theoretical CDFs of estimated distributions. Among all the parametric distributions considered, the gamma distribution seems to provide the best fit to $\hat{S}_1$.

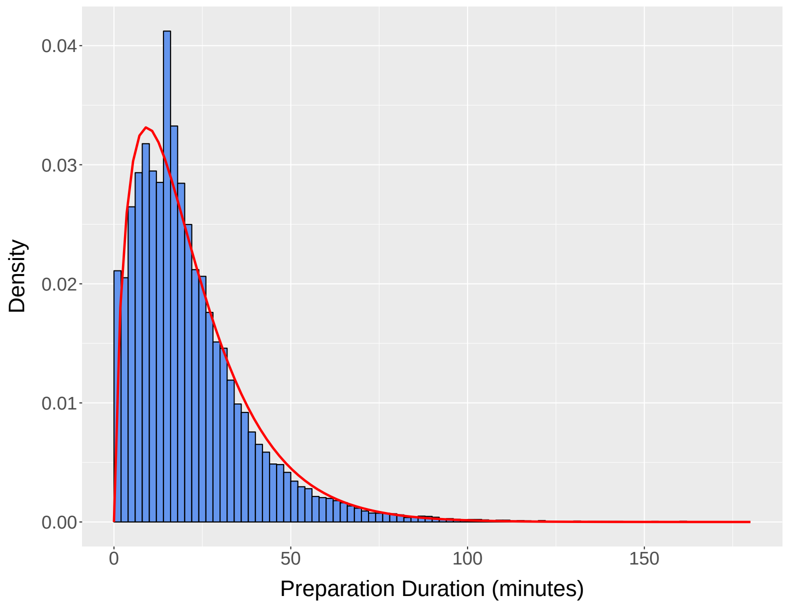


Figure A.2: PDF of sample $\hat{S}_1$ with a gamma distribution fitted

Figure A.2 displays the probability density function (PDF) of $\hat{S}_1$, with a gamma distribution (shape 1.73 and rate 0.08) fitted. It can be seen that the gamma distribution can provide reasonable fit to $\hat{S}_1$. Thus, we choose to use the gamma distribution to model the preparation duration of recovery 1 services.

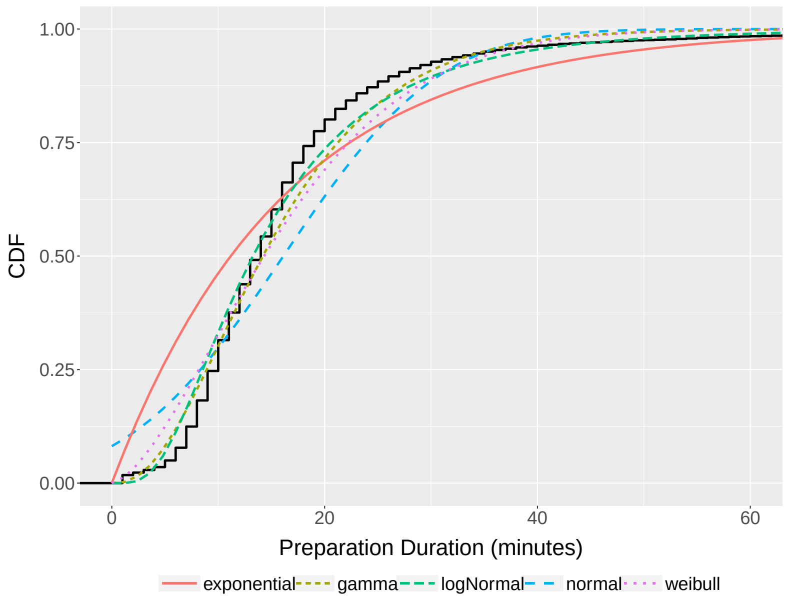


Figure A.3: Empirical CDF of sample $\hat{S}_2$ against theoretical CDFs of estimated distributions

Figure A.3 shows the empirical CDF of sample $\hat{S}_2$ and theoretical CDFs of estimated distributions. Among all the distribution considered, the gamma and log-normal distribution appear to give good fit to $\hat{S}_2$. However, it is difficult to determine which one of them has a better fit. Thus, we use the QQ plot to compare the fitting performance of the gamma and log-normal distributions to $\hat{S}_2$. The plot in Figure A.4 indicates that the log-normal has a better fit than the gamma distribution.

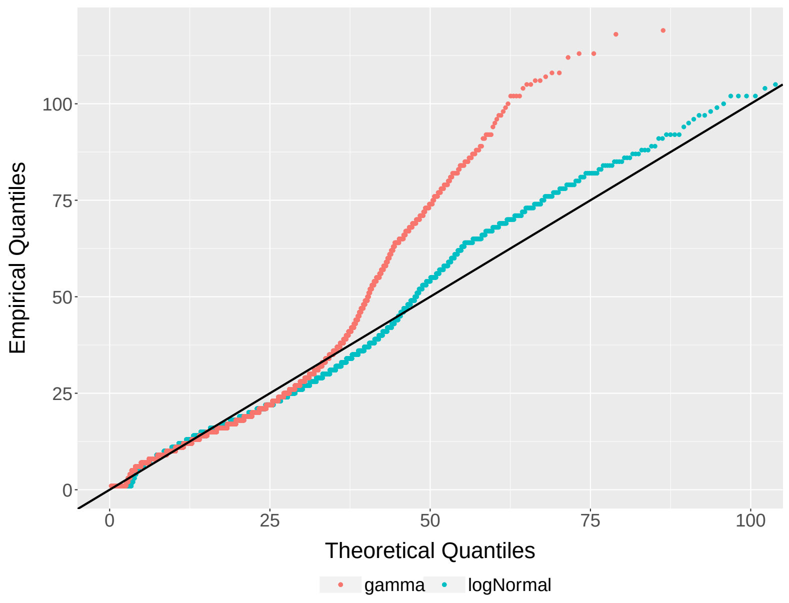


Figure A.4: QQ plot of sample $\hat{S}_2$ against the gamma and log-normal distribution estimated

Figure A.5 presents the PDF of $\hat{S}_2$, fitting with a log-normal distribution, of which the parameters are estimated to be 2.58 and 0.65. It can be seen that the log-normal distribution is able to give a reasonable fit to $\hat{S}_2$. Thus, we use this distribution to model the preparation duration of recovery 2 services.

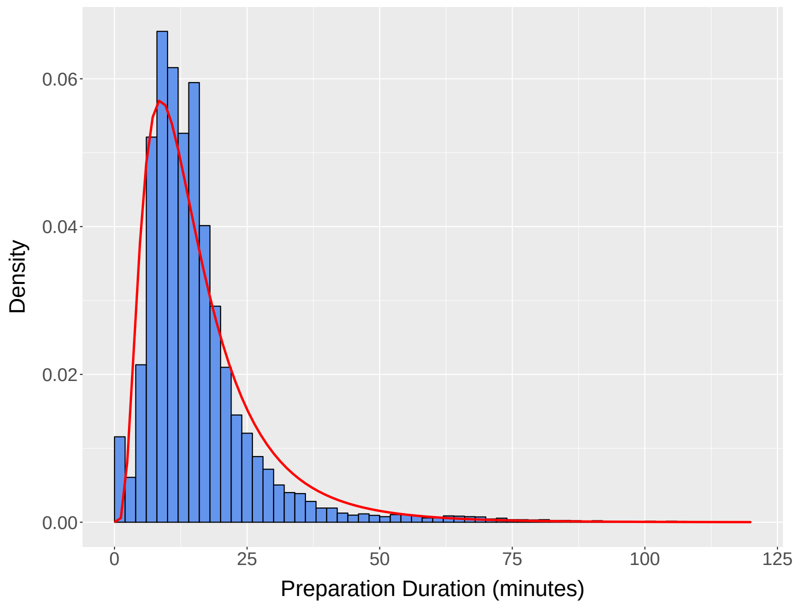


Figure A.5: PDF of sample $\hat{S}_2$ with a log-normal distribution fitted

A.2 Distribution Fitting for Recovery Distance

This section presents the results of fitting distributions to the sample $\hat{D}_2$, which consists of the distance data of tasks requiring recovery 2 services.

Figure A.6 shows the empirical CDF of $\hat{D}_2$ and theoretical CDFs of estimated distributions. It can be seen that the lines of the gamma and weibull distribution are close to the one representing the empirical CDF of $\hat{D}_2$, and they have similar shapes. Apart from these two distributions, the rest do not present a good fit to $\bar{D}_2$.

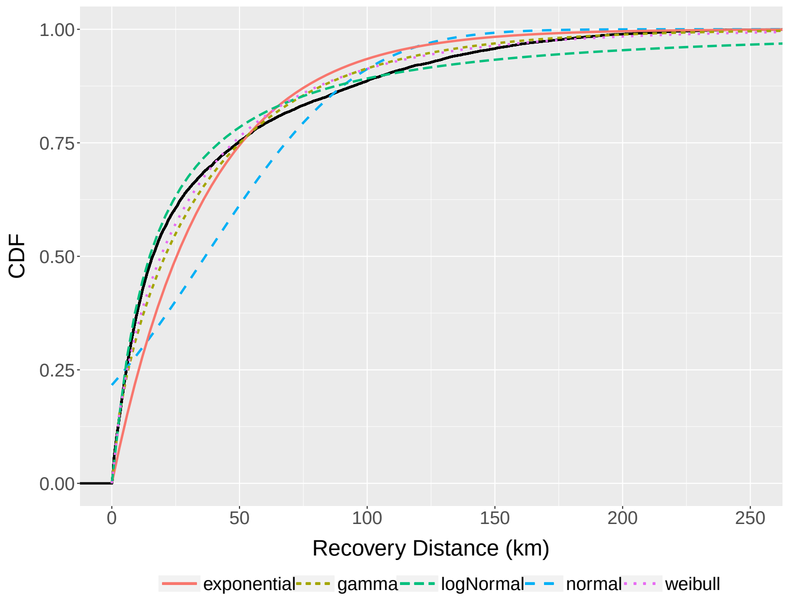


Figure A.6: Empirical CDF of sample $\bar{D}_2$ against theoretical CDFs of estimated distributions

We use the QQ plot to compare the fitting of the gamma and weibull distribution to sample $\bar{D}_2$. The figure shows that the points from both distributions are fairly close to the straight line, except in the extreme tails. This means that the proposed distributions have higher probability of having large values than the actual distribution of $\bar{D}_2$. Since there is no suitable parametric distribution for $\bar{D}_2$, we choose to construct an empirical distribution, which is shown in Figure A.8.

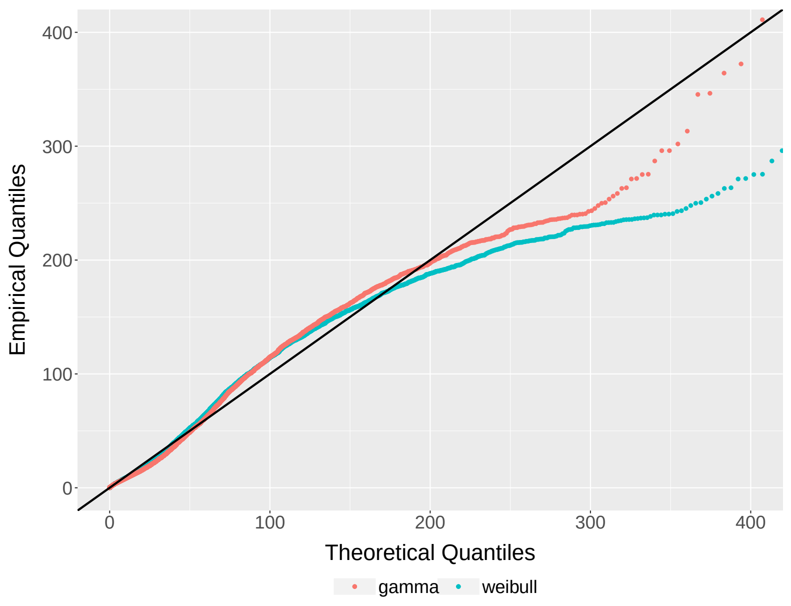


Figure A.7: QQ plot of sample $\bar{D}_2$ against the gamma and weibull distribution estimated

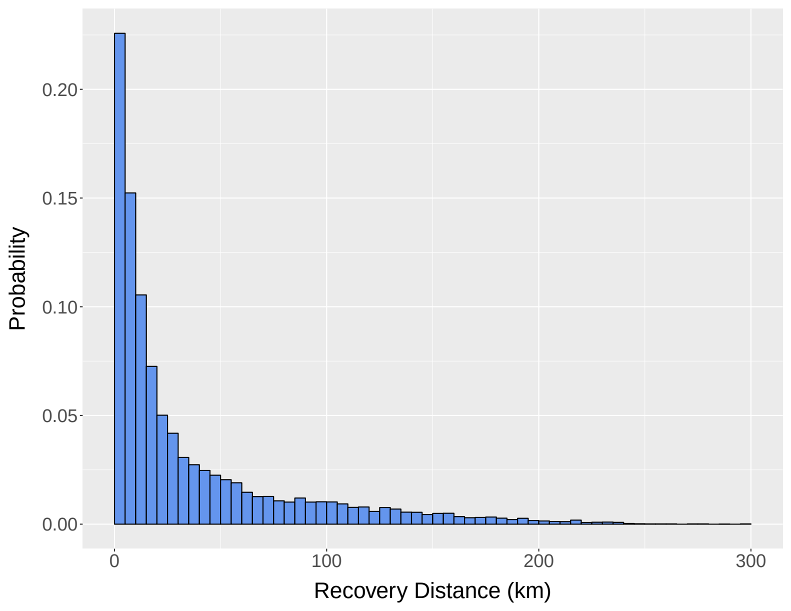


Figure A.8: Probability distribution of the empirical distribution fitted for $\bar{D}_2$

A.3 Results of Determining the Warm-Up Period

This appendix contains additional results from the experiments of determining the warm-up period.

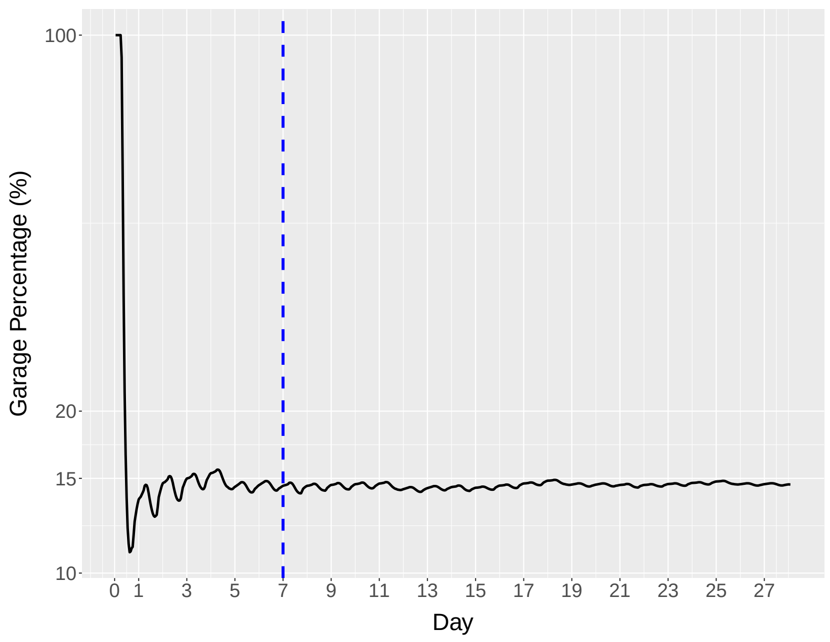


Figure A.9: Time-series of garage percentage from ten replications

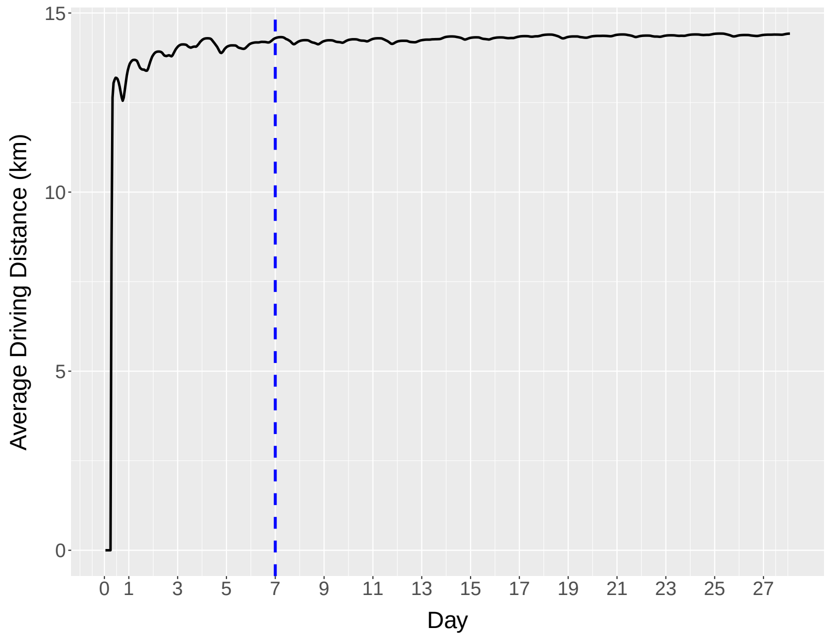


Figure A.10: Time-series of average driving distance from ten replications

A.4 Results of Selecting the Number of Replications

This appendix contains additional results from the experiments of selecting the number of replications.

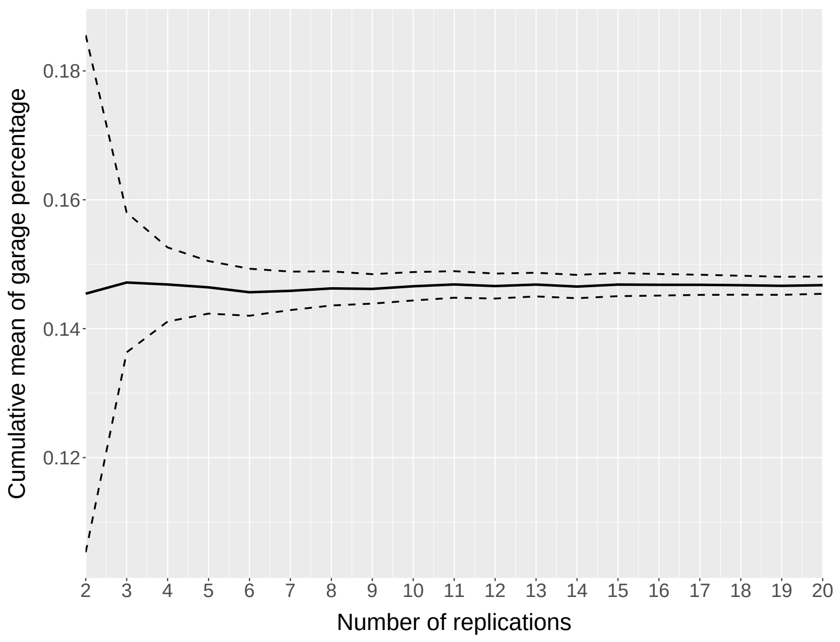


Figure A.11: Cumulative mean of garage percentage with 95% confidence intervals

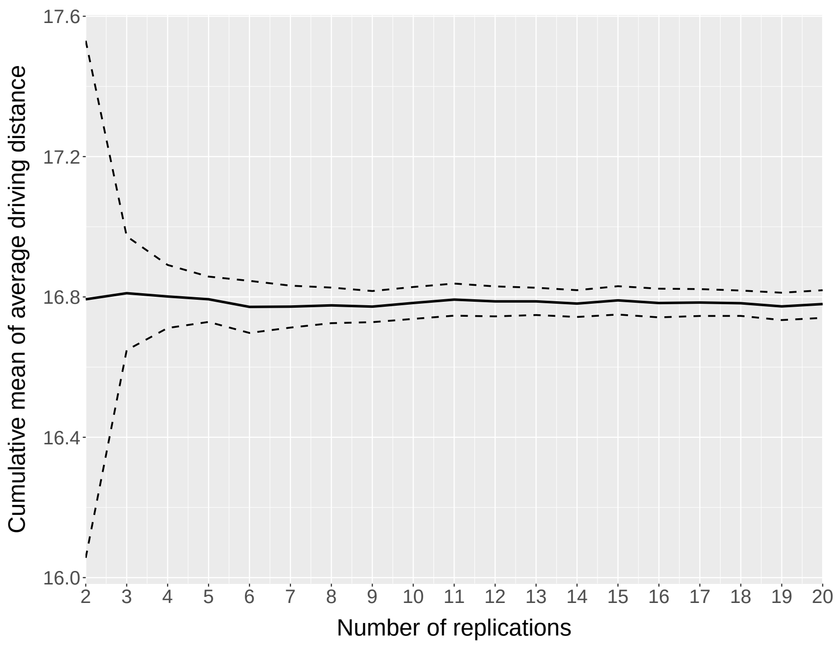


Figure A.12: Cumulative mean of average driving distance with 95% confidence intervals

Appendix B

B.1 Move Evaluation

This section describes the move evaluation method proposed by Vidal et al. (2013b), which can be applied to compute the violations of duration and time window constraints of the routes in amortized $O(1)$ time for most classical neighbourhood operators. Their method was inspired by the following observation: any neighbourhood move can be viewed as a separation of routes into subsequences, which are then concatenated into new routes. Therefore, the proposed procedure requires a preprocessing phase to develop the following data for each subsequence σ, containing visits to depots or customers:

- The accumulated distance $C(\sigma)$;
- The minimum duration $D(\sigma)$;
- The minimum time-warp use $TW(\sigma)$, which measures the time window violation in our study;
- The earliest time $E(\sigma)$ and the latest time $L(\sigma)$ to visit the first vertex allowing a schedule with minimum duration and minimum time-warp use.

For a sequence σ containing a single vertex i, the above data can be easily obtained as $C(\sigma) = 0$, $D(\sigma) = d_i$, $TW(\sigma) = 0$, $E(\sigma) = e_i$ and $L(\sigma) = l_i$. Then, we can compute the same data on concatenations of sequences based on the following proposition:

Proposition *Concatenation of two sequences* (Vidal et al., 2013b). Let $\sigma = (i, ..., j)$ and $\sigma' = (i', ..., j')$ be two subsequences of visits. The concatenated subsequence $\sigma \oplus \sigma'$

is characterised by the following data:

$$D(\sigma \oplus \sigma') = D(\sigma) + D(\sigma') + t_{ji'} + \Delta_{WT} \tag{B.1}$$

$$TW(\sigma \oplus \sigma') = TW(\sigma) + TW(\sigma') + \Delta_{TW} \tag{B.2}$$

$$E(\sigma \oplus \sigma') = \max\{E(\sigma') - \Delta, E(\sigma)\} - \Delta_{WT} \tag{B.3}$$

$$L(\sigma \oplus \sigma') = \min\{L(\sigma') - \Delta, L(\sigma)\} + \Delta_{TW} \tag{B.4}$$

$$C(\sigma \oplus \sigma') = C(\sigma) + C(\sigma') + c_{ji'} \tag{B.5}$$

where $\Delta = D(\sigma) - TW(\sigma) + t_{ji'}$, $\Delta_{WT} = \max\{E(\sigma') - \Delta - L(\sigma), 0\}$ and $\Delta_{TW} = \max\{E(\sigma) + \Delta - L(\sigma'), 0\}$.

With the above proposition and the relevant data developed in the preprocessing phase, we can evaluate the costs, as well as the violations of duration and time window constraints of the new routes generated by the neighbourhood moves in constant time.

Appendix C

C.1 Deterministic Vehicle Flow Formulation (DVF)

This section presents a deterministic vehicle flow (DVF) model proposed for the offline WSRP. We define the problem on a complete graph $G = (V, A)$, where $V = \{0, 1, ..., n, n+1\}$ is a set of vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is a set of arcs. The vertices 0 and $n+1$ are two dummy depots, and $C = V \setminus \{0, n+1\}$ represents the set of vertices that each has a unique task. Depending on the context, we refer to a task i or a vertex i for any $i \in C$. All feasible routes are defined by starting at the dummy depot 0, visiting a subset of customer vertices and ending at the dummy depot $n+1$. Let t_{ij}, $\forall i, j \in C$ be the travel time between task vertices i and j. Since each route actually starts at the technician's current location, the travel time from the dummy depot 0 to any vertex $j \in C$ is technician-dependent, denoted by t^k_{0j}. For arcs that ends at the dummy depot $n+1$, the associated travel time are set to 0. Figure C.1 shows an example of two feasible routes. Vertices 1 and 6 are the two dummy depots and vertices {1,..,5} represent 5 different tasks. The two solid lines represent two actual routes carried out by two technicians, where one route starts from the location of technician 1 and visits tasks 1 and 4, and the other starts from the location of technician 2 and visits tasks 2, 3 and 5. However, both routes are assumed to start at the dummy depot 1 and end at the dummy depot 2, as indicated by dotted lines. Thus, the travel times of arcs (0,1) and (0,2) are set to the travel times from technician 1 to vertex 1 and technician 2 to vertex 2, respectively. The travel times of arcs ending at vertex 6 are set to 0. Finally, we introduce a binary parameter q^k_i, $\forall i \in C, \forall k \in K$ to model the skill compatibility between technician k and task i, where $q^k_i = 1$ if technician k is qualified to perform task i, and 0 otherwise.

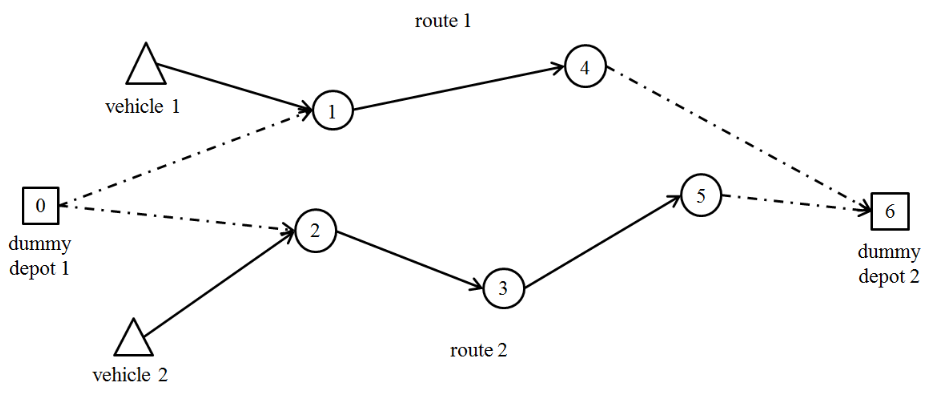


Figure C.1: Illustration of feasible routes

The problem can be formulated as a MIP model with following binary variables:

$$x_{ij}^k = \begin{cases} 1 & \text{if technician } k \text{ travels directly from vertex } i \text{ to } j, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall k \in K,\ i,j \in V, i \neq j,$$

$$z_i = \begin{cases} 1 & \text{if task } i \text{ is rejected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in C,$$

and non-negative continuous variables b_i, w_i and u_i, $\forall i \in C$, that denote respectively the time at which service commences, the response time and the amount of time delay at task i. The mathematical model is presented as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in C} w_i + \beta \sum_{i \in C} u_i + \sum_{i \in C} o_i z_i && && \text{(C.1)} \\
\text{subject to:} \quad & \sum_{k \in K} \sum_{j \in V \setminus \{0\}} x_{ij}^k + z_i = 1 && \forall i \in C && \text{(C.2)} \\
& \sum_{j \in V} x_{ij}^k \leq q_i^k && \forall k \in K, \forall i \in C && \text{(C.3)} \\
& \sum_{i \in C} \sum_{j \in V \setminus \{0\}} x_{ij}^k \leq \alpha && \forall k \in K && \text{(C.4)} \\
& \sum_{j \in V \setminus \{0\}} x_{0j}^k = 1 && \forall k \in K && \text{(C.5)} \\
& \sum_{i \in V \setminus \{n+1\}} x_{i,n+1}^k = 1 && \forall k \in K && \text{(C.6)} \\
& \sum_{i \in V \setminus \{n+1\}} x_{ih}^k - \sum_{j \in V \setminus \{0\}} x_{hj}^k = 0 && \forall k \in K,\ \forall h \in C && \text{(C.7)} \\
& e_0^k + t_{0j}^k \leq b_j + M_{0j}^k (1 - x_{0j}^k) && \forall k \in K,\ \forall j \in C && \text{(C.8)} \\
& b_i + d_i + t_{ij} \leq b_j + M_{ij} (1 - x_{ij}^k) && \forall k \in K,\ \forall i, j \in C && \text{(C.9)} \\
& b_i \geq e_i && \forall i \in C && \text{(C.10)} \\
& w_i \geq b_i - e_i && \forall i \in C && \text{(C.11)} \\
& u_i \geq b_i - l_i && \forall i \in C && \text{(C.12)} \\
& u_i \leq U^{MAX} && \forall i \in C && \text{(C.13)} \\
& u_i \geq 0 && \forall i \in C && \text{(C.14)} \\
& z_i = 0 && \forall i \in C' && \text{(C.15)} \\
& x_{ij}^k \in \{0, 1\} && \forall k \in K,\ \forall (i, j) \in A && \text{(C.16)} \\
& z_i \in \{0, 1\} && \forall i \in C. && \text{(C.17)}
\end{aligned}
$$

The objective function (C.1) minimizes the total cost comprising the response times, delay penalty, and cost of rejecting tasks. Constraints (C.2) impose that each task is either visited exactly once by a technician or rejected. Constraints (C.3) ensure that the tasks can only be carried out by technicians satisfying the skill requirements, and (C.4) are the route size constraints. Constraints (C.5) and (C.6) enforce that each technician departs from the dummy depot 0 and returns to the dummy depot $n+1$. Constraints (C.7) are the typical flow conservation equations. Constraints (C.8) and (C.9) set the time variables b_i, where M_{0j}^k and M_{ij} are large constants which guarantee the schedule feasibility with respect to time considerations when $x_{0j}^k = 0$ and $x_{ij}^k = 0$ respectively. Constraints (C.10), (C.11), (C.12), (C.13) and (C.14) specify the ranges of variables b_i, w_i and u_i. Constraints (C.15) prevent rejection of tasks in the subset C'. Constraints (C.16) and (C.17) represent the binary restrictions on variables x_{ij}^k and z_i.

The large constants M_{0j}^k can be replaced by $\max\{e_0^k + t_{0j}^k - e_j, 0\}$, and constraints (C.8) need only be imposed for arcs with $M_{0j}^k > 0$, because when $\max\{e_0^k + t_{0j}^k - e_j, 0\} = 0$, constraints (C.8) can be written as $e_0^k + t_{0j}^k < b_j$, which are satisfied for any values of b_j and x_{0j}^k since $e_0^k + t_{0j}^k \leq e_j$ and $e_j \leq b_j$. In terms of large constants M_{ij}, they can be replaced by $\max\{l_i + U^{MAX} + d_i + t_{ij} - e_j, 0\}$, and we only need to consider constraints (C.9) for arcs having $M_{ij} > 0$, because when $M_{ij} = 0$, constraints (C.9) are equivalent to $b_i + d_i + t_{ij} \leq b_j$, and they are always satisfied because of inequalities $b_i \leq l_i + U^{MAX}$, $l_i + U^{MAX} + d_i + t_{ij} \leq e_j$ and $e_j \leq b_j$.

C.2 Comparison of DVF and DSP Model

The comparison of the DVF and DSP models are conducted using instances containing up to 50 tasks. For each instance, we test three resource ratios, namely $\theta = \{0.2, 0.5, 0.8\}$, which correspond to low, median and high staffing level. The results are displayed in Table C.1. In addition to the columns already described in Section **??**, we use columns titled Cons and IG to show, for each instance, the number of constraints and the integrality gap (the percentage gaps of the best integer solutions to the linear relaxation solutions).

			DVF Model					DSP Model				
Instance	$\|C\|$	θ	Var	Cons	Obj	IG	T(s)	Var	Cons	Obj	IG	T(s)
1	10	0.2	263	286	1812.37	0.00	0.38	55	12	1812.37	0.00	0.00
2	10	0.5	596	625	1011.24	0.00	5.48	125	15	1011.24	0.00	0.00
3	10	0.8	929	964	713.05	0.00	19.38	208	18	713.05	0.00	0.00
4	20	0.2	1765	1812	3440.51	56.46	3600.00	829	24	3440.51	0.00	0.02
5	20	0.5	4291	4350	1486.19	67.14	3600.00	3669	30	1471.23	0.00	0.05
6	20	0.8	6817	6888	1147.31	68.53	3600.00	5843	36	1139.79	0.00	0.03
7	30	0.2	5707	5778	5065.19	63.36	3600.00	3329	36	5014.01	0.00	0.08
8	30	0.5	14086	14175	2240.27	98.72	3600.00	13914	45	1759.71	0.00	0.47
9	30	0.8	22465	22572	1381.11	99.61	3600.00	20142	54	1212.66	0.00	0.11
10	40	0.2	13289	13384	7320.34	66.95	3600.00	4669	48	6722.15	0.00	0.06
11	40	0.5	32981	33100	5212.66	99.69	3600.00	11849	60	2957.06	0.00	0.25
12	40	0.8	52673	52816	3559.33	99.76	3600.00	19289	72	2022.16	0.00	0.11
13	50	0.2	25711	25830	8964.14	66.48	3600.00	49616	60	7052.32	0.00	0.83
14	50	0.5	63976	64125	3497.13	100.00	3600.00	117956	75	2666.54	0.00	1.15
15	50	0.8	102241	102420	5506.91	100.00	3600.00	212368	90	1983.97	0.00	1.23
Average			23186.00	23275.00	3490.52	65.78	2881.68	30924.07	45.00	2731.92	0.00	0.29

Table C.1: The comparison of the DVF and DSP models

Of the 15 instances, the DVF model is only able to find optimal solutions for three, while the DSP model can solve all instances to optimality using an average time of 0.29 seconds. For instance 4, both models find exactly the same solution value, however the DVF model is not able to prove the optimality within the given time limit. Except large instances containing 50 tasks, the numbers of variables of the DSP model are much less than that of the DVF model. This implies that the hard time window constraints (the amount of time delay must be less than U^{MAX}) and the route size constraints significantly restrict the number of feasible routes. Moreover, the DSP model has only a very small number of constraints, which is equal to the number of tasks $|C|$ plus the number of technicians $|K|$. The number of constraints of the DVF model is similar to the number of variables, and they increase significantly with the size of instances. Therefore, the DSP model clearly outperforms the DVF model for the problem considered in this paper.

C.3 Stochastic Set-Partitioning and Assignment Model

In addition to the parameters and variables already defined for the DSP model, we introduce a binary parameter $q_j^r, \forall r \in R, j \in C^+$ to describe the skill compatibility between technician route r and a future task j, where $q_j^r = 1$ if the associated technician of route r is qualified to perform task j, and 0 otherwise. Moreover, we define the following binary variables:

$$y_{rj} = \begin{cases} 1 & \text{if future task } j \text{ is assigned to route } r, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall j \in C^+,\ r \in R;$$

$$z_j^+ = \begin{cases} 1 & \text{if future task } j \text{ is rejected,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall j \in C^+.$$

and the continuous variables w_j^+ and u_j^+, $\forall j \in C^+$, that denote respectively the response time and the amount of time delay at a future task j.

The proposed formulation is presented below:

$$\text{minimize} \quad \sum_{r\in R}(w_r + \beta u_r)x_r + \sum_{i\in C} o_i z_i + \lambda \sum_{s\in S} p_s\Big(\sum_{j\in C_s^+} w_j^+ + \beta' \sum_{j\in C_s^+} u_j^+ + \sum_{j\in C_s^+} o_j' z_j^+\Big) \tag{C.18}$$

$$\text{subject to:} \quad (5.6)\text{–}(5.10) \text{ and}$$

$$\sum_{j\in C_s^+} y_{rj} \le x_r \qquad \forall r \in R, \forall s \in S \tag{C.19}$$

$$\sum_{j\in C_s^+} y_{rj} \le q_j^r \qquad \forall r \in R, \forall s \in S \tag{C.20}$$

$$\sum_{r\in R} y_{rj} + z_j = 1 \qquad \forall j \in C^+ \tag{C.21}$$

$$w_j^+ \ge (c_r + t_{rj} - e_j)y_{rj} \qquad \forall j \in C^+, \forall r \in R \tag{C.22}$$

$$w_j^+ \ge t_{rj}y_{rj} \qquad \forall j \in C^+, \forall r \in R \tag{C.23}$$

$$u_j^+ \ge (c_r + t_{rj} - l_j)y_{rj} \qquad \forall j \in C^+, \forall r \in R \tag{C.24}$$

$$u_j^+ \ge (e_j + t_{rj} - l_j)y_{rj} \qquad \forall j \in C^+, \forall r \in R \tag{C.25}$$

$$u_j^+ \ge 0 \qquad \forall j \in C^+ \tag{C.26}$$

$$z_j^+ \in \{0,1\} \qquad \forall j \in C^+ \tag{C.27}$$

$$y_{rj} \in \{0,1\} \qquad \forall r \in R, \forall j \in C^+ \tag{C.28}$$

The objective function (C.18) minimises the first-stage cost plus the expected second-stage cost weighted by parameter λ. Constraints (5.6) to (5.10) are taken from the DSP model. Constraints (C.19) guarantee that a future task can be assigned to a route only if the route is selected in the first-stage, while (C.20) ensure that a future task can only be assigned to a technician route satisfying the skill requirements. Constraints (C.21) impose that each future task is either assigned to a route or rejected. Constraints (C.22), (C.23), (C.24), (C.25) and (C.26) set the lower bounds of variables w_j and u_j, $\forall j \in C^+$. Constraints (C.27) and (C.28) are the binary restrictions on the decision variables.

C.3.1 Comparison of SSPA and SSP Model

The comparison of the SSPA and SSP model is performed based on the instances containing up to 30 tasks. Each instance is solved with a set of scenarios S, where $|S| \in \{1, 5, 10, 15, 20\}$. The resource ratio is set to 0.5, which corresponds to a median staffing level. The results are presented in Table C.2.

Of the 15 instances tested, the SSPA model is only able to find optimal solution for one instance within the one hour time limit, while the SSP model can solve all instances to optimality using an average computational time of 11.50 seconds. Moreover, the SPP model has significantly less numbers of variables and constraints than those of the SSPA

			SSPA Model					SSP Model				
No.	$\|C\|$	$\|S\|$	Var	Cons	Obj	IG	T(S)	Var	Cons	Obj	IG	T(S)
1	10	1	2606	4755	1314.19	0.00	64.62	1498	450	1314.19	0.00	0.03
2	10	5	11266	23715	1340.69	11.62	3600.00	5817	2190	1340.69	0.00	0.77
3	10	10	22091	47415	1422.89	18.42	3600.00	9312	4365	1422.89	0.00	0.56
4	10	15	32916	71115	1413.15	19.59	3600.00	14456	6540	1408.86	0.00	0.62
5	10	20	43741	94815	1402.74	24.08	3600.00	20315	8715	1371.58	0.00	1.01
6	20	1	19235	36674	2648.12	27.50	3600.00	5582	1784	2612.18	0.00	0.13
7	20	5	89115	183250	2943.90	38.81	3600.00	23451	8800	2718.64	0.00	1.31
8	20	10	176465	366470	2767.74	36.14	3600.00	52242	17570	2635.49	0.00	2.17
9	20	15	263815	549690	3510.87	48.71	3600.00	72041	26340	2665.13	0.00	4.09
10	20	20	351165	732910	3825.91	52.68	3600.00	95650	35110	2669.03	0.00	6.41
11	30	1	127420	246804	4577.22	33.20	3600.00	46280	8019	4341.94	0.00	0.94
12	30	5	605140	1233840	5517.91	43.42	3600.00	181119	39915	4415.67	0.00	5.49
13	30	10	1202290	2467635	13500.00	100.00	3600.00	341313	79785	4396.10	0.00	13.62
14	30	15	1799440	3701430	13500.00	100.00	3600.00	520068	119655	4396.99	0.00	31.95
15	30	20	2396590	4935225	13500.00	100.00	3600.00	689695	159525	4406.63	0.00	103.37
Avg.			476219.67	979716.20	4879.02	43.61	3364.31	138589.27	34584.20	2807.73	0.00	11.50

Table C.2: The comparison of the SSPA and SSP Model

model. Therefore, the SSP model clearly outperforms the SSPA model in terms of the solution quality and computational speed.

C.4 Pseudo-code of Test Instance Generation

Algorithm 4 Construct an instance containing N tasks

```
1: Input: Set $Z$ of demand zones, each $j \in Z$ defined by the coordinates of its bottom-
   left corner $(X_j, Y_j)$ and an associated density factor $D_j$.
2: procedure GENERATE RANDOM TASKS
3:     Initialize the arrival rates: $\bar{a} \leftarrow N$ and $a_j \leftarrow D_j\bar{a}, j \in Z$
4:     $T^{Start} \leftarrow 0$, $T^{Limit} \leftarrow 1$ and $LS \leftarrow$ empty task list
5:     while $|LS| < N$ do
6:         for (each demand zone $j$, $j \in Z$) do
7:             $T^{Cur} \leftarrow 0$
8:             while $(T^{Cur} < T^{Limit})$ do
9:                 $t \leftarrow$ generate_exponential_distribution $(a_j)$
10:                $T^{Cur} \leftarrow T^{Cur} + t$
11:                if $(T^{Cur} < T^{Limit})$ then
12:                    Create a task $i$ with time window $e_i = T^{Start} + T^{Cur}$, $l_i = e_i + T^{Tar}$
13:                    $rand_1, rand_2 \leftarrow$ generate_random $[0, 10]$
14:                    Coordinates: $x_i = X_j + rand_1$ and $y_i = Y_j + rand_2$
15:                    Service duration: $d_i \leftarrow$ generate_empirical_distribution
16:                    Insert task $i$ to $LS$
17:                end if
18:            end while
19:        end for
20:        Sort list $LS$ in ascending order by the earliest times of tasks
21:        $T^{Start} \leftarrow$ the arrival time of the last task on $LS$
22:    end while
23:    if $(|LS| > N)$ then
24:        Remove the last $|LS| - N$ tasks from the task list $LS$
25:    end if
26:    return $LS$
27: end procedure
```

Bibliography

Abounacer, R., Rekik, M., and Renaud, J. (2014). An exact solution approach for multi-objective location–transportation problem for disaster response. *Computers & Operations Research*, 41:83–93.

Adulyasak, Y. and Jaillet, P. (2015). Models and algorithms for stochastic and robust vehicle routing with deadlines. *Transportation Science*, 50(2):608–626.

Akjiratikarl, C., Yenradee, P., and Drake, P. R. (2007). PSO-based algorithm for home care worker scheduling in the UK. *Computers & Industrial Engineering*, 53(4):559–583.

Alfares, H. K. (2004). Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1-4):145–175.

Alsalloum, O. I. and Rand, G. K. (2006). Extensions to emergency vehicle location models. *Computers & Operations Research*, 33(9):2725–2743.

Altay, N. and Green, W. G. (2006). OR/MS research in disaster operations management. *European Journal of Operational Research*, 175(1):475–493.

Andersson, T. and Värbrand, P. (2007a). Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58(2):195–201.

Andersson, T. and Värbrand, P. (2007b). Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58(2):195–201.

Attanasio, A., Cordeau, J.-F., Ghiani, G., and Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2012). A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1):103–112.

BañOs, R., Ortega, J., Gil, C., Fernández, A., and De Toro, F. (2013). A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40(5):1696–1707.

Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A. (2004). An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Baldacci, R., Toth, P., and Vigo, D. (2007). Recent advances in vehicle routing exact algorithms. *4OR: A Quarterly Journal of Operations Research*, 5(4):269–298.

Ballou, R. H., Rahardja, H., and Sakai, N. (2002). Selected country circuity factors for road travel distance estimation. *Transportation Research Part A: Policy and Practice*, 36(9):843–848.

Banks, J., Carson II, J. S., Barry L., N., and Nicol, D. M. (2005). *Discrete-Event System Simulation*. Prentice Hall, New Delhi, India.

Barbarosoğlu, G., Özdamar, L., and Çevik, A. (2002). An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. *European Journal of Operational Research*, 140(1):118–133.

Bektaş, T., Repoussis, P. P., and Tarantilis, C. D. (2014). Dynamic vehicle routing problems. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, chapter 11, pages 299–347. SIAM, Philadelphia, PA.

Bent, R. and Van Hentenryck, P. (2007). Waiting and relocation strategies in online stochastic vehicle routing. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1816–1821, Hyderabad, India.

Bent, R. W. and Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

Berger, J. and Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 31(12):2037–2053.

Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866–2890.

Bertsimas, D. (1988). *Probabilistic combinatorial optimization problems*. Thesis, Massachusetts Institute of Technology, Department of Mathematics.

Biesinger, B., Hu, B., and Raidl, G. (2016). An integer L-shaped method for the generalized vehicle routing problem with stochastic demands. *Electronic Notes in Discrete Mathematics*, 52:245–252.

Binart, S., Dejax, P., Gendreau, M., and Semet, F. (2016). A 2-stage method for a field service routing problem with stochastic travel and service times. *Computers & Operations Research*, 65:64–75.

Birge, J. R. and Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer, New York.

Blais, M., Lapierre, S. D., and Laporte, G. (2003). Solving a home-care districting problem in an urban setting. *Journal of the Operational Research Society*, 54(11):1141–1147.

Blum, C., Puchinger, J., Raidl, G., and Roli, A. (2010). A brief survey on hybrid metaheuristics. In Filipič, B. and Šilc, J., editors, *Proceedings of 4th International Conference on Bioinspired Optimization Methods and their Applications*, pages 3–18, Ljubljana, Slovenia.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York.

Boyer, V., Gendron, B., and Rousseau, L.-M. (2014). A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. *Journal of Scheduling*, 17(2):185–197.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Branchini, R. M., Armentano, V. A., and Løkketangen, A. (2009). Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, 36(11):2955–2968.

Branke, J., Middendorf, M., Noeth, G., and Dessouky, M. (2005). Waiting strategies for dynamic vehicle routing. *Transportation Science*, 39(3):298–312.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368.

Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139.

Burke, E. K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J. A., and Gendreau, M. (2010). Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Barcelona, Spain.

Calvete, H. I., Galé, C., Oliveros, M.-J., and Sánchez-Valverde, B. (2007). A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research*, 177(3):1720–1733.

Cappanera, P., Gouveia, L., and Scutellà, M. G. (2011). The skill vehicle routing problem. In Pahl, J., Reiners, T., and Voß, S., editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 354–364. Springer, Berlin, Heidelberg.

Caramia, M. and Giordani, S. (2009). A new approach for scheduling independent tasks with multiple modes. *Journal of Heuristics*, 15(4):313–329.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2012). A survey on workforce scheduling and routing problems. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 283–302, Son, Norway.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2015). A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints. In *Proceedings of the 4th International Conference on Operations Research and Enterprise Systems*, pages 367–375, Lisbon, Portugal.

Caunhye, A. M., Nie, X., and Pokharel, S. (2012). Optimization models in emergency logistics: A literature review. *Socio-Economic Planning Sciences*, 46(1):4–13.

Chen, P., Huang, H.-K., and Dong, X.-Y. (2010). Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications*, 37(2):1620–1627.

Chen, X., Thomas, B. W., and Hewitt, M. (2015). The technician routing problem with experience-based service times. *Omega*, 61:49–61.

Chen, Z.-L. and Xu, H. (2006). Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88.

Chern, C.-C., Chen, Y.-L., and Kung, L.-C. (2010). A heuristic relief transportation planning algorithm for emergency supply chain management. *International Journal of Computer Mathematics*, 87(7):1638–1664.

Chiang, W.-C. and Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27.

Chiarandini, M. and Stützle, T. (2002). An application of iterated local search to graph coloring problem. In Johnson, D. S., Mehrotra, A., and Trick, M., editors, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 112–125, Ithaca, New York, USA.

Chiu, Y.-C. and Zheng, H. (2007). Real-time mobilization decisions for multi-priority emergency response resources and evacuation groups: model formulation and solution. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):710–736.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282.

Church, R., Sorensen, P., and Corrigan, W. (2001). Manpower deployment in emergency services. *Fire Technology*, 37(3):219–234.

Ciancio, C., Laganà, D., Musmanno, R., and Santoro, F. (2018). An integrated algorithm for shift scheduling problems for local public transport companies. *Omega*, 75:139–153.

Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.

Cordeau, J.-F. and Laporte, G. (2001). A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 39(3):292–298.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2004). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5):542–546.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409.

Cortés, C. E., Gendreau, M., Leng, D., and Weintraub, A. (2011). A simulation-based approach for fleet design in a technician dispatch problem with stochastic demand. *Journal of the Operational Research Society*, 62(8):1510–1523.

Côté, M.-C., Gendron, B., and Rousseau, L.-M. (2013). Grammar-based column generation for personalized multi-activity shift scheduling. *INFORMS Journal on Computing*, 25(3):461–474.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Curtin, K. M., Hayslett-McCall, K., and Qiu, F. (2010). Determining optimal police patrol areas with maximal covering and backup covering location models. *Networks and Spatial Economics*, 10(1):125–145.

Dahmen, S. and Rekik, M. (2015). Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic. *Journal of Scheduling*, 18(2):207–223.

D'Amico, S. J., Wang, S.-J., Batta, R., and Rump, C. M. (2002). A simulated annealing approach to police district design. *Computers & Operations Research*, 29(6):667–684.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

De Bruecker, P., van den Bergh, J., Beliën, J., and Demeulemeester, E. (2015). Workforce planning incorporating skills: state of the art. *European Journal of Operational Research*, 243(1):1–16.

Department for Transport (DfT) (2016). *Travel time measures for local A roads, England: April 2015 to March 2016 report.* Department for Transport, United Kingdom.

Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

Desrosiers, J., Soumis, F., and Desrochers, M. (1984). Routing with time windows by column generation. *Networks*, 14(4):545–565.

Dohn, A., Kolind, E., and Clausen, J. (2009). The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4):1145–1157.

Dongarra, J. J. (2014). Performance of various computers using standard linear equations software. Report CS-89-85, Electrical Engineering and Computer Science Department, University of Tennessee.

Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483.

Elshafei, M. and Alfares, H. K. (2008). A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs. *Journal of Scheduling*, 11(2):85–93.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., and Sier, D. (2004a). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1-4):21–144.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., and Sier, D. (2004b). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27.

Fagerholt, K. (2001). Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, 131(3):559–571.

Fisher, M. L., Jörnsten, K. O., and Madsen, O. B. G. (1997). Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45(3):488–492.

Fujiwara, O., Makjamroen, T., and Gupta, K. K. (1987). Ambulance deployment analysis: a case study of bangkok. *European Journal of Operational Research*, 31(1):9–18.

Galindo, G. and Batta, R. (2013). Review of recent developments in OR/MS research in disaster operations management. *European Journal of Operational Research*, 230(2):201–211.

Gauvin, C., Desaulniers, G., and Gendreau, M. (2014). A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 50:141–153.

Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390.

Gendreau, M., Jabali, O., and Rei, W. (2016). 50th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, 50(4):1163–1173.

Gendreau, M., Laporte, G., and Séguin, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29(2):143–155.

Gendreau, M., Laporte, G., and Séguin, R. (1996). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477.

Gendreau, M., Laporte, G., and Semet, F. (2006). The maximal expected coverage relocation problem for emergency vehicles. *Journal of the Operational Research Society*, 57(1):22–28.

Geroliminis, N., Kepaptsoglou, K., and Karlaftis, M. G. (2011). A hybrid hypercube–genetic algorithm approach for deploying many emergency response mobile units in an urban network. *European Journal of Operational Research*, 210(2):287–300.

Ghandehari, M. and Abdollahi, S. M. (2014). Solving a novel two-objective facility allocation-routing problem in large-scale emergencies using a two-stage simulated annealing algorithm. *International Journal of Decision Sciences, Risk and Management*, 5(3):277–292.

Ghiani, G., Manni, E., Quaranta, A., and Triki, C. (2009). Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):96–106.

Ghiani, G., Manni, E., and Thomas, B. W. (2012). A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, 46(3):374–387.

Glover, F. and Hao, J.-K. (2011). The case for strategic oscillation. *Annals of Operations Research*, 183(1):163–173.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston.

Golden, B. L., Raghavan, S., and Wasil, E. A. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, New York.

Goodson, J. C., Ohlmann, J. W., and Thomas, B. W. (2012). Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand. *European Journal of Operational Research*, 217(2):312–323.

Goodson, J. C., Ohlmann, J. W., and Thomas, B. W. (2013). Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research*, 61(1):138–154.

Green, L. V. and Kolesar, P. J. (2004). Anniversary article: Improving emergency responsiveness with management science. *Management Science*, 50(8):1001–1014.

Grujičić, I. and Stanimirović, Z. (2012). Variable neighborhood search method for optimizing the emergency service network of police special forces units. *Electronic Notes in Discrete Mathematics*, 39:185–192.

Haghani, A., Tian, Q., and Hu, H. (2004). Simulation model for real-time emergency vehicle dispatching and routing. *Transportation Research Record: Journal of the Transportation Research Board*, 1882:176–183.

Hashimoto, H., Yagiura, M., and Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456.

Henderson, S. G. and Mason, A. J. (2004). Ambulance service planning: simulation and data visualisation. In Brandeau, M. L., Sainfort, F., and Pierskalla, W. P., editors, *Operations Research and Health Care, A Hand Book of Methods and Application*, volume 70, pages 77–102. Springer, Boston, MA.

Hogg, J. M. (1968). The siting of fire stations. *Journal of the Operational Research Society*, 19(3):275–287.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.

Hvattum, L. M., Løkketangen, A., and Laporte, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438.

Hvattum, L. M., Løkketangen, A., and Laporte, G. (2007). A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, 49(4):330–340.

Iannoni, A. P., Morabito, R., and Saydam, C. (2011). Optimizing large-scale emergency medical system operations on highways using the hypercube queuing model. *Socio-Economic Planning Sciences*, 45(3):105–117.

Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., and Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics*, 156(11):2050–2069.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34(4):426–438.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2006). Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225.

Ingolfsson, A., Erkut, E., and Budge, S. (2003). Simulation of single start station for Edmonton EMS. *Journal of the Operational Research Society*, 54(7):736–746.

Jabali, O., Rei, W., Gendreau, M., and Laporte, G. (2014). Partial-route inequalities for the multi-vehicle routing problem with stochastic demands. *Discrete Applied Mathematics*, 177:121–136.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

Jotshi, A., Gong, Q., and Batta, R. (2009). Dispatching and routing of emergency vehicles in disaster mitigation using data fusion. *Socio-Economic Planning Sciences*, 43(1):1–24.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330.

Kanoun, I., Chabchoub, H., and Aouni, B. (2010). Goal programming model for fire and emergency service facilities site selection. *INFOR: Information Systems and Operational Research*, 48(3):143–153.

Kindervater, G. A. and Savelsbergh, M. W. (1997). Vehicle routing: handling edge exchanges. In Aarts, E. and Lenstra, J. K., editors, *Local Search in Combinatorial Optimization*, pages 337–360. Wiley, Chichester, UK.

Kohl, N., Desrosiers, J., Madsen, O. B., Solomon, M. M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116.

Kohl, N. and Madsen, O. B. (1997). An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, 45(3):395–406.

Kolen, A. W., Rinnooy Kan, A., and Trienekens, H. (1987). Vehicle routing with time windows. *Operations Research*, 35(2):266–273.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600.

Kyngas, J. and Nurmi, K. (2011). Days-off scheduling for a bus transportation company. *International Journal of Innovative Computing and Applications*, 3(1):42–49.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

Larsen, J. (1999). *Parallelization of the vehicle routing problem with time windows.* Thesis, Department of Informatics and Mathematical Modeling, Technical University of Denmark.

Law, A. M. (2007). *Simulation Modeling and Analysis.* McGraw-Hill, New York, 4th edition.

Lee, E. K., Maheshwary, S., Mason, J., and Glisson, W. (2006). Large-scale dispensing for emergency response to bioterrorism and infectious-disease outbreak. *Interfaces*, 36(6):591–607.

Lei, H., Laporte, G., and Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775–1783.

Lei, H., Laporte, G., and Guo, B. (2012). A generalized variable neighborhood search heuristic for the capacitated vehicle routing problem with stochastic service times. *Top*, 20(1):99–118.

Lewis, P. A. and Shedler, G. S. (1979). Simulation of nonhomogeneous poisson processes by thinning. *Naval Research Logistics*, 26(3):403–413.

Li, X., Tian, P., and Leung, S. C. (2010). Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, 125(1):137–145.

Li, X., Zhao, Z., Zhu, X., and Wyatt, T. (2011). Covering models and optimization techniques for emergency response facility location and planning: a review. *Mathematical Methods of Operations Research*, 74(3):281–310.

Liu, Z., Yu, H., Sui, J., Liu, D., and Wang, J. (2011). A research on vehicle scheduling problem to rescue the victims from chemical and biological terrorist attacks. In *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL)*, pages 356–361, Chongqing, China.

Lourenço, H., Martin, O., and Stützle, T. (2003). Iterated local search. In Glover, F. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, pages 320–353. Springer, Boston, MA.

Lourenço, H. R. (1995). Job-shop scheduling: Computational study of local search and large-step optimization methods. *European Journal of Operational Research*, 83(2):347–364.

Madansky, A. (1960). Inequalities for stochastic linear programming problems. *Management Science*, 6(2):197–204.

Maenhout, B. and Vanhoucke, M. (2010). Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of Scheduling*, 13(1):77–93.

Marić, M., Stanimirović, Z., and Božović, S. (2013). Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals of Operations Research*, 227(1):3–23.

Martí, R. and Reinelt, G. (2011). *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Springer, Berlin Heidelberg, 1st edition.

Maxwell, M. S., Restrepo, M., Henderson, S. G., and Topaloglu, H. (2010). Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281.

McCormack, R. and Coates, G. (2015). A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. *European Journal of Operational Research*, 247(1):294–309.

Mendoza, J. E., Rousseau, L.-M., and Villegas, J. G. (2016). A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints. *Journal of Heuristics*, 22(4):539–566.

Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., and Vitry, G. (2014). Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41:196–207.

Mišković, S., Stanimirović, Z., and Grujičić, I. (2015). An efficient variable neighborhood search for solving a robust dynamic facility location problem in emergency service network. *Electronic Notes in Discrete Mathematics*, 47:261–268.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Montemanni, R., Gambardella, L. M., Rizzoli, A. E., and Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343.

Montoya, C., Bellenguez-Morineau, O., Pinson, E., and Rivreau, D. (2014). Branch-and-price approach for the multi-skill project scheduling problem. *Optimization Letters*, 8(5):1721–1734.

Nagata, Y., Bräysy, O., and Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724–737.

Naoum-Sawaya, J. and Elhedhli, S. (2013). A stochastic optimization model for real-time ambulance redeployment. *Computers & Operations Research*, 40(8):1972–1978.

Novoa, C. and Storer, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509–515.

Pal, R. and Bose, I. (2009). An optimization based approach for deployment of roadway incident response vehicles with reliability constraints. *European Journal of Operational Research*, 198(2):452–463.

Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G., and Prastacos, G. P. (2008). A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, 14(5):425–455.

Pegden, C. D., Sadowski, R. P., and Shannon, R. E. (1995). *Introduction to Simulation using SIMAN*. McGraw-Hill, New York, 2nd edition.

Penna, P. H. V., Subramanian, A., and Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013a). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

Pillac, V., Guéret, C., and Medaglia, A. L. (2012). On the dynamic technician routing and scheduling problem. In *Proceedings of the 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2012)*, Mykonos, Greece.

Pillac, V., Guéret, C., and Medaglia, A. L. (2013b). A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7):1525–1535.

Polacek, M., Hartl, R. F., Doerner, K., and Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):613–627.

Powell, W. B. (1996). A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers. *Transportation Science*, 30(3):195–219.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154.

Quimper, C.-G. and Rousseau, L.-M. (2010). A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392.

Reeves, C. R. (1995). *Introduction. Modern heuristic techniques for combinatorial problems*. McGraw-Hill, London.

Rizzoli, A. E., Montemanni, R., Lucibello, E., and Gambardella, L. M. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(2):135–151.

Robinson, S. (2004). *Simulation: the Practice of Model Development and Use*. John Wiley, Chichester, UK.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Rubinstein, R. Y. and Melamed, B. (1998). *Modern Simulation and Modeling*. Wiley, New York.

Sarasola, B., Doerner, K. F., Schmid, V., and Alba, E. (2016). Variable neighborhood search for the stochastic and dynamic vehicle routing problem. *Annals of Operations Research*, 236(2):425.

Sathe, A. and Miller-Hooks, E. (2005). Optimizing location and relocation of response units in guarding critical facilities. *Transportation Research Record: Journal of the Transportation Research Board*, 1923:127–136.

Savas, E. (1969). Simulation and cost-effectiveness analysis of New York's emergency ambulance service. *Management Science*, 15(12):B–608–B–627.

Schmid, V. (2012). Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219(3):611–621.

Schwarze, S. and Voß, S. (2012). Improved load balancing and resource utilization for the skill vehicle routing problem. *Optimization Letters*, 7(8):1805–1823.

Secomandi, N. and Margot, F. (2009). Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57(1):214–230.

Simpson, N. and Hancock, P. (2009). Fifty years of operational research and emergency response. *Journal of the Operational Research Society*, pages S126–S139.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Taş, D., Dellaert, N., van Woensel, T., and de Kok, T. (2013a). Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, 40(1):214–224.

Taş, D., Dellaert, N., van Woensel, T., and de Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies*, 48:66–83.

Taş, D., Gendreau, M., Dellaert, N., van Woensel, T., and de Kok, A. G. (2013b). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3):789–799.

Talbi, E.-G. (2009). *Metaheuristics: from Design to Implementation.* John Wiley & Sons, Hoboken, New Jersey, USA.

Thomas, B. W. (2007). Waiting strategies for anticipating service requests from known customer locations. *Transportation Science*, 41(3):319–331.

Thomas, B. W. and White III, C. C. (2004). Anticipatory route selection. *Transportation Science*, 38(4):473–487.

Tirado, G. and Hvattum, L. M. (2016). Improved solutions to dynamic and stochastic maritime pick-up and delivery problems using local search. *Annals of Operations Research*, 253(2):825–843.

Tirado, G., Hvattum, L. M., Fagerholt, K., and Cordeau, J.-F. (2013). Heuristics for dynamic and stochastic routing in industrial shipping. *Computers & Operations Research*, 40(1):253–263.

Tocher, K. D. (1963). *The Art of Simulation.* English University Press, London.

Toro-Díaz, H., Mayorga, M. E., Chanta, S., and McLay, L. A. (2013). Joint location and dispatching decisions for emergency medical services. *Computers & Industrial Engineering*, 64(4):917–928.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications.* SIAM, Philadelphia, PA, USA, 2nd edition.

Valinsky, D. (1955). Symposium on applications of operations research to urban services-a determination of the optimum location of fire-fighting units in new york city. *Journal of the Operations Research Society of America*, 3(4):494–512.

Van Barneveld, T., Bhulai, S., and Van der Mei, R. D. (2016). The effect of ambulance relocations on the performance of ambulance service providers. *European Journal of Operational Research*, 252(1):257–269.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385.

Van Laarhoven, P. J. and Aarts, E. H. (1987). *Simulated annealing: theory and applications.* Springer, Netherlands.

van Veldhoven, S., Post, G., van der Veen, E., and Curtois, T. (2016). An assessment of a days off decomposition approach to personnel shift scheduling. *Annals of Operations Research*, 239(1):207–223.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013a). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013b). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics*, 21(3):329–358.

Viswanath, K. and Peeta, S. (2003). Multicommodity maximal covering network design problem for planning critical routes for earthquake response. *Transportation Research Record: Journal of the Transportation Research Board*, 1857:1–10.

Walker, J. D., Ochoa, G., Gendreau, M., and Burke, E. K. (2012). Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In Hamadi, Y. and Schoenauer, M., editors, *Learning and Intelligent Optimization*, pages 265–276. Springer, Berlin, Heidelberg.

Waters, C. (1989). Vehicle-scheduling problems with uncertainty and omitted customers. *Journal of the Operational Research Society*, 40(12):1099–1108.

Weintraub, A., Aboud, J., Fernandez, C., Laporte, G., and Ramirez, E. (1999). An emergency vehicle dispatching system for an electric utility in chile. *Journal of the Operational Research Society*, 50(7):690–696.

Winston, W. L. and Goldberg, J. B. (2004). *Operations Research: Applications and Algorithms*. Thomson Brooks/Cole, Belmont, CA, 4th edition.

Xu, J. and Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7(5):495–509.

Yan, L., Jinsong, B., Xiaofeng, H., and Ye, J. (2009). A heuristic project scheduling approach for quick response to maritime disaster rescue. *International Journal of Project Management*, 27(6):620–628.

Yan, S. and Shih, Y.-L. (2007). A time-space network model for work team scheduling after a major disaster. *Journal of the Chinese Institute of Engineers*, 30(1):63–75.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148.

Yang, L., Jones, B. F., and Yang, S.-H. (2007). A fuzzy multi-objective programming for optimization of fire station locations through genetic algorithms. *European Journal of Operational Research*, 181(2):903–915.

Yi, W. and Kumar, A. (2007). Ant colony optimization for disaster relief operations. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):660–672.

Yue, Y., Marla, L., and Krishnan, R. (2012). An efficient simulation-based approach to ambulance fleet allocation and dynamic redeployment. In *Proceedings of the 26th Conference on Artificial Intelligence*, Toronto, Ontario, Canada.

Zhan, S.-l. and Liu, N. (2011). A multi-objective stochastic programming model for emergency logistics based on goal programming. In *Proceedings of the 4th International Joint Conference on Computational Sciences and Optimization*, pages 640–644, Kunming and Lijiang City, China.

Zhang, J.-H., Li, J., and Liu, Z.-P. (2012). Multiple-resource and multiple-depot emergency response problem considering secondary disasters. *Expert Systems with Applications*, 39(12):11066–11071.

Zhang, Y. and Brown, D. (2014a). Simulation optimization of police patrol district design using an adjusted simulated annealing approach. In *Proceedings of the Spring Simulation Multi-Conference*, pages 125–132, Tampa, FL.

Zhang, Y. and Brown, D. (2014b). Simulation optimization of police patrol districting plans using response surfaces. *Simulation*, 90(6):687–705.

Zhang, Y. and Brown, D. E. (2013). Police patrol districting method and simulation evaluation using agent-based model & GIS. *Security Informatics*, 2(1):1–13.

Zhen, L., Wang, K., Hu, H., and Chang, D. (2014). A simulation optimization framework for ambulance deployment and relocation problems. *Computers & Industrial Engineering*, 72:12–23.

Zheng, Y., Chen, S., and Ling, H. (2013). Efficient multi-objective tabu search for emergency equipment maintenance scheduling in disaster rescue. *Optimization Letters*, 7(1):89–100.