METAHEURISTIC ALGORITHMS FOR THE VEHICLE ROUTING PROBLEM WITH TIME WINDOW AND SKILL SET CONSTRAINTS

by

Lu Han

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
December 2016

Table of Contents

List of Tables

List of Figures

Abstract

A subcontractor assigns installation services requested by a group of geographically scattered customers to its technicians daily. Requested services by the customers require a set of skills and should start during customers' availability periods. Each technician has certain skills, and limited availability daily to perform the services. Subcontractor's revenue from the installation jobs is fixed based on standard time for each type of job. The problem of assigning the optimum subset of the jobs to eligible technicians and identifying the optimum route to perform the services is classified as NP-Hard and is a challenging problem to solve.

In this research, a vehicle routing problem with time window and skill set constraints (VRPTWSS) is considered to address the problem. Due to the NP-hardness of vehicle routing problem, two metaheuristic algorithms are proposed to solve this problem. Performance of these approaches is evaluated against an MILP model for smaller problems via simulation.

List of Abbreviations Used

VRP	Vehicle Routing Problem
mVRP	m-Vehicle Routing Problem
TSP	Travelling Salesman Problem
TRP	Travelling Repairman Problem
CVRP	Capacitated Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
IRP	Inventory Routing Problem
SDVRP	Split Delivery Vehicle Routing Problem
GVRP	Green Vehicle Routing Problems
PVRP	The Period Vehicle Routing Problem
VRPSS	Vehicle Routing Problem with Skill Sets
TSRP	Technician Scheduling and Routing Problem
FSSP	Field Service Scheduling Problem
MILP	Mixed Integer Linear programming
TS	Tabu Search
SA	Simulated Annealing

Acknowledgements

I would like to take this opportunity to express my profound gratitude to my supervisor Dr. Alireza Ghasemi for his guidance, encouragement, and patience. In the last two years, he gave me great help by providing inspiration of new ideas and considerate suggestions on my research and life.

I would also like to thank Dr. Pemberton Cyrus and Dr. Jenny Chen for being my committee members, Dr. Claver Diallo and Dr. Uday Venkatadri for teaching me and providing me with instructive suggestions on my seminar, and all my teachers for their scholarly advice on my graduate courses.

My appreciation also extends to my friends Parth Pancholi, Fatemeh Mortazavi, Pin Hou for their selfless suggestions and encouragement on my research.

I would like to give my sincere appreciation to my parents, who give me strong support both emotionally and financially. Thanks for providing me the opportunity to study abroad and to pursue advanced degrees.

Chapter 1 Introduction

The research considers a subcontractor providing utility installation services that receives next day installation orders from its client. Each customer requiring an installation service has a predetermined available time window to start the job and each job requires a specific skill. This thesis will deal with the jobs supplied by the client who is responsible for defining the skill requirements of each job and scheduling installation appointments with customers. Besides, the client will estimate the time spent in each job and negotiate a predetermined cost with the subcontractor. Each technician hired by the subcontractor has a skill set with limited availability. Figure 1.1 illustrated the overall process.

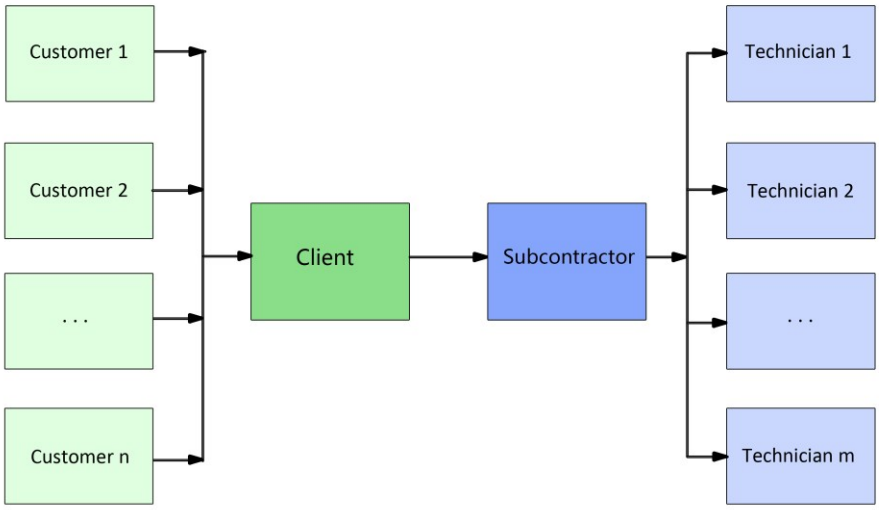


Figure 1.1 Overall process

Ultimately, the subcontractor's goal is to minimize their total cost since the revenue is fixed based on standard time for each type of job and complete information of each specific job. This problem can be regarded as a special case of Vehicle Routing Problem (VRP).

The VRP is one of the most studied combinatorial optimization and integer-programming problem and can be described as designing the optimal routes of a fleet of vehicles to serve a number of scattered customers from one or several depots (Golden et al. (2008)). In a VRP, each customer is visited exactly once by only one vehicle, and all vehicles must leave and return to the depot. The VRP attracts great attention and plays a significant role in the field of distribution and logistics since it is practical and hard to solve. The concept of VRP was first proposed by Dantzig and Ramser (1959) and a mathematical programming formulation and algorithm method for the VRP were also developed in this study. Clarke and Wright (1964) developed a greedy heuristic to reach an approximate solution of the VRP. Since then, hundreds of papers have been focused on the topic of looking for exact or approximate solutions for this problem and many of its variants. In a typical single VRP, the purpose is to find a tour with the minimum cost starting from the depot and connecting all the customers, then return to the depot. The m-Vehicle Routing Problem (mVRP) extends the single VRP to m tours.

VRP generalizes the well-known Travelling Salesman Problem (TSP) that is one of the simplest routing problems. Ropke (2005) describes TSP as the problem of finding the shortest route that visits all the nodes exactly once and returns back to the starting node given a set of nodes and a way of measuring distances between nodes. A similar problem relative to our research is the Travelling Repairman Problem (TRP). The objective of this problem is to find a route that minimizes the total waiting time of all the nodes.

Many variants of the VRP have been extensively studied. The Capacitated Vehicle Routing Problem (CVRP) is a variant in which vehicles have capacity limitations and customers have certain amount of goods to be picked up or unloaded. Thus, the vehicle capacity must be taken into consideration when designing the routes. A customer can only be served by a vehicle if the remaining capacity of the vehicle is greater than or equal to the capacity requirement of the customer. Another commonly studied variant of VRP is the Vehicle Routing Problem with Time Windows (VRPTW). For a VRPTW, all nodes have time windows within which the visits or deliveries must be made.

More complex variants of the VRP are featured by multiple depots, multiple vehicle types with different capacity or other constraints. For example, Heterogeneous Fleet VRP

or the Mixed Fleet VRP is characterized by a fleet of vehicles with different capacities and costs. Baldacci et al. (2008) present an overview of methods in solving Heterogeneous Fleet VRP. The aim is to find the optimal routes for each vehicle. The Inventory Routing Problem (IRP) or sometimes known as VRP with Inventory Constraints integrates and coordinates the two components of supply chain: inventory management and vehicle routing (Campbell et al. (1998)). Customers have an inventory capacity up to a predetermined maximum. A fleet of homogeneous vehicles with certain capacity is available for the distribution. The objective is to minimize total cost without leading to stockouts at any customers in the basic model. The Split Delivery Vehicle Routing Problem (SDVRP) was first introduced and defined by Dror and Trudeau (1990). SDVRP allows each customer to be visited more than once and the demand of each customer might exceed the capacity of the vehicle. However, the sum of quantities in each route cannot exceed the capacity of the vehicle. The development of Green Vehicle Routing Problems (GVRP) is motivated by the long-term sustainable requirement in distribution and logistics strategies. GVRP are identified by the objective of minimizing economic and environmental cost. Lin et al. (2014) present a survey on GVRP. Another extension of the VRP problem relevant to our research is the Vehicle Routing Problem with Skill Sets (VRPSS). It deals with a limited number of technicians that serves a bunch of customers with different requests. In one of its variants, Technician Scheduling and Routing Problem (TSRP), other constraints including tools, spare parts and requests with different urgency levels are considered (Pillac et al. (2013)).

Lenstra and Kan (1981) pointed out that most VRPs belong to NP-hard problems and are not likely to be solved in polynomial time. This NP-hardness characteristic accounts for the great attention on VRPs and the importance of solving VRPs by different algorithms. Both exact and approximate algorithms were developed in the last several decades. Exact methods are only effective to relatively small problems considering the NP-hardness of VRPs, while some approximate algorithms can provide excellent solutions and save a lot of time in the meanwhile according to Laporte (1992). Thus, heuristic methods are very popular in dealing with VRP-related problem.

The purpose of this research is therefore to develop models and algorithms for the subcontractor to arrive at a planning and scheduling solution for assigning customers to technicians, thereby minimizing total cost and improving the scheduling speed.

The remainder of this thesis is organized as below: In chapter 2, a literature review relevant to VRP and its variants is presented. In chapter 3, the exact problem definition is presented in further detail and several assumptions are made and explained. In chapter 4, a linear programming mathematical model, a heuristic method with local improvement and two metaheuristic algorithms (Tabu Search and Simulated Annealing) are presented to solve this problem. In chapter 5, the methods are examined using simulated data and the comparisons among different methods on cost and computation time are presented. Finally, conclusions and discussions are presented and further researches are identified.

Chapter 2 Literature Review

In this chapter, a review of VRP and its variants is demonstrated. VRP is an extensively researched area and is defined quite broadly. Desrochers et al. (1990) proposed a classification scheme for vehicle routing and scheduling problem and Desrochers et al. (1999) presented a modified methodology of classifying VRP in terms of real-life problem situation, abstract problem type, and algorithms applied. Eksioglu et al. (2009) then defined the domain of VRP and classified the literature in much greater detail based on type of study, scenario characteristics, problem physical characteristics and so forth. The type of study includes theory, applied methods, survey or review. In the second category, various scenario characteristics include whether the number of stops on route is deterministic or not, whether the load splitting is allowed, and so forth. Problem physical characteristics include the number of depot, number of vehicles, capacity considerations and so on. Since the concept of VRP was proposed, various methods have been developed to deal with it. Commonly used methods include exact algorithms, heuristic algorithms and metaheuristics and researches on these methods will be discussed in the following.

2.1 General VRP

2.1.1 Exact Algorithms

Exact algorithms have been extensively studied to solve VRPs in the last decades. Laporte and Nobert (1987) classified exact algorithms into three categories: direct tree search methods; dynamic programming; and integer linear programming. The latter category is very broad and can be subdivided into three categories according to Magnanti (1981): set partitioning formulations, vehicle flow formulations, and commodity flow formulations. Among them, vehicle flow formulations account for the most research efforts in early researches.

Direct tree search method works by sequentially constructing vehicle routes by means of a branch and bound tree. The branch and bound algorithm was first proposed to deal with TSP and then applied to VRP according to Christofides et al. (1981). They computed

lower bounds by shortest spanning k-degree center tree and q routes. Computational results showed that problems with up to 25 customers could be solved optimally. Laporte et al. (1986) applied this algorithm into VRP through transforming the distance matrix and solved asymmetrical CVRPs optimally including up to 260 nodes. Kumar and Jain (2015) solved a school bus routing problem with 65 buses to optimum.

Dynamic programming was first introduced in VRP by Eilon et al. (1974). This method requires a large number of computations. Efficient dynamic programming requires a relaxation procedure or feasibility or dominance criteria to reduce the number of states. Christofides et al. (1981) solved CVRPs with up to 50 nodes optimally using this approach. Desrosiers et al. (1984) considered a VRP in which customers request to be picked up at a given location and delivered to another one and solved the problem involving 80 nodes to optimum in less than 6 seconds.

Over the years, several integer linear programming models have been proposed for VRPs. Examples of mathematical models of VRP and its variants can be found in Laporte and Nobert (1987), Laporte (1992), Ropke (2005), Toth and Vigo (2014). Among integer linear programming algorithms, set partitioning formulations account for many research efforts. Balinski and Quandt (1964) were the first to propose set partitioning formulations. The set partitioning formulation consists of an m*n binary matrix in which each column represents a feasible route and each row represents a customer. All the rows should be covered at a minimal cost by subsetting columns. However, the formulations include an exponential number of variables in problems with many feasible solutions and usually cannot be used directly to solve VRPs. Many researchers have applied column generation schemes to deal with the difficulties generated by using the set partitioning approach. The idea of column generation is that many linear programs are too large to consider all variables explicitly and since most of the variables are assumed a value of zero in the optimal solution, only a subset of variables need to be considered. Thus, the column generation only generates variables with the potential to improve the objective function value. The problem is then split into two problems: the master problem is the original problem with only a subset of variables being considered and the subset problem adds variables to the master problem. For instance, Foster and Ryan (1976) proposed a

column generation method and obtain routes by dynamic programming. Agarwal et al. (1989) presented an exact algorithm based on set partitioning formulations. The results demonstrated that this method is roughly 13 times faster than that of Christofides et al. (1981). Vehicle flow formulations use binary variables to demonstrate whether the vehicle travels between two nodes in the optimal solution. In a two-index formulation, x_{ij} indicates whether edge (i, j) is traversed by a vehicle. A two-index formulation is widely used in symmetrical CVRPs and DVRPs. In a three-index formulation, binary variables x_{ijk} indicates that vehicle k travels directly from node i to node j. Naddef and Rinaldi (2001) used a branch-and-cut algorithm to solve a CVRP based on a two-index vehicle flow formulation. Fisher and Jaikumar (1981) developed a three-index vehicle flow formulation in a VRP with time windows and capacity constraints. In commodity flow formulations, flow variables that indicate the quantity of demand travelling on an arc are associated. Baldacci et al. (2004) developed a commodity flow formulation and then obtained a lower bound from the relaxation of this formulation. Baldacci et al. (2012) present a review of mathematical formulations and recent algorithms for CVRP and VRPTW.

2.1.2 Heuristic and Metaheuristic Algorithms

In recent years, although several sophisticated exact algorithms have been proposed for solving the VRPs, only relatively small problems with around 100 customers can be solved to optimum and the variance of computation time is high (Laporte et al. (2014)). However, practical problems are often large and require more efficient methods to solve the problem within reasonable computation time. Thus, efficient heuristics attract the attention of researchers. Overview of classical heuristics and metaheuristics of VRPs can be found in Laporte et al. (2000), Laporte (2007), Cordeau et al. (2005), and Toth and Vigo (2014).

Classical heuristics for VRP can be categorized typically into construction heuristics and improvement heuristics. Construction heuristics created a feasible solution and pay attention to the objective value in the meanwhile. However, they do not include any improvement attempts. Construction heuristics in VRPs usually falls into three

categories: insertion heuristics, saving heuristics, and clustering heuristics (Ropke (2005)).

Insertion heuristics create routes by inserting customers into routes. Routes can be built one at a time (sequential insertion heuristics) or several at the same time (parallel insertion heuristics). The insertion heuristics apply different criterion to determine which customer to insert and where to insert the customer. Recent applications of insertion heuristics include the VRPTW insertion heuristic proposed by Ioannou et al. (2001) and a pickup and delivery problem by Lu and Dessouky (2006).

The classical savings heuristic was first proposed in 1964 by Clarke and Wright (1964) to solve CVRPs. It initialized with single node routes and then two routes are merged at each step according to the largest saving that can be generated. Several enhancements of this heuristic can be found in the work of Nelson et al. (1985) and Paessens (1988). Nelson et al. (1985) proposed six methods for implementing the savings heuristic and tested 55 problems to compare these methods. The results clearly provided suggestions on the choice of methods for different VRPs with given characteristics. Paessens (1988) modified the savings heuristic which showed less computation time and reduced storage requirements. Most savings algorithms have been developed for CVRPs. For example, Altinel and Öncan (2005) proposed a new enhancement of savings heuristic to solve a CVRP. This research differs from previous ones in considering customer demands in addition to distances as the saving criterion. The results proved this enhancement to be fast and accurate. Laporte and Semet (2001) proposed several variants of the savings algorithm to speed up computations. The examples of applying savings algorithms on other variants of VRP include a VRPTW by Liu and Shen (1999) and a pickup and delivery problem by Gronalt et al. (2003). Liu and Shen (1999) described several insertion-based savings heuristics for solving the VRPTW with a heterogeneous fleet and took into account a sequential route construction parameter to avoid too many short routes. Experimental results demonstrated that heuristics with the consideration of this parameter yields better solution than all other heuristics tested. Gronalt et al. (2003) proposed four savings heuristics for the pickup and delivery problem under time window constraints. Computational results showed that these heuristics find very good solutions

quickly and the consideration of the opportunity costs significantly improved the solution quality.

Clustering algorithms are two-phase algorithms. The first phase works on grouping customers into subsets and each subset of customers is served by one vehicle. The second phase constructs routes for each subset. Fisher and Jaikumar (1981) proposed a clustering heuristic for CVRP. This clustering algorithm is proved to outperform the existing heuristics on some standard test examples and can always find a feasible solution if it exists. Another classical clustering algorithm is the sweep algorithm that was first proposed in the work of Wren (1971). This algorithm selects a vehicle and assigns unrouted nodes with the smallest angle to the vehicle until the capacity of the vehicle is exceeded. Once all the nodes have been assigned, each vehicle route is optimized internally by solving the corresponding TSP and externally by exchanges between adjacent routes. Gillett and Miller (1974) started to call this method the sweep algorithm and popularized it. Computational results in their research figured out that sweep algorithm generally produced better results than the savings approach while less efficient in computational time.

Classical improvement heuristics work on intra-route or inter-route moves. Intra-route moves improve the route internally by exchanging the order of nodes on a single route. Intra-route moves were first proposed for TSP and then applied to all VRPs. Lin (1965) proposed a k-opt exchange in which k consecutive nodes are removed and the first node in one sequence is connected to the last node of the second sequence. Later on, Lin and Kernighan (1973) modified the parameter k dynamically during the improvement procedure. Or (1976) proposed the Or-opt heuristic in which a sequence of one, two or three consecutive customers from one route is removed and inserted into another location in the same route or in another route (see Figure 2.1). Renaud et al. (1996) developed a restricted version of 4-opt algorithm in which four links in a single route are removed and then four other links are added to rebuild a feasible route. Results presented in their research indicated that this algorithm outperformed Or-opt but not as good as 3-opt in terms of solution quality, but it is faster than 3-opt. Johnson and McGeoch (1997) analyzed and compared various improvement procedures for TSP. The computation

results showed that the dynamic k-opt heuristic proposed by Lin and Kernighan (1973) lead to the best results.

In practice, inter-route heuristics are significant in reaching good results according to Toth and Vigo (2014). Inter-route improvement heuristics move nodes from their current routes to other routes. Van Breedam (1994) classified improvement heuristics as string cross, string exchange, string relocate and string mix. Relocate heuristic removes one node from its current route and then insert it into another route (see Figure 2.2). A node can also be relocated from its current route to an empty route, leading to the generation of a new route; Exchange heuristic swaps two nodes from their original routes to the other route separately (see Figure 2.3); In cross heuristic, two links in two routes are broken separately and the first sub-route of the first original route is linked to the second sub-route of the second original route and vice versa. Thus, each new route is the combination of two sub-routes from both two routes. The string mix is the combination of the string exchange and the string relocate. These heuristics are widely used in different variants of VRP. Thompson and Psaraftis (1993) proposed cyclic transfers to multivehicle routing problems in which b routes are considered and k nodes of each route are relocated to the next route.

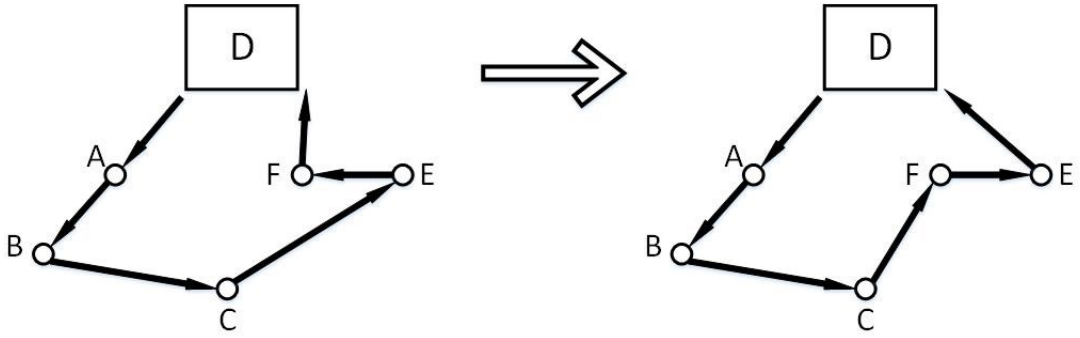


Figure 2.1 Or-opt heuristic

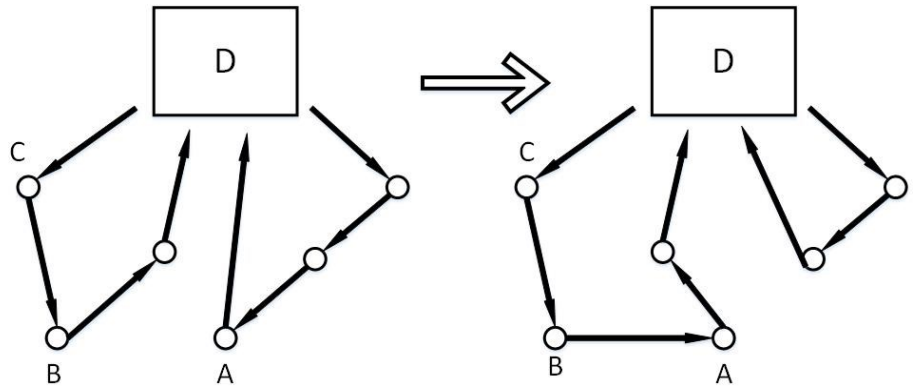


Figure 2.2 Relocate (Transfer) heuristic

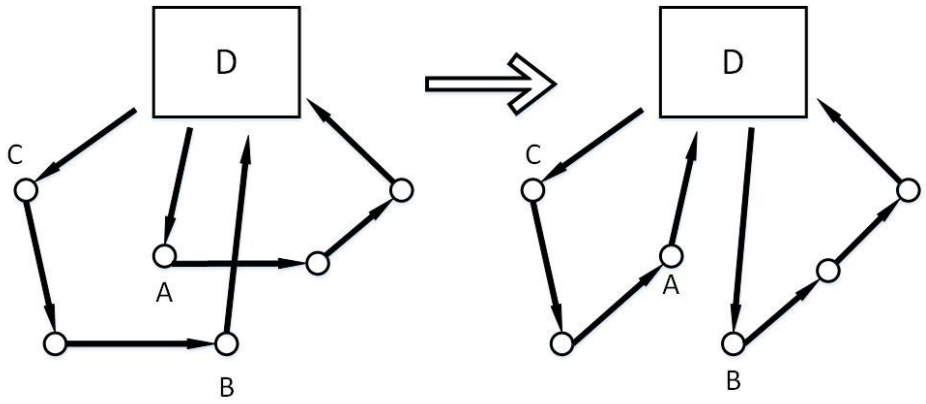


Figure 2.3 Exchange (Swap) heuristic

Common metaheuristic algorithms in VRPs include tabu search, simulated annealing, genetic algorithms, and ant colony optimization. The basic idea of tabu search is to accept the best neighbor of current solution as the new trial solution even if it is a worse solution. Different memory structures are applied to avoid cycling back to the same solution. Tabu search is the most extensively studied metaheuristic algorithm in VRPs. Recent examples can be found in Wassan (2006), Derigs and Kaiser (2007), and Lai et al. (2016). In simulated annealing algorithm, a solution is randomly selected from the neighborhood of the current solution and cycling is prevented. Examples include Zeng et al. (2005) and Osman (1993). Genetic algorithms mimic the nature selection and combine

selection, recommendation and mutation processes. Examples can be found in Baker and Ayechew (2003), Berger and Barkaoui (2003), and Prins (2004). In ant colony optimization, artificial ants construct solutions in a greedy and random way in each cycle and then chooses new element to be included into the current solution based on the evaluation of the element. Bell et al. (2004) modified ant colony optimization algorithm used to solve TSP in order to apply it on multiple routes of VRP. Reimann et al. (2004) proposed a savings based ant colony algorithm in which solutions are constructed based on the savings algorithm and Doerner et al. (2005) proposed a parallelization version of the same algorithm in which the problem is decomposed into a number of sub-problems and the sub-problems are processed in parallel.

2.2 Variants of Vehicle Routing Problem

In this section, a review of several important variants of VRP is presented.

2.2.1 The Travelling Salesman Problem (TSP)

TSP is one of the most studied NP-hard problem and current solution methods have reached a very high level. A TSP aims at finding the shortest tour that visits each node exactly once and return to the starting node. TSPs have been extensively studied in the literature and many algorithms proposed for TSPs have been modified to solve other variants of VRP.

Mathematical formulations of TSP can be found in Laporte (1992), Hoffman et al. (2013) and many other researches. Letchford et al. (2013) provided a review of different formulations of the TSP.

Common exact algorithms derived from the mathematical formulations of TSP include branch-and-bound algorithms and shortest spanning tree algorithms. Laporte (1992), Lawler et al. (1985), and Gutin and Punnen (2006) provided a review of different exact algorithms.

Local search heuristics are among the main tools to search for near optimal solutions in TSP. As mentioned above, improvement heuristics for TSP can be found in Lin (1965), Lin and Kernighan (1973), Or (1976), Renaud et al. (1996) and Johnson and McGeoch

(1997). These algorithms have been applied to all VRPs as well to improve the route internally.

Metaheuristic approaches are widely applied in solving large-scale TSPs. Potvin (1996), Bryant and Benjamin (2000), and Yuan et al. (2013) applied genetic algorithms. Asrts et al. (1988), Wang et al. (2013) applied simulated annealing. Flechter (1994) developed a parallel tabu search algorithm to solve large-scale TSPs. Xu et al. (2015) compared different tabu search algorithms and proposed hybrid algorithms by combining tabu search with other methods.

2.2.2 The Capacitated Vehicle Routing Problem (CVRP)

Other than the TSP, many VRPs deal with problems including a fleet of vehicles instead of working on a single vehicle. The CVRP is the most studied variant of VRP since it was first introduced by Dantzig and Ramser (1959). Vehicles in the CVRPs have limited capacity of goods that must be delivered. Among mathematical models developed for the CVRP, Achuthan and Caccetta (1991) proposed a mixed integer linear programming formulation constrained by distance travelled. Kara et al. (2004) provided a formulation for the CVRP and extended the subtour elimination constraints that prevent subtours from arising. Mathematical models in recent research can also be found in Baldacci et al. (2012) and Semet et al. (2014) that provided a review of different formulations.

Most extensively applied exact algorithms in the CVRP are based on branch-and-cut algorithms and set partitioning formulation. Augerat et al. (1995) were the first to apply an exact branch-and-cut algorithm in the CVRP and this algorithm was able to solve a CVRP with 135 customers according to the computational results. Lysgaard et al. (2004) presented a new branch-and-cut algorithm in which several classes of valid inequalities were used and the results showed the algorithm to be competitive with others. The first set partitioning formulation of CVRP was developed by Balinski and Quandt (1964). Fukasawa et al. (2006) proposed a set partitioning formulation which could solve up to 100 nodes optimally. Baldacci et al. (2008) improved a set partitioning model by using valid inequalities. Computational results showed that the lower bounds generated by this model are better than the best lower bounds generated by Fukasawa et al. (2006).

Many common heuristic algorithms designed for VRP have been used to solve the CVRP as well. For instance, Toth and Vigo (2001) applied a 3-opt algorithm to search for neighbors in an improved savings algorithm. Some researches used a two-step method (Fisher and Jaikumar (1981), Gillett and Johnson (1976)) where in the first step the customers are clustered into several groups and the second step constructs each route.

Metaheuristic approaches attract much attention in solving CVRP especially in recent years. Extensively used metaheuristics in CVRP include simulated annealing algorithms (Robust et al. (1990), Osman (1993), Tavakkoli et al. (2006), Xiao et al. (2014)), tabu search algorithms (Osman (1993), Gendreau et al. (1994), Jin et al. (2012)), and genetic algorithms (Baker and Ayechew (2003), Prins (2004), Nazif and Lee (2012)). Lin et al. (2009) applied a hybrid algorithm of simulated annealing and tabu search to solve CVRP.

2.2.3 The Inventory Routing Problem (IRP)

The IRP differs from other variants of VRPs due to the consideration of inventory of customers. The supplier must ensure that customers do not experience a stock-out by managing inventory of each customer. Meanwhile, the inventory cost is incurred in the process and is considered in the objective function in many cases. The subject is usually to trade off between transportation cost and inventory cost in studies taking into consideration the inventory cost.

IRPs have been studied since the eighties. Among papers considering the inventory cost, Speranza and Ukovich (1994) was the first to propose a model for a single retailer case with fixed shipping frequencies. They also built up a mixed-integer programming model to solve the problem and showed that the model is NP-hard (Speranza and Ukovich (1996)). Bertazzi et al. (2000) applied both an exact algorithm based on a branch-and-bound algorithm and some heuristics to solve the problem. Later on, Bertazzi et al. (2007) analyzed different dispatch policies to decide when to make shipments and the how to load vehicles. In multiple retailers' cases, Archetti et al. (2007) proposed a mixed integer linear programming model and used a branch-and-cut algorithm to solve it optimally. Bertazzi et al. (2002) applied a two-step heuristic algorithm in which a feasible solution is constructed and then improved to solve the IRP. Then Bertazzi et al. (2005) proposed both exact and heuristic algorithms to analyze two vendor-managed inventory policies in

which the facility takes charge of the replenishment policies of the retailer. Cousineau-Ouimet (2002) used a tabu search algorithm to design the route and determine the delivery size and frequency for IRP.

There are also a group of papers without considering inventory costs. For example, Berman and Larson (2001) proposed a dynamic programming method to dynamically determine the amount of product provided to each customer. Campbell and Savelsbergh (2004) proposed a two-phase method. The first phase constructs a delivery schedule and then the routes are built up in the second phase. Savelsbergh and Song (2007) compared the performance of different greedy heuristics.

2.2.4 The Period Vehicle Routing Problem (PVRP)

In the PVRP, routes are constructed over a planning period with more than one day. In each single day, a fleet of vehicles should travel from and end at a depot.

Solution methods have focused on the classical two-phase construction-improvement methods for PVRP. For example, Tan and Beasley (1984) determined the delivery day for each customer in the first phase. Then they assign the customer to a vehicle in the chosen delivery day. Russell and Gribbin (1981) also presented a multi-phase method. The first phase achieved a feasible initial solution, followed by two improvement phases applying interchange heuristics. This multiphase approach generates improvements over previous research according to the computational results.

Among metaheuristic algorithms for PVRP, Chao et al. (1995) created an initial solution and then improved it iteratively by moving a customer from one schedule to another. The movement is always accepted if it leads to a decrease in total distance. Otherwise, the move is accepted if the total distance is less than a threshold. With the increase of iterations, the threshold gradually decreases. This metaheuristic is similar to the simulated annealing except that they use different acceptance criteria. Cordeau et al. (1997) proposed a tabu search method to solve PVRP, the period TSP, and multi-depot VRP. Drummond et al. (2001) presented a hybrid metaheuristic algorithm based on the combination of parallel genetic algorithms and local search heuristics.

2.2.5 The Split Delivery Vehicle Routing Problem (SDVRP)

In the SDVRP, customers can be visited by more than one vehicle. This variant of VRP attracts the attention of researchers because it leads to up to 50% cost savings by splitting deliveries according to Archetti et al. (2006).

Exact approaches for the SDVRP include Lee et al. (2006) and Jin et al. (2007). Lee et al. (2006) applied a shortest path search algorithm to solve the SDVRP. Jin et al. (2007) proposed a two-stage algorithm. The first stage creates clusters and establishes lower bound, while the second stage constructs routes for each cluster. Both methods are able to solve only small instances. Better solutions can be reached in Feillet et al. (2006) in terms of cost and number of vehicles. They considered the SDVRP with time windows and were able to solve instances with 100 customers.

The first heuristic algorithm for the SDVRP is introduced by Dror et al. (1990) through a local search process. Frizzell and Giffin (1995) proposed a construction heuristic and two improvement heuristics (swap and transfer) to solve the SDVRP with time windows. Archetti et al. (2006) proposed a tabu search metaheuristic and applied the relocate heuristic (See Figure 2.2) as the neighborhood structure. The results were compared with that of Dror et al. (1990) and showed the algorithm can almost always provide better solutions. Later on, Archetti et al. (2008) combined tabu search with optimization method to solve the SDVRP.

2.2.6 Vehicle Routing Problem with Time Windows (VRPTW)

In VRPTW, each customer has a specified time interval and the service for each customer has to begin within this time interval. VRPTWs have been thoroughly studied during the last few decades due to its extensive applications in practice.

Mathematical model of the VRPTW can be found in a great number of papers such as Kallehauge et al. (2005) and Kohl and Madsen (1997). Among exact algorithms, Kohl and Madsen (1997) developed a method exploiting Lagrangian relaxation of the assignment constraint that every customer is served exactly once. The algorithm is able to solve problems with up to 100 customers. Desrosiers et al. (1984) applied a column generation method for solving the VRPTW, and then Desrochers et al. (1992) presented a

more effective version of the same model by using a set partitioning formulation. The results proved the capability of this algorithm to solve a 100-customer problem. As the case of other VRPs, the exact methods often perform poorly in computation time and may take days or more to find good solutions. Thus, heuristic methods are more attractive in VRPTWs.

Solomon's research (1987) has always been regarded as Benchmark for construction type approaches and he illustrated several route construction heuristics including an extension to the savings heuristics, a time-oriented, nearest-neighbor heuristic, two insertion heuristics and a time-oriented sweep heuristic. After a thorough analysis of their performance of all above methods based on minimum number of vehicles, schedule time, distance, and waiting time, the author summarized that the first insertion heuristics has the best performance in this problem. Different form Solomon's sequential algorithm (1987) that may generate poor quality of the last routes, Potvin and Rousseau (1993) adopted the parallel algorithm for initializing routes. Sequential algorithms build one route at a time; parallel algorithms construct a set of routes simultaneously. The computational results showed that this approach is better than Solomon's sequential approach in most cases. Russell (1995) also developed a parallel algorithm based on Solomon's insertion heuristic. Ioannou et al. (2001) used an insertion heuristic with a new insertion criteria based on the minimization function of the greedy look-ahead framework. The algorithm was proved to be of good quality for large-scale problems in short computational time.

Various examples of route improvement heuristics can be found in VRPTW. For instance, Potvin and Rousseau (1995) introduced a new 2-opt* exchange heuristic and applied a hybrid heuristic based on 2-opt* and Or-opt exchange heuristics. In 2-opt* exchange heuristic, two routes are selected and one link connecting two nodes from each route is removed. Then the first node of the link in the first route is attached to the second node of the second route and the second node of the link in the first route is linked to the first node of the second route (see Figure 2.4). The 2-opt* heuristic is especially capable in problems with time windows since it reserves the direction of the previous routes. Prosser and Shaw (1996) compared one intra-tour heuristic (the 2-opt heuristic) with three inter-

tour heuristics (relocate, exchange, and cross heuristics). Comparisons of these neighborhood heuristics imply that the relocate heuristic is most effective in leading to a better solution, while exchange provides the least benefit. Besides, the best combination of two neighbourhood heuristic is relocate with cross. Kontoravdis and Bard (1995) first set the number of routes to a fixed lower bound in the route construction phase. Then the authors adopted Solomon's insertion algorithm (1987) in selecting customers. The computational tests indicated an improvement over previous procedures. Bräysy (2002) proposed a three-phase approach for solving VRPTWs. The first phase is to create an initial solution. The second phase combines routes with only few customers into one route. In the last phase, the Or-opt heuristic was applied again to reduce the total distance.

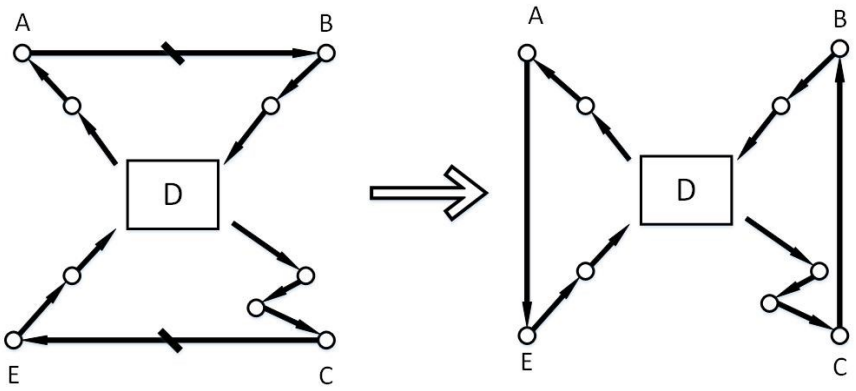


Figure 2.4 2-opt* heuristic

Metaheuristic methods received the most attention in the field of VRPTW. Chiang and Russell (1996) and Afifi et al. (2013) both applied simulated annealing metaheuristic with different neighborhood structures. Chiang and Russell (1996) applied the k-node interchange process and the λ-interchange mechanism as proposed by Osman (1993). They also improved the simulated annealing process by a tabu list. The computational tests generated better results in four of six data sets compared to previous research. In Afifi et al. (2013), simulated annealing algorithm is applied in solving a VRPTW with synchronization constraints, in which customers are allowed to be visited by more than vehicles but the visits need to be synchronized. This paper used a 2-opt* and Or-opt

neighborhood heuristics in the local search procedure. The experiments showed this algorithm to be fast and outperform existing approaches in solving this problem.

Tabu search algorithm is widely used in VRPTWs. Potvin et al. (1996) applied a neighborhood structure based on the combination of 2-opt* and Or-opt methods. Taillard et al. (1997) applied the cross change neighborhood heuristic. It used an adaptive memory to store and select solutions. This methodology generated many good solutions on Solomon's test sets. Chiang and Russell (1997) developed four tabu search metaheuristics including simple tabu search and tabu search including intensification, diversification, and reactive strategies and compare their results. The above three research applied sequential construction heuristic. Badeau et al. (1997) applied a parallel tabu search algorithm and the cross neighborhood heuristic, while Schulze and Fahle (1999) used a shift-sequence neighborhood heuristic based on simple customer shifts. Computational results showed an evident improvement on computational time while maintaining a high quality of solution. Thangiah et al. (1991) described a genetic algorithm method called GIDEON. The first phase of this method used genetic algorithm to divide customers into sectors or clusters and the second phase applied λ-interchange local optimization to relocate infeasible customers to other routes. Thangiah et al. (1994) applied a hybrid metaheuristic method involving all three metaheuristics and the results demonstrated the successful combination of tabu search, simulated annealing and genetic algorithm.

2.2.7 Vehicle Routing Problem with Skill Sets (VRPSS)

In VRPSS, each technician has a set of skills and each requested service by the customers requires a skill. The technician can serve a customer only if he or she masters the skill the customer requires. Another problem relative to VRPSS is referred to as Technician Scheduling and Routing Problem (TSRP). TSRP includes routing staff to serve requests while taking into account time windows, skills, tools and spare parts.

The researches on VRPSS are very limited. Cappanera et al. (2011) was the first to formulate the mathematical model for skill vehicle routing problem. They first defined the problem of deciding a set of tours and each of them is operated by a skilled technician in order to fulfill the service required by customers within a given tour. Then they

developed three mathematical models with increasing level of disaggregation and some related valid inequalities that help enhance some less disaggregated models. Tests on randomly generated examples proved that increasing disaggregation levels strengthen the associated MILP bounds. However, the skill VRP is still hard to arrive at an optimal solution within reasonable computational time according to their tests. Krishnamurti and Iranmanesh (2012) discussed the vehicle routing problem with skill sets using both integer programming model and local search. The authors tested the integer programming model in small instances using Cplex. Then they applied the combination of two neighborhood heuristics (intra-route and 2-opt) as the neighborhood structure in local search. The intra-route heuristic searches neighborhood internally by removing customers and insert them in the same route, while 2-opt heuristic removes customers from two different routes and exchanges them if the technicians can serve the exchanged customers. After the comparison of optimal solution and local search algorithm, the authors found that with the increase of jobs and vehicles, the differences of computation time and solutions between local search and integer programming model become increasingly more significant.

Among TSRP, Xu and Chiu (2001) aimed at maximizing the number of requests served while taking into consideration the skill constraints and priorities. Tang et al. (2007) also accounted for different urgency levels and used a multi-period maximum collection problem formulation to solve the problem. Pillac et al. (2013) proposed a parallel metaheuristic to solve the TSRP without considering the dynamic setting such as unexpected delays and new requests. The results showed a tiny gap of 0.23% to the optimal solutions on Solomon's benchmark (1987). Another research conducted by Pillac et al. (2012) was the first to consider all the important components of TSRP. These components include skills, tools, spare parts, and dynamically arriving requests. The authors proposed a period optimization approach as well as a continuous optimization approach. The computational results showed that the first method yields better results within limited computation time compared to the other.

Another relative problem is the Field Service Scheduling Problem (FSSP) that focuses on situations when the mobile workforce has to accomplish scattered tasks. Each task

requires a specific skill and each workforce possesses a combination of several skills. Tasks also have deadlines and different priorities. Lesaint et al. (2003) worked on a FSSP by developing a heuristic algorithm based on simulated annealing.Petrakis et al. (2012) proposed both static and dynamic algorithms to solve this particular problem. Static algorithms generate routes once a day in the morning, while dynamic algorithms process new tasks throughout the day and update the routing schedules dynamically.

2.2.8 Travelling Repairman Problem (TRP)

Another problem relative to our research is the TRP. This problem is also known as the minimum latency problem or the deliveryman problem. In a TRP, each node represents a machine to be repaired, and there is only one repairman. The total waiting time is the sum of the waiting time of all the nodes. The objective of this problem is to find a route that minimizes the total waiting time. The repair times are assumed to be equivalent.

Among exact approaches for TRP, Yang (1989) applied a dynamic programming algorithm. Simichi-Levi and Berman (1991) used a branch and bound algorithm to search for the optimal tour sequence. Fischetti et al. (1993) built up a linear programming model and obtained lower bounds for TRP. The proposed algorithm can optimally solve problems with up to 60 nodes.

The first approximation algorithm is proposed by Blum et al. (1994). They proposed an approximation algorithm with an approximation factor of 144. The best approximation algorithm for general metric spaces known now was proposed by Chaudhuri et al. (2003) with an approximation factor of 3.59. Considering the edge-weighted tree, the smallest approximation factor of 3.03 was found by Archer and Blasiak (2010).

Researches on TRP using metaheuristics are limited in literature. Salehipour et al. (2011) presented the first metaheuristic approach for TRP. These metaheuristics consist of a greedy randomized approach used in the construction phase and a Variable Neighborhood Descent or Variable Neighborhood Search used in the improvement phase. Silva et al. (2012) proposed a metaheuristic approach based on a greedy randomized approach in the construction phase and Variable Neighborhood Descent with random

neighborhood ordering in the improvement phase. This approach was tested on many instances with up to 1000 customers and can optimally solve instances with up to 107 customers. Dewilde et al. (2013) developed a tabu search algorithm to solve the TRP with profits. They applied the construction-improvement two-phase algorithm and used multiple neighborhood heuristics. Computational results demonstrated the high quality and efficiency of tabu search algorithm in finding very good solutions.

From the literature, common approaches of solving VRPs can be classified into exact algorithms, heuristic algorithms, and metaheuristics. Thus, we would develop these three types of methods to solve our problem as well. In heuristic algorithms, we select the neighborhood structure based on the constraints of our problem and the computational results demonstrated in the literature. In metaheuristic algorithms, we develop tabu search and simulated annealing to solve the problem since these two approaches received the most attention in metaheuristics according to the literature.

Chapter 3 Problem Context and Assumptions

The problem considered in this thesis aims at finding the optimal set of routes for a group of technicians with limited availability in order to satisfy the installation service requirements of customers with different appointment time windows. This problem can be regarded as The Vehicle Routing Problem with Time Window and Skill Set Constraints (VRPTWSS), which is a combination of the VRPTW and the VRPSS. VRPTWSS has not received much attention in the literature. However, it is a problem originated from real application and is worthwhile for further studies.

The process of allocating available technicians to suitable customers is time-consuming and difficult especially considering the large number of customers in practice and the complicated constraints in this particular problem. In this chapter, the definition and assumptions of this problem are provided in detail and the scope of this thesis is clarified.

3.1 Problem Definition

There are a large number of installation jobs received by the subcontractor in each working day. Every job has a specific skill requirement from the technician to complete the job. When booking the installation appointments, customers can choose from a variety of time windows and jobs must be started within the appointment time windows.

Each technician has a combination of skills with limited availability. A technician must satisfy the skill requirement of the installation job and be able to start the job during the installation time window of a customer. Due to the variety of experience, knowledge and aptitude, technicians have distinct working efficiencies. Technicians are responsible for both driving and serving customers. Besides, technicians arriving before the earliest available time of a customer have to wait and start the service at the customer's earliest available time.

Regional difficulties are taken into account in this problem too. The client company served by the subcontractor has identified that the time to complete the same jobs at different regions varies in light of old or nonexistent infrastructure in some neighborhoods. Besides, the client is responsible for estimating the basic service time to

complete each installation job and negotiating a predetermined service price with the subcontractor based on the basic service time. Since the number of jobs offered by the client is predetermined, the total revenue of the subcontractor in a given day is fixed and the only way to increase the profit is to reduce the total cost. The objective of this research is therefore to determine the combination of installation jobs and optimal routes for a group of technicians under time window and skill set constraints in order to reduce the total cost and computation time.

3.2 Problem Assumptions and Requirements

This section lists the detailed assumptions made and requirements of this problem:

- There is one depot in the entire system. Technicians must start from and return to the depot for daily tasks.
- Customers can select a time window consisting of the earliest and latest available time from several choices. In this research, we assume that time windows of customers do not partially overlap.
- Technicians can select a time window consisting of their earliest and latest available time from several choices as well. We assume that time windows of customers are no longer than that of technicians and the time windows of technicians do not partially overlap with the time windows of any customers.
- The service start time should be within the time window of each customer and the available time period of the technician. However, technicians are not required to finish the task within the available time periods of both the customer and the technician.
- Technicians are required to return to the depot before 18 p.m.
- Not all available technicians have to be assigned to the installation jobs.
- Technicians have different working efficiencies, while working efficiencies of a technician are assumed equal in performing different types of jobs. Hourly wage of different technicians are not much variant and are assumed equal in this study. The hourly wage of a technician is assumed equal during driving, waiting, and performing tasks as well.

- The actual service time spent during performing each job is based on three elements: basic service time, technician efficiency, and regional difficulty. Basic service time is the estimated time to complete a specific job predetermined by the client based on historic information of specific installation service or the industry standards. Regional difficulty reflects the condition of infrastructure required to complete the job at a customer's location. The rating of regional difficulty is also provided by the client based on existent records and evaluations. The rating of technician efficiencies is predetermined by the subcontractor through the analysis of history data.
- The total cost consists of technicians' salary and vehicle operation cost. Technician's salary is calculated based on the total time spent during driving, waiting, and performing jobs. Waiting cost is caused by early arrival at a customer's location. Vehicle operation cost is assumed to be fixed per unit of driving time.
- Factors that cannot be realized in advance or controlled by the model are not considered in this problem. Those factors may include: weather or traffic caused delays, customers not at home or sudden cancellations, stochastic variations of actual service time, lack or disruption of necessary infrastructure that cannot be solved within one working day and so forth.

Chapter 4 Methodology

In this chapter, four models and algorithms are presented to allocate the customers to eligible technicians and to determine the sequence of each technician route considering time window and skill set constraints. A mixed integer linear programming (MILP) model with binary variables is developed to find the optimal solution of this specific problem. Preprocessing is adopted to determine the eligibility of technicians to serve each customer in advance to simplify the MILP model and speed up computation. Since VRPs belong to NP-hard problems (Clarke and Wright (1964)), it is hard to arrive at optimal solutions for large-scale problems using an MILP model. Thus, a heuristic method is proposed by integrating two inter-route neighborhood heuristics and two intra-route heuristics. Finally, two metaheuristic algorithms (tabu search and simulated annealing) are applied to search for better solutions. The following content will describe these four methods in detail.

4.1 MILP Model

MILP (Hillier (2012)) is a method used to obtain an optimal result in a mathematical model in which all the mathematical functions are linear. The mathematical model consists of a linear function to be optimized (minimize or maximize), linear constraints, and usually non-negative variables. Parameters and variables used in the introduced MILP model are presented here followed by mathematical model and the preprocessing method.

4.1.1 Sets and Parameters

$Technicians$	Set of technicians, {0, … , $number\ of\ technicians$ -1}
$Nodes$	Set of nodes including customers and the depot. i.e., $Nodes = \{0, \cdots, number\ of\ customers\}$, 0 denotes the depot
$Customers$	Set of customers. i.e., $Customers = \{1, \cdots, number\ of\ customers\}$
$ARCS$	Set of arcs from one node to another in $Nodes$ set
et_i, lt_i	Time window of node i. Installation service at node i can only

start within this time period.

$tech\_et_k$, $tech\_lt_k$ Time window of technician k. Technician k can only start work at a Customer's location within this time period

$basictime_i$ Basic service time of completing the job at node i. The basic service time at the depot is 0 since the depot is assumed as a node

$regionaldiff_i$ Regional difficulty of node i

$efficiency_k$ Efficiency of technician k

$actualtime_{ik}$ Actual service time for technician k at node i. i.e., $= basictime_i \times regionaldiff_i / efficiency_k$.

$Traveltime_{ij}$ Travel time from node i to node j, $(i,j) \epsilon ARCS$

f Vehicle operation cost per hour

M The big M parameter

$wage$ Hourly wage of technicians

$eligibility_{ik} = \begin{cases} 1 & \text{If technician } k \text{ can serve customer } i, \\ 0 & \text{Otherwise.} \end{cases}$

4.1.2 Variables

$x_{ijk} = \begin{cases} 1 & \text{If technician } k \text{ visits node } j \text{ right after node } i, (i,j) \epsilon ARCS \\ 0 & \text{Otherwise.} \end{cases}$

$v_k = \begin{cases} 1 & \text{If technician } k \text{ is assigned to work,} \\ 0 & \text{Otherwise.} \end{cases}$

$s_i \geq 0$ Service start time at node i.

$d0_k \geq 0$ Departure time of technician k from the depot

$a0_k \geq 0$ Arrival time of technician k to the depot

$O_{ik} \geq 0$ Waiting time of technician k at customer i's location

$u_i \geq 0$ Subtour elimination variable for customer i

4.1.3 Objective Function

Minimize:

$$\sum_{k \epsilon Technicians} wage * (a0_k - d0_k) + \sum_{(i,j) \epsilon ARCS,\, k \epsilon Technicians} f * Traveltime_{ij} * x_{ijk} \quad (4.1)$$

4.1.4 Constraints

Assignment Constraint

$$\sum_{i \epsilon Nodes} \sum_{k \epsilon Technicians} x_{ijk} = 1;\ \forall j \epsilon Customers \quad (4.2)$$

Starting from and Ending at the Depot

$$\sum_{j \epsilon Customers} x_{0jk} = v_k;\ \ \forall k \epsilon Technicians \quad (4.3)$$

$$\sum_{i \epsilon Customers} x_{i0k} = v_k;\ \ \forall k \epsilon Technicians \quad (4.4)$$

$$x_{ijk} \leq v_k;\ \ \forall i \epsilon Customers, \forall j \epsilon Customers, \forall k \epsilon Technicians \quad (4.5)$$

Flow Balance Constraint

$$\sum_{i \epsilon Nodes} x_{ijk} - \sum_{i \epsilon Nodes} x_{jik} = 0;\ \ \forall j \epsilon Customers, \forall k \epsilon Technicians \quad (4.6)$$

Sub-tour Elimination Constraint

$$u_i - u_j + n * \left(\sum_{k \epsilon Technicians} x_{ijk} \right) \leq n - 1;\ \forall i \epsilon Customers, \forall j \epsilon Customers \quad (4.7)$$

Time Window Constraints

$$et_i \leq s_i \leq lt_i;\ \forall i \epsilon Nodes \quad (4.8)$$

$$d0_k \geq tech\_et_k;\ \forall k \epsilon Technicians \quad (4.9)$$

$$a0_k \leq lt_0;\ \forall k \in Technicians \tag{4.10}$$

$$a0_k \geq d0_k;\ \forall k \in Technicians \tag{4.11}$$

Service start time Constraints

$$s_i + Traveltime_{ij} + actualtime_{ik} \leq (1 - x_{ijk}) * M + s_j;\ \forall\ (i,j) \in ARCS,$$
$$k \in Technicians, i, j! = 0 \tag{4.12}$$

$$d0_k + Traveltime_{0j} \leq (1 - x_{0jk}) * M + s_j;\ \forall\ j \in Customers,$$
$$k \in Technicians \tag{4.13}$$

Eligibility constraint

$$x_{ijk} \leq \frac{eligibility_{ik} + eligibility_{jk}}{2} \tag{4.14}$$

The objective function in 4.1 minimizes the total cost of serving all the customers. The objective function consists of the vehicle operation cost during driving and the salary of technicians during driving, waiting, and actual service. The salary of technicians can be calculated as the product of the hourly wages and the total working time of technicians from the time of departure to the time of return to the depot.

Constraint 4.2 forces each customer to be visited exactly once and by only one technician arriving from another node (depot or another customer).

Constraints 4.3, 4.4, and 4.5 jointly force technicians with some assigned jobs to start from and end at the depot. In equation 4.3, x_{0jk} equals 1 if j is the first customer visited by technician k. Similarly, in equation 4.4, x_{i0k} equals 1 if i is the last customer visited by technician k. The sum of x_{0jk} for each technician k and the sum of x_{i0k} both equal the binary variable v_k. If v_k equals 1, meaning that technician k is assigned with jobs, the technician must leave and return to the depot for only once. Otherwise, a technician without jobs would not leave or return to the depot, which means all the x_{0jk} and x_{i0k} are 0 for this technician k. Therefore, constraints 4.3 and 4.4 guarantee the consistency of leaving and returning to the depot. In constraint 4.5, all x_{ijk} should be less than or equal

to the binary variable v_k for each technician k. The binary variable v_k equals zero if technician k is not assigned with any jobs (i.e., all x_{ijk} for technician k is 0).

Constraint 4.6 forces a technician to leave one customer and then visit another one or return to the depot. Customers in each route have to be consecutively connected. In this equation, the total number of links pointing to each customer should equal the total number of links leaving from this customer in the route of technician k. Associating with constraint 4.2, the total number equals 1.

Constraint 4.7 is the Sub-tour Elimination Constraint proposed by Miller–Tucker–Zemlin (1960). This constraint eliminates tours not starting from and ending at the depot. Continuous variables u should be non-negative. This constraint forces $u_j \geq u_i + 1$, when customer j is visited directly after customer i (i.e., $\sum_{k \in Technicians} x_{ijk}$=1). If there is a sub-tour, the u_i value will also be greater than u_j value, which makes the above inequality invalid.

Constraints 4.8, 4.9, 4.10, and 4.11 are time constraints for customers and technicians. Constraint 4.8 forces service at each customer to start within their time windows. Constraint 4.9 restricts technicians' departure time from the depot to be no earlier than their earliest available time. However, the job does not have to be finished within the time window constraints of both the customer and the technician. Constraint 4.10 forces all technicians to return to the depot before the latest available time of the depot. Constraint 4.11 guarantees that each technician returns to the depot after departure time from the depot.

Constraint 4.12 uses the big M method to force the service start time to be no earlier than the arrival time at a customer. The LHS of the equation is the arrival time at customer j represented by the summation of service start time, actual service time of customer i, and the travel time from i to j. If technician k visits customer j right after serving customer i (i.e., $x_{ijk} = 1$), the RHS equals s_j, which is the service start time of customer j. Thus, the constraint guarantees that the delivering of service to a customer starts after the technician arrives. If $x_{ijk} = 0$, the RHS is a big number and the constraint is inactive under this condition. Constraint 4.8 and 4.12 jointly restrict the service start time to be the

maximum of the technician's arrival time and the earliest available time of the customer. Similar to constraint 4.12, constraint 4.13 forces the service start time of the first customer visited by each technician to be equal to or greater than the arrival time of technicians.

Constraint 4.14 considers the eligibility of technicians to serve each customer in view of both time window constraints and skill set constraints. Parameter $eligibility_{ik}$ equals 1 if technician k is eligible to serve customer i. Otherwise, the parameter equals 0. The LHS x_{ijk} equals 1 only if technician k is eligible to serve both customer i and customer j.

4.1.5 Preprocessing Phase

Instead of determining the eligibility of technicians in the MILP model, a preprocessing phase is adopted to determine the eligibility of technicians, which simplifies the MILP model considering the complex combination of time window and skill set constraints. Parameters required to determine the feasibility include the skill requirement and time window of each customer as well as skill sets and time window of each technician.

To be specific, for each technician-customer pair, the first step is to check if the skill required to complete the job is satisfied by the technician. The next step is to check if the technician is available during the time window of the customer. If both requirements are satisfied, the technician is eligible for the customer and the parameter eligibility equals 1 for this technician-customer pair.

4.2 Heuristic Algorithm

Heuristics are practical problem solving methods not guaranteed to find an optimal solution, but to find a satisfactory feasible solution respecting the current objective function. Heuristic algorithms can speed up the process of finding a near optimum solution and are widely used when searching for an optimal solution is impractical. As mentioned in Clarke and Wright (1964), most VRPs are NP-hard and large-scale problems cannot be solved in reasonable time. Thus, a heuristic method is proposed in the following to solve this problem. The heuristic method consists of two parts: initialization and local improvement.

4.2.1 Initialization

The first step of the heuristic method is to find a good initial solution since the selection of initial solution is of great significance to the final solution. The main steps of initialization are summarized as follows (Pseudo Code 4.1):

- Step 1: Count the number of eligible technicians that can serve each customer
- Step 2: Pre-allocate all customers that can only be served by $\leq n$ technicians. n is a predetermined parameter.
- Step 3: While there are unrouted customers:
  Select a technician (a route) and insert feasible unrouted customers into it based on Solomon's insertion heuristic (Solomon (1987))

These three steps in initialization are illustrated in detail as follows.

1. Count the number of eligible technicians

Due to time and skill sets constraints, there might be customers that can be served by only one technician in some cases. In order to avoid useless attempts and improve efficiency, customers with limited feasible choices should be allocated first, which is done in Step 2. Thus, it is necessary to check the number of eligible technicians for each customer in this step. If no customer has been assigned to a technician, the technician is eligible for a customer if both time window and skill set constraints are satisfied. However, when the route of a technician is not empty, besides satisfying the time window and skill sets constraints, the technician is eligible for the new customer only if the service start time of both the new customer and those current customers in the route is within their time windows respectively after the insertion of the new customer.

To be specific, when determining the feasibility of inserting a customer, we check the feasibility of inserting the new customer into each potential insertion place in the route and calculate the new service start time of each customer in the new route. The service start time of a customer j is the maximum of arrival time of the technician and the earliest available time of the customer j. Suppose that the current technician visits customer j directly after customer i, then the arrival time at customer j is the sum of the service start time s_i at customer i, the actual service time a_i at customer i, and the travel time t_{ij}

between customer i and customer j. Thus, the service start time for each customer can be expressed by the following formulation:

$$s_j = \max\{e_j, s_i + a_i + t_{ij}\} \tag{4.16}$$

The insertion place for the new customer is only acceptable if the service start time of all customers after the insertion is still within their time windows. If the customer can be inserted into at least one position of the current non-empty route, the technician is eligible to serve the customer. According to the above analysis, the process of checking the eligibility of a technician for a specific customer is summarized below.

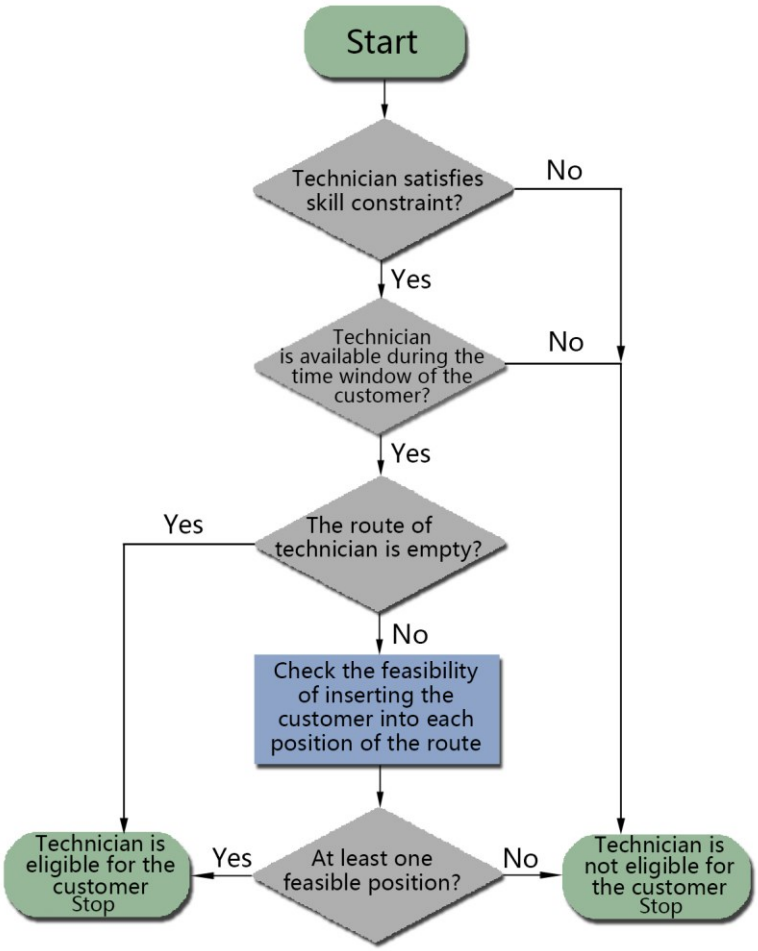


Figure 4.1 Process of checking the eligibility of a technician for a specific customer

2. Pre-allocation

After examining the eligibility of each technician following the above process, the number of eligible technicians for each customer can be determined. Customers with less than or equal to n eligible technicians are allocated. We first set n to be 1 and allocate customers that can only be served by one technician. When trying to allocate these customers, if the route of the only eligible technician is empty, the new route becomes depot-customer-depot. If the route is not empty and has more than one feasible place for the customer to be inserted, the insertion place is decided by Solomon's first type insertion heuristic (1987) (See Appendix A). Two factors are considered in this method to determine the best insertion place for a customer in a technician route. The first factor is the increase in total distance of the current route after the insertion; the second factor is the delay of service start time of the subsequent customer of the new customer in the current route. The customer is inserted into the feasible place with the lowest value of the sum of these two factors.

After the allocation of a customer, the eligibility of the technician might change and unrouted customers with only one eligible technician might appear again. Thus, after customers with only one eligible technician are allocated, the number of eligible technicians to serve all unrouted customers must be counted again. The newly generated customers with only one eligible technician should be allocated as a matter of priority until all remaining customers can be served by at least two technicians.

However, this pre-allocation does not guarantee a feasible solution. For example, a customer might have two eligible potential technicians. However, after the allocation of some customers, both these two technicians might become infeasible for the current customer. In this case, the procedure should restart from the pre-allocation stage by setting n to 2, which means once all customers with one eligible technician have been allocated, customers with two eligible technicians should be allocated. Since hourly wage of technicians are assumed equal, the higher efficiency a technician has, the less service time the technician will spend, which may result in more cost saving. Thus, the technician with higher efficiency will be selected to serve the customer in this method. If the route of the technician is empty, the new route becomes a single customer route.

Otherwise, the insertion place should be decided using Solomon's method (1987) as just described. Every time a new customer is allocated, the number of feasible technicians to serve each unrouted customer must be checked again and unrouted customers with only one eligible technician will be allocated first. The process repeats until all remaining customers can be served by at least $n + 1$ technicians. Similarly, after allocating all customers with less than or equal to n eligible technicians, if a feasible initial solution cannot be reached, the parameter n will be set to $n + 1$. The procedure repeats until a feasible initial solution is reached–or until the number of eligible technicians of a particular customer becomes 0, which means the data set is infeasible.

The process of allocating customers with less than or equal to n eligible technicians is described as follows.

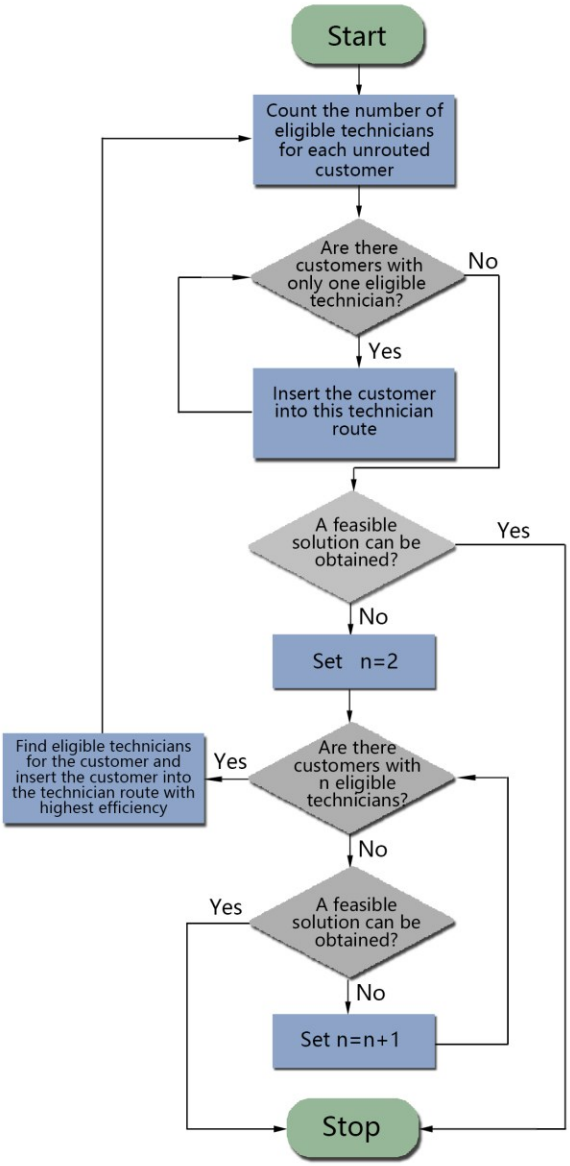


Figure 4.2 Process of pre-allocation

3. Select technicians and allocate unrouted customers

After the pre-allocation, the next step in initialization is to select a technician and then allocate customers into his route. As explained before, preference is given to more efficient technicians who can complete a job in less time, therefore potentially bringing greater income into the company. However, if the technician with the highest efficiency cannot serve any of the unrouted customers due to time or skill constraints, the technician

with the second highest efficiency is examined and the process repeats until a technician is selected.

Once a technician is selected to initialize a route, customers that can be served by the technician are then allocated into the route. If the route is empty, the algorithm will select the first customer to be allocated following the initialization in Solomon's heuristic (1987). In this algorithm, two initialization criteria are considered to determine the first customer in a route:

a) The farthest unrouted customer, and
b) The unrouted customer with the earliest deadline

The cost function for selecting the first customer i is calculated by:

$$C_i = -\alpha * d_{0i} + \beta * l_i, \alpha + \beta = 1;\ \alpha, \beta \geq 0; \tag{4.17}$$

Where d_{0i} is the distance between the depot and customer i, and l_i is the latest available time of customer i. The farthest unrouted customer is allocated first in order to avoid leaving the last unrouted customers widely apart. Among all customers that the current technician can serve, the one with the minimum value of the cost function is selected as the first customer to be visited. The two weights were derived empirically and the priority concern for the selection of the first customer is the distance (Thangiah et al. (1994)). Thus, these two weights were set to $\alpha = 0.9$ and $\beta = 0.1$.

After the first customer is selected, the algorithm selects the next customer to be inserted into the current technician route using two criteria proposed by Solomon' insertion heuristic (1987) (See Appendix A). The first criterion, as mentioned before, is applied to determine the best feasible insertion place for each unrouted customer that can be served by the technician. The second criterion for each customer is calculated as the difference between the distance from the depot to the customer to be inserted and the first criterion value. The customer with the minimum value of the second criterion is selected as the next customer to be inserted in the current route.

Every time a new customer is inserted into the current route, the feasibility of the technician might change and the process of checking eligibility should be implemented

again. The above process is repeated until all customers that can be served by the current technician without violating time constraints are inserted. Then a new technician is selected and customers are inserted. The process is repeated until all customers are assigned to technicians and a feasible initial solution is created. The process of selecting a technician and inserting customers into the route is as follows.

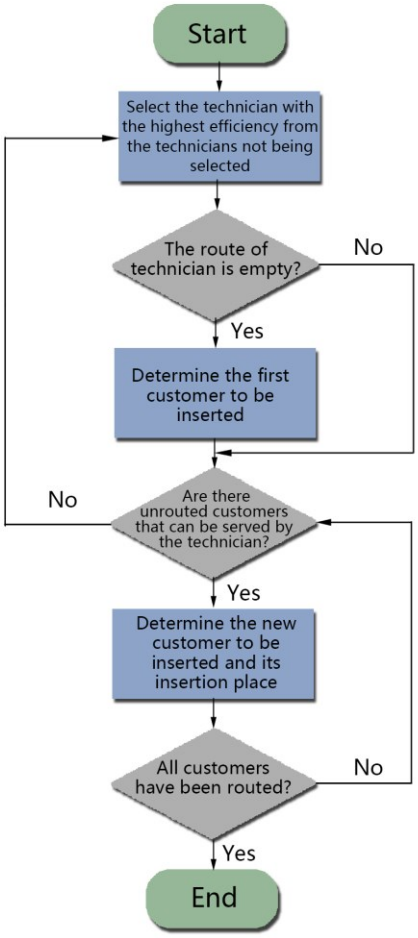


Figure 4.3 Process of selecting a technician and inserting customers

4.2.2 Local Improvement

Beginning with a feasible solution obtained from the initialization of this heuristic method, a local search procedure will repeatedly search for better solutions from the neighborhood of the current solution until no better solution can be found. The neighborhood of a feasible solution is a set of solutions generated by a specific

neighborhood structure. In the literature, several commonly applied neighborhood heuristics from previous researches on VRPTW are introduced. These neighbourhood heuristics can be classified into inter-route heuristics that improve the current solution by exchanging nodes between several routes and intra-route heuristics that achieve improvements by working on a single route.

Transfer (also called relocate) and swap (or exchange) neighborhood heuristics proposed by Prosser and Shaw (1996) are applied as inter-route heuristics in this research. A transfer heuristic relocates a customer from one route to another, while a swap heuristic exchanges two customers between two routes. Among inter-route neighborhood heuristics, transfer is proved to be the most powerful operator leading to a better solution among four neighborhood heuristics in Prosser and Shaw (1996). Besides, transfer and swap heuristics only change at most one customer at a time from a route. Considering the tight constraints of time and skills in the problem under study, the feasible neighborhood in each iteration is very limited. Big changes from the previous solution may only lead to infeasible solutions. Thus, although some neighborhood heuristics such as 2-opt* and Shift-sequence are effective in VRPTWs (Potvin and Rousseau (1995), Schulze and Fahle (1999)), they sometimes result in big changes from the previous solution and spend a lot of time on searching for areas that can only generate infeasible solutions.

Once customers assigned to a specific technician are determined, the total actual service time of the technician is fixed. The route of the current technician can be improved in two ways. The first one is to rearrange the customers by sub-tour reversal in order to reduce the total distance traveled, thereby reducing traveling cost. Besides, we decide to delay departure time from the depot in order to potentially reduce unnecessary waiting time. We operate both methods for intra-route improvements. The intra-route heuristics are applied every time two new routes are generated by a transfer or swap neighbourhood heuristic.

In summary, our local search starts off finding all feasible neighbors of the initial solution through transfer heuristic and replaces the current solution with the neighbor leading to the most decrease in total cost. The process repeats until no more improvement can be achieved by any feasible neighbors of the current solution. Then the swap heuristic is

applied and the solution is repeatedly replaced by the best feasible neighbor with improvement until no further improvement can be achieved. If swap heuristic leads to improvement, transfer heuristic is applied again. The entire process terminates if neither of these two heuristics can bring about any improvement. The intra-route heuristics are applied after the generation of each feasible neighbor.

The three heuristics are explained in greater detail in the following.

1. Intra-route heuristics

In intra-route heuristics, we rearrange the customers by sub-tour reversal to shorten the travel distance and delay technicians' departure time from the depot to reduce the waiting time.

In the process of rearrangement, only customers with the same time windows are allowed to be exchanged because there is no overlapping time between time windows of customers. For every sequence of two consecutive customers with the same time windows, the algorithm exchanges these two customers and calculates the new service start time of all customers in the current route. If the service start time of all customers is within their time windows, the cost of this new route is calculated. The new route will be accepted if it leads to a decrease in cost. If sequences of two consecutive customers with the same time windows exist in the same route, all the sequences of three consecutive customers with the same time windows will be examined. These sequences will be reversed and the service start time is checked again. The process repeats until no more consecutive customers with the same time windows can be reversed for improvement. Through this procedure, the current route can be improved and the sequence of visiting customers can be determined. Figure 4.4 provides an example of sub-tour reversal in which the sequence of two customers A and B is reversed and then the sequence of three customers C, E, and F is reversed.

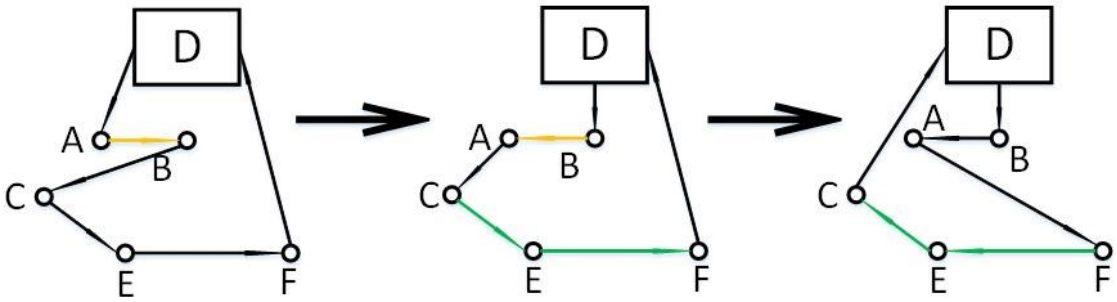


Figure 4.4 Sub-tour reversal

Waiting time is incurred by early arrival at a customer's location. Given that technicians are paid from the time they leave the depot to the time of return, it may be beneficial to delay technicians' departure time from the depot on the premise of not violating the time window constraints. The original service start time of each customer and the waiting time of the technician at each customer's location are initially calculated under the assumption that all technicians leave the depot at their earliest available time. The original service start time is calculated by equation 4.16. Suppose the current technician visits customer j after customer i, waiting time at customer j is calculated by:

$$w_j = \max\{0, e_j - s_i - a_i - t_{ij}\} \tag{4.18}$$

where s_i is the service start time of customer i, a_i is the actual service time of customer i, t_{ij} is the travel time between customer i and j, e_j is the earliest available time of customer j, and $s_i + a_i + t_{ij}$ is the arrival time at customer j's location and e_j is the earliest available time of customer j as mentioned in equation 4.16. Waiting time at each customer is the difference between the earliest available time of a customer and the technician's arrival time at this customer's location if technician arrives before the earliest time. Otherwise, waiting time equals 0, meaning the service starts once the technician arrives.

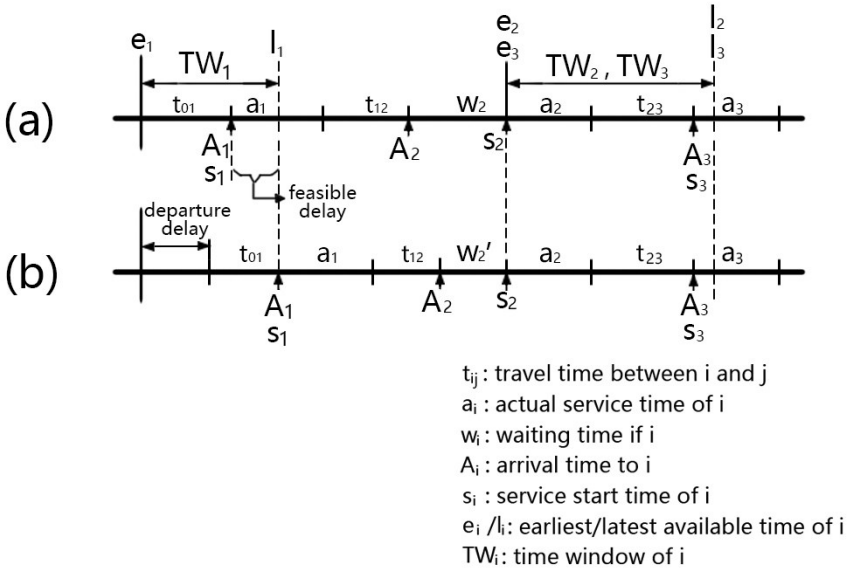


Figure 4.5 Waiting time improvement when feasible delay is less than waiting time

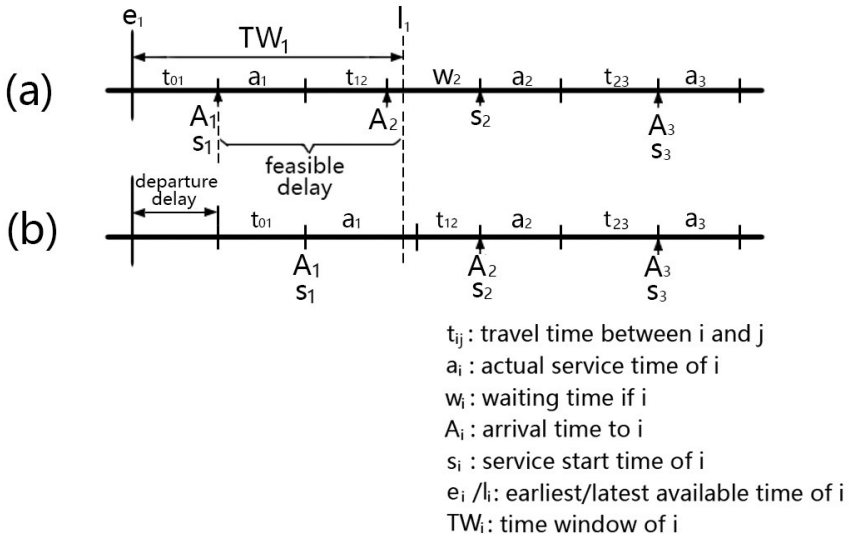


Figure 4.6 Waiting time improvement when feasible delay is greater than waiting time

Figure 4.5 and 4.6 provide examples of delaying departure time under different situations. Assume the technician leaves the depot at his earliest available time (Figure 4.5 (a), Figure 4.6 (a)), the non-zero waiting time w_2 arises since the technician's arrival time A_2

is earlier than the earliest available time e_2 of $customer_2$. When trying to minimize the waiting time, we calculate the feasible delay of $customer_1$ as the difference between the latest available time and the current service start time of this customer (equation 4.19).

$$feasible\_delay_i = l_i - s_i \tag{4.19}$$

Then the technician can delay his departure time by the feasible delay at most. If the technician visits more than one customer before visiting the customer u with a non-zero waiting time, the maximum feasible delay of the technician is calculated as:

$$\max\_delay = \min(feasible\_delay_0, feasible\_delay_1 \ldots feasible\_delay_{u-1}) \tag{4.20}$$

In Figure 4.5, the feasible delay of $customer_1$ is less than the waiting time w_2. Thus, the technician will delay the departure time and service start time at $customer_1$ by the feasible delay. The waiting time also decreases as much as the feasible delay. In Figure 4.6, in contrast, the feasible delay of $customer_1$ is greater than the waiting time w_2. The technician will delay the departure time from the depot by the waiting time w_2 and the waiting time decreases to 0. In both situations, the service start time of $customer_2$ and subsequent customers will not change. The departure delay can be calculated as:

$$departure\_delay = \begin{cases} waiting\_t, & if\ waiting\_t < \max\_delay \\ \max\_delay, & Otherwise \end{cases} \tag{4.21}$$

If the maximum feasible delay is greater than the non-zero waiting time in the route, once the waiting time is eliminated, the next non-zero waiting time in the route can be dealt with in the same way. Otherwise, the entire waiting time improvement process should be terminated since no feasible delay is available considering time window constraints of customers.

The two intra-route improvement methods are combined together to improve a single route in this research. Once customers allocated to a technician are determined, feasible sequences of visiting these customers are identified through the sub-tour reversal method. For each feasible arrangement, if non-zero waiting time exists, the waiting time improvement method should be applied to reduce the waiting time. Once the new departure time from the depot is determined, the total cost of this route with the current

order can be calculated. The sequence with the lowest cost among all the feasible sequences is recorded as the best one. The process of intra-route improvement of a specific route is described as below (Pseudo Code 4.2):

- Set n=2

  While there are sequences of n consecutive customers with same time window:

    For every sequence of n consecutive customers with the same time windows:

      Reverse the sequence

      If this new route after the reversal satisfies the time constraints:

        Calculate the waiting time at all the customers' location

        If non-zero waiting time exists:

          Delay the departure time

    Set $n = n + 1$

  The sequence with the lowest cost is the best route for the technician with given customers

2. Transfer Heuristic

When looking for an improvement over the current solution by transferring a specific customer, all eligible technicians and feasible insertion positions in the route of these technicians should be identified and compared to find an optimal position for this customer to be inserted. To be specific, if there are eligible technicians for a specific customer other than its current technician, the customer is removed from the current route and this route after removing the customer is improved by the intra-route heuristics. The customer is then tested for insertion into each new eligible technician using Solomon's insertion heuristic (1987). Each new route of the eligible technicians after the insertion of this customer is improved by the intra-route heuristics. Thus, the transfer heuristic only changes two routes at a time, while other routes remain the same. If the total cost of these two new routes is less than that of their original routes, there is an improvement and the new solution is recorded.

Assume customer C is currently in the route of technician k_1 and technician k_2 and k_3 are eligible for customer C to be inserted, Figure 4.7 provides an example of transferring customer C from the route of technician k_1 to the route of technician k_2 and k_3.

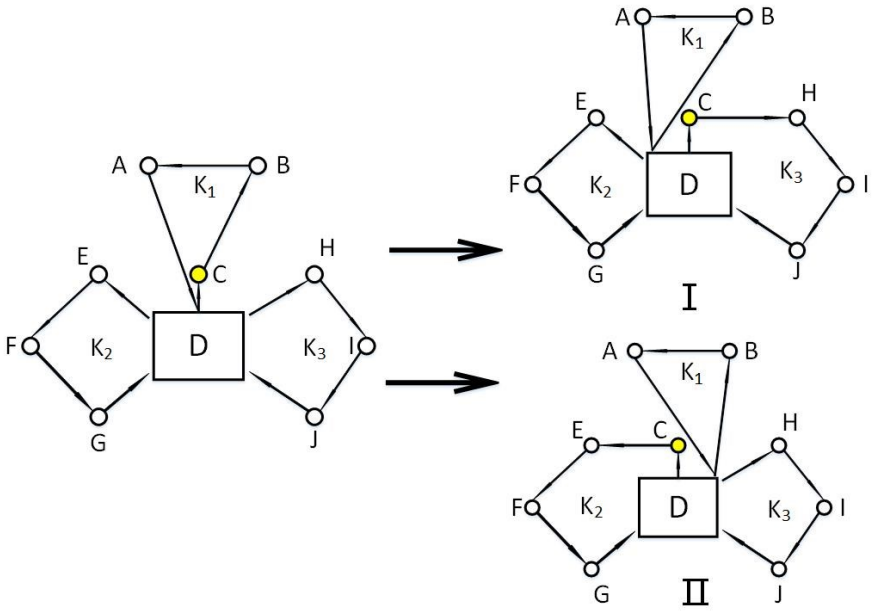


Figure 4.7 The transfer of customer C

Once all eligible technicians for the current customer are examined, the customer is transferred to the route leading to the lowest cost. If no improvement can be achieved by transferring the customer, the solution will remain the same.

This Procedure repeats for all customers until no more improvement can be achieved by transferring customers. The process of transferring a specific customer i is described as (Pseudo Code 4.3):

- Step 1: Find eligible technicians of customer i through the process of Figure 4.1
- Step 2: Remove customer i from the original route and improve the original route after removing customer i using intra-route heuristics
- Step 3: For each eligible technician k in the current solution:
  Insert customer i into the route of technician k
  Apply intra-route heuristics to improve the new route
  Calculate the new total cost

- Step 4: If at least one transfer attempt leads to the decrease in cost:

  Relocate customer i into the route leading to the most decrease in cost

3. Swap Heuristic

The first step in swap heuristic is to find potential eligible technicians for each customer. A technician is potentially eligible for a customer if he satisfies the skill requirement and is available during the time window of the customer without considering the allocated customers in his route. The feasibility of inserting the new customer into the current route of this potentially eligible technician is not checked here, because a customer in the current route of the technician will be removed in the swap heuristic and this will influence the feasibility of inserting the new customer into this route.

Similar to transfer heuristic, all possibilities of swapping a particular customer with another customer from a different route are explored to seek improvement. The current customer can be swapped with another customer only if their original technicians satisfy the skill requirements of both customers and if the service start time of all the customers in the two new routes is within their time windows after the swap. If the time window and skill sets constraints are both satisfied by swapping the two customers, these two new routes after the swap are tested for further improvement by intra-route heuristics. After analyzing all the possibilities of swapping the current customer with other customers in the route of all potential eligible technicians, the best solution leading to the decrease in cost will be selected. Otherwise, the solution remains unchanged.

Suppose customer C is currently in the route of technician k_1 and technician k_2 and k_3 are potentially eligible for customer C to be swapped with another customer, Figure 4.8 provides an example of swapping customer C with customer E in the route of technician k_2 and customer H in the route of technician k_3.

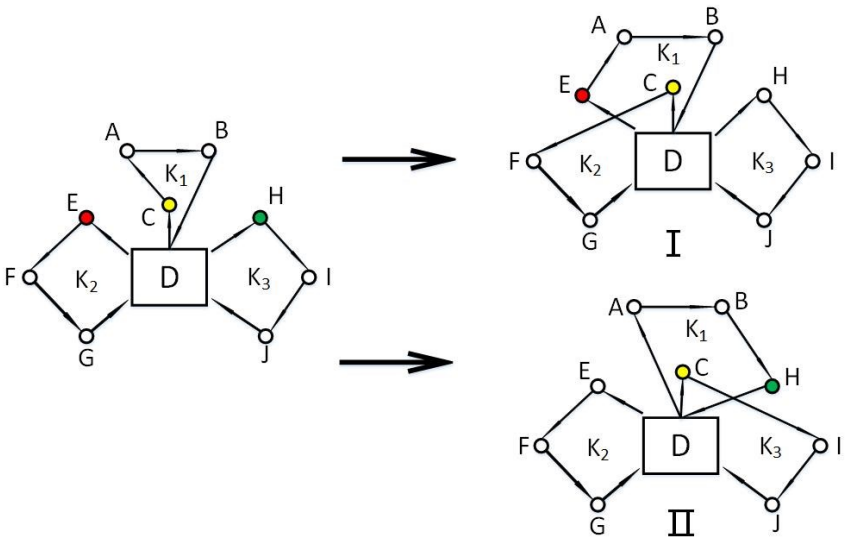


Figure 4.8 The swap of customer C

The process above repeats for all customers in each route until no more improvement can be achieved by swapping the customers. The process of swapping a specific customer i with another customer which lead to the most savings on cost is as below (Pseudo Code 4.4):

- Step 1: Find potential eligible technicians of customer i in consideration of time windows and skill sets constraints
- Step 2: Remove customer i from the original route and improve this new route after the remove of i by intra-route heuristics
- Step 3: For each potential feasible technician k_2:

  For each customer j in the route of technician k_2:

  Remove customer j from the route

  If customer i can be inserted in the new route of k_2 and j can be inserted in the new route of k_1:

  Swap customers i and j and apply intra-route heuristics to the two new routes after

  Calculate the new total cost and the decrease in cost
- Step 4: If at least one transfer attempt leads to the decrease in cost:

  Swap customer i with the customer that can lead to the most decrease

4.3 Metaheuristic Algorithms

A heuristic model finds a local optimum through the local improvement process, while a metaheuristic model escapes from a local optimum and looks for better local optimum in the hope to land at the global optimum (Hillier (2012)). This sometimes means to accept inferior solutions to the local optimum. Accepting worse solutions is the essential property of some metaheuristic algorithms since it allows for searching for the global optimal solutions in a more extensive region. The process of searching optimum or near optimum solution is like climbing the highest hill among several hills for maximization problems. The highest hill can be regarded as the global optimum solution for a specific problem. Rather than stopping at the top of the current hill, some metaheuristic algorithms allow the process to search a little way down the hill until it can start climbing to the top of another hill. Even if it reaches the global optimum, the system will not realize it and will continue searching until a stopping rule is applied. For minimization problems like the problem on hand, the process of searching for global optimum is like looking for the lowest foot of several mountains (see Figure 4.9).

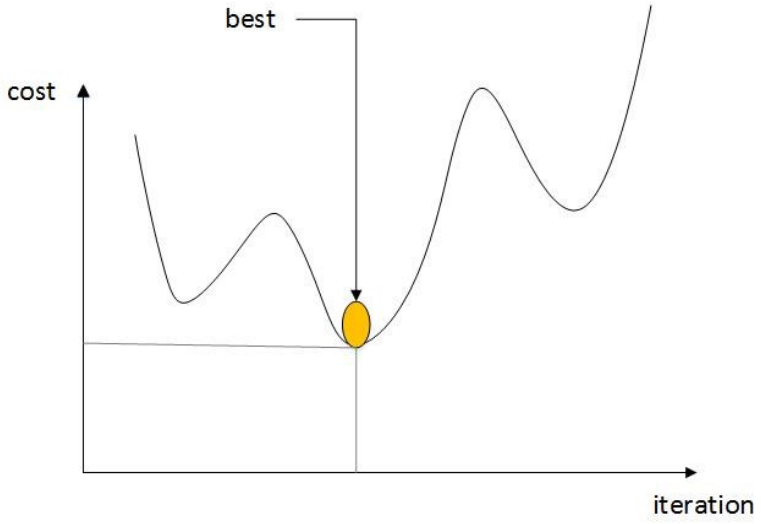


Figure 4.9 Searching for the best solution

In this part, two metaheuristic algorithms are applied to search for global optimal solution. These two algorithms are tabu search and simulated annealing.

4.3.1 Tabu Search

Tabu Search (TS) has a strong skill in "climbing the mountain" (Hillier (2012)). Tabu search applies a local search procedure to find a local optimum. It accepts the best solution in the neighborhood of the local optimum even if it is a non-improving solution. However, once a local optimum is reached, the method may lead back to the same local optimum in further iteration. To avoid this, a tabu list is applied to record a predetermined number of recent changes and forbids moves reversing these changes temporarily. This memory function permits a tabu search to jump out of local search and explores other areas. The framework of a basic tabu search is as:

- Initialization: Begins with a feasible initial solution
- Iteration: During each iteration, it applies neighbourhood structure to find all neighborhood of the current solution through local search. Then it gets rid of any move that is currently forbidden by tabu list unless the move leads to the best solution so far. For all other moves, the one with the best solution is adopted as the new solution no matter if it leads to improvement to the current solution or not. Then the tabu list is updated to avoid a cycle back to current solution. If the tabu list is full, the oldest member in the list is removed to bring more flexibility for future moves.
- Stopping rule: Certain stopping criterion must be adopted to end the iterations. Commonly used stopping rules include a fixed number of iteration, a fixed number of consecutive iterations without an improvement, or a fixed amount of computer running time. Then the best solution from all iterations is accepted as the final solution.

In applying a tabu search metaheuristic, several details must be worked out to fit a particular problem before implementation. These details include the neighborhood structure, the length of tabu list, the form of tabu moves, and the stopping rule adopted.

In our tabu search algorithm, we use the combination of transfer and swap neighbourhood heuristics as the neighbourhood structure. The length of tabu list is decided based on parameter analysis presented in the next chapter. When applying transfer neighbourhood heuristic, the tabu move consists of the customer being relocated and its original route in

each iteration. In swap heuristic, since two customers from two different routes are exchanged, the tabu move consists of the two customers being swapped and their original routes.

The local optimal solution generated in the local improvement section is applied as the initial solution in tabu search algorithm. In each iteration using transfer neighborhood heuristic, all feasible neighbors of the current solution are explored. The immediate neighbor with the lowest total cost among all neighbors is selected as the new trial solution if it leads to lower cost than the current best solution, or if the move is not in the tabu list. Otherwise, this neighbor is eliminated from consideration and the next best neighbor is examined until the move of a neighbor is not forbidden by the tabu list. Once a neighbor is selected as the new trial solution, the customer being relocated and its original route are added to the tabu list. If the tabu list is full, the oldest tabu move is deleted. The above procedure repeats and generates a new solution in each iteration. We record the solutions obtained from each iteration and calculate the total cost.

After a fixed number of consecutive iterations without an improvement, the above procedure stops. The solution with the lowest cost among all the iterations applying transfer neighborhood heuristic is recorded as the initial solution for tabu search applying swap heuristic. Then the same procedure repeats using swap heuristic until a fixed number of non-improvement consecutive iterations are reached. The entire process repeats until another predetermined number of iterations without an improvement is executed using both transfer and swap heuristics.

Suppose TI is the predetermined total number of consecutive iterations without an improvement, and I is the predetermined number of consecutive iterations without an improvement by applying transfer or swap neighbourhood heuristic. The process of tabu search metaheuristic is described as follows:

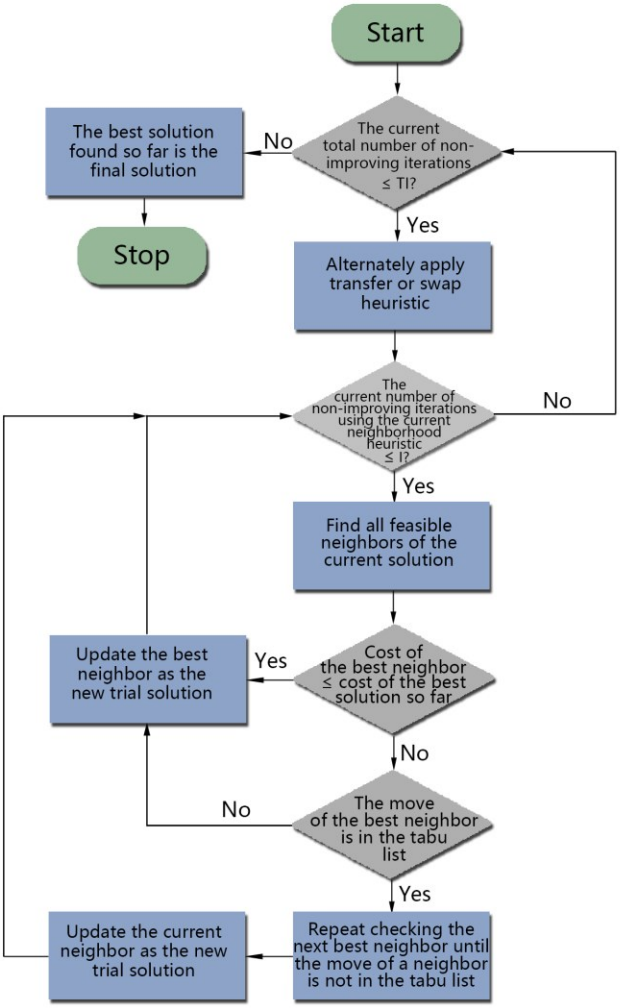


Figure 4.10 Process of tabu search

4.3.2 Simulated Annealing

The simulated annealing algorithm (SA) simulates the process of annealing in metal work (Hillier (2012)). This physical annealing process initially melts a metal or glass at a high temperature and gradually cools the substance until a stable state with preferred physical properties is achieved. Early emphasis of simulated annealing algorithm is on exploring as much feasible region as possible by taking steps in random directions and then the emphasis gradually turns toward climbing upward by rejecting an increasing portion of downward moves. Therefore, the process will often reach the top of the tallest hill given enough time.

Specifically, the basic process of simulated annealing, just as for tabu search, is to move from the current solution to a random neighbor from the local neighborhood in each iteration. However, it differs from the tabu search in the selection of the immediate neighbor to be the new solution. Let

Z_c= Objective function value for the current solution

Z_n= Objective function value for the next solution

T = A parameter influencing the probability of accepting a worse solution

Simulated Annealing algorithm creates a candidate solution randomly. Supposing a minimization problem, the move selection rule for determining whether the candidate solution can be accepted as the next trial solution is:

If $Z_n \leq Z_c$, always accept the current candidate as the next solution.

If $Z_n > Z_c$, accept the candidate with the probability of:

Probability {acceptance} $= e^{\frac{Z_c - Z_n}{T}}$

Thus, if the current candidate has better objective function value than the current solution, it is always accepted as the new solution. If it is worse, the probability of acceptance depends on the parameter T and the degree of how worse it is. Obviously, the algorithm normally accepts candidate that is slightly worse than the current solution.

In this research, the local optimal solution generated in local search procedure is applied as the initial solution in simulated annealing algorithm as well. The combination of transfer and swap neighbourhood heuristics is applied as the neighbourhood structure to search for new trial solutions. When applying transfer heuristic, we randomly select a customer and find all eligible technicians of this customer. If there is more than one eligible technician other than the current technician, the candidate solution is generated by transferring this customer into the route of a randomly selected eligible technician. If the customer cannot be relocated to other routes, another customer is randomly selected until a feasible neighbor of the current solution is generated. As for swap heuristic, our algorithm searches all feasible neighbors of the current trial solution in each iteration

since the number of feasible neighbors generated by swap heuristic is fewer than transfer heuristic. Then the candidate solution is randomly selected from the feasible neighbors.

An essential question when designing a simulated annealing algorithm is to determine an appropriate temperature (T) schedule. The algorithm initially sets a relatively high temperature and then allows it to slowly decrease. This temperature schedule gradually decrease the probability of accepting worse solutions, therefore forcing the algorithm to gradually focus on the search area that has the potential to generate a close to optimum solution after searching for large feasible areas. The temperature scheme usually includes the starting temperature, the cooling down parameter, and the number of temperature values.

In this research, the transfer neighbourhood heuristic is first applied to generate immediate neighbors in simulated annealing algorithm. Once a predetermined number of iterations are performed at each temperature value, the procedure stops. The swap neighborhood heuristic is then applied and the above process repeats. The best solution found is selected as the initial solution for the. The procedure of simulated annealing of using either neighborhood heuristic is given as follows:

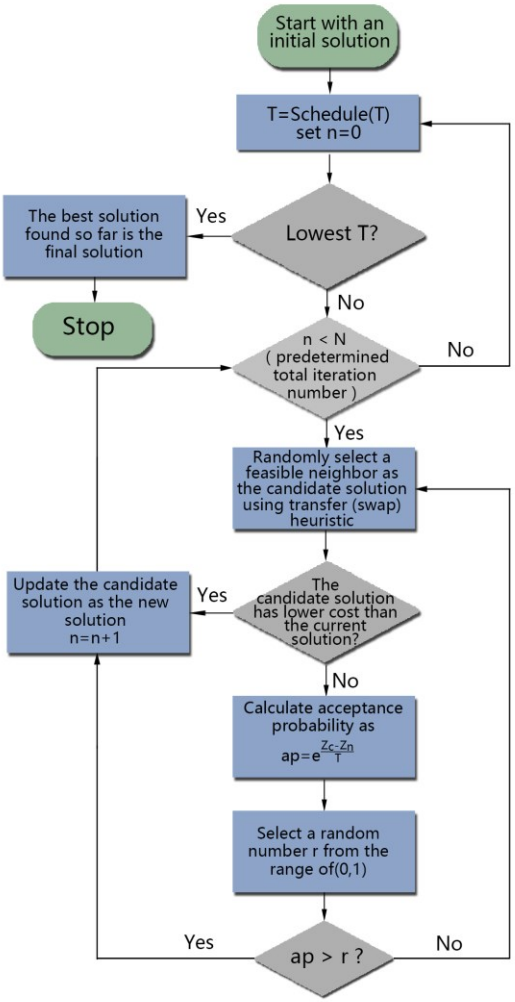


Figure 4.11 Process of simulated annealing

Chapter 5 Results and Discussions

In the methodology section, MILP model, heuristic algorithm as well as metaheuristic algorithms are proposed to solve the VRPTWSS. For metaheuristic algorihms, both tabu search and simulated annealing algorithms are applied. The neighborhood structure used in heuristic and metaheuristic algorithms is the combination of transfer and swap heuristics. In this chapter, a simulation process written in Python is used to generate data and several examples are created by the simulation process to validate, verify and compare different methods applied in this research. The MILP model is implemented in GLPK 0.2.12 and Gurobi 7.0, while heuristic and metaheuristic algorithms are written and conducted in Python 2.7.11. All the software is operated on an E1-SXTNTLIB-105 personal computer (CPU clock rate: 3.30GHz; RAM: 8.00 GB).

These four methods are compared based on the total cost and computation time. Since both GLPK and Gurobi are only capable of solving small-scale examples within reasonable time, we only provide comparisons of different algorithms by testing examples with 10, 15, 20, and 25 customers. Finally, heuristic and metaheuristic algorithms are examined on large-scale examples.

5.1 Data Simulation

Three distinct data sets can be assumed for the problem at hand: customer data (including the depot for simplicity's sake), technician data, and travel time between each pair of nodes (depot and customers). The following section describes the data generation procedure in further detail.

5.1.1 Customer data

Among customer-related data, the quantity of customers determines the scale of the problem. Other data involve the skill requirement, regional difficulty, basic service time, and time window of each customer.

In our examples, we assume that there are five different skill types denoted by $1, 2, 3, 4, 5$ and 1 is the basic skill every technician masters. The depot has actual service time of 0

and all the technicians should depart from it. Thus, the depot can be regarded as a customer with basic service time of zero and the basic skill requirement 1.

Basic service time is the estimated time to perform the job and is evaluated based on the job type and requirement of each particular customer. The basic service time of customers is randomly selected from the range of 30 to 90 minutes with equal chance.

Regional difficulties reflect the condition of infrastructure at each customer's location. In our examples, Regional difficulties are assumed to be greater than or equal to 1. Regional difficulty of 1 means that the regional factor of a particular customer does not increase the actual service time. In this problem, we suppose that most customers are located in urban areas with necessary infrastructure in good condition. Thus, regional difficulties of most customers' location are close to 1 and they are set to be random parameters with a Pareto distribution that matches our assumption according to its probability density function (see Figure 5.1). Potential problems of lack or disruption of necessary infrastructure that cannot be solved within reasonable time is excluded from this problem, so the values of regional difficulties are also set to be less than 2. Regional difficulty of the depot is set to be 1.

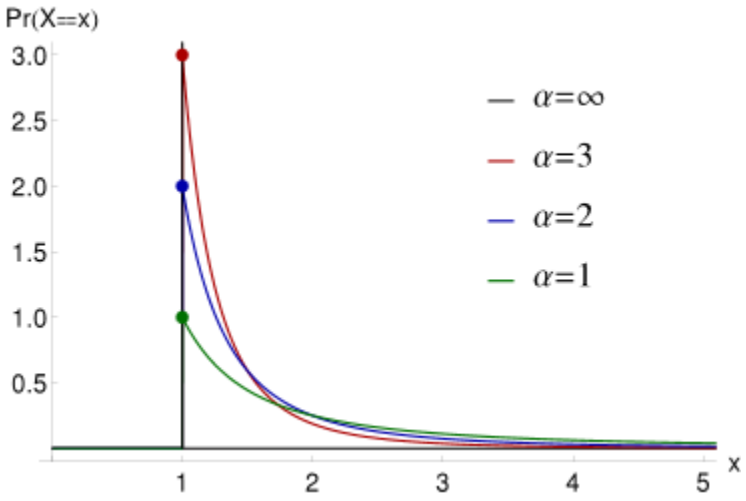


Figure 5.1 Probability density function of Pareto distribution (Danvildanvil (2014))

In our examples, time windows of customers are assumed to have the same length of two hours and are randomly selected from the four possible choices with equal chance: 8 a.m. to 10 a.m., 10 a.m. to 12 p.m., 12 p.m. to 14 p.m., and 14 p.m. to 16 p.m. We also assume

that all technicians are required to leave the depot after 8 a.m. and return before 18 p.m. Thus, time window of the depot is set to be from 8 a.m. to 18 p.m.

5.1.2 Technician data

Technician-related data are the number of technicians, working efficiencies, availabilities and skill set of the technicians.

The number of technicians is determined based on the number of customers. In our examples, in order to improve the feasibility of simulated data, when the number of customers is small, the ratio of the number of customers to the number of technicians is set to be relatively small because the random characteristic of data requires more technicians to deal with the uneven distribution of time windows and skill requirements of both technicians and customers. With the increase of customers, the number of technicians increases as well and the proportion is set to be larger. For relatively large-scale problems, the proportion is preset to be 3.2 according to the calculation of average number of customers a technician can serve. The average available time for a technician is 6 hours and the average travel time between every two nodes is 40 min. The average basic service time is 60 min and the average working efficiency of each technician is 1. Besides, the regional difficulty is estimated as 1.2 on average. Thus, each job requires 112 min including travel time and service time and each technician can therefore serve around 3.2 customers on average. To be specific, the amount of technicians is calculated by following formulation.

$$num_{technician} = \begin{cases} \dfrac{num_{customer}}{2.5}, & num_{customer} \leq 10 \\ \dfrac{num_{customer}}{3}, & num_{customer} \leq 20 \\ \dfrac{num_{customer}}{3.2}, & else. \end{cases} \tag{5.1}$$

According to equation 5.1, there are 4, 5, 6 and 7 technicians respectively used in problems with 10, 15, 20, and 25 customers.

The efficiencies of technicians must be greater than 0. In this problem efficiencies are designed to follow normal distribution with 1 as mean and 0.4 as the standard deviation. Efficiencies less than or equal to 0 will be removed.

To increase the feasibility of data, all technicians are assumed to have the basic skill 1 and two more among the five skills. As for availabilities, all technicians are assumed to have at least four hours' available time randomly selected from the following time with equal chance: 8 a.m. to 12 p.m., 8 a.m. to 14 p.m., 8 a.m. to 16 p.m., 10 a.m. to 14 p.m., 10 a.m. to 16 p.m., 12 p.m.to 16 p.m. To reduce the probability of generating infeasible problem, the first two technicians are designed to have the complementary skill sets: [1, 3, 5] and [1, 2, 4] and have the full availability from 8 a.m. to 16 p.m. Therefore, every customer has at least one qualified technician considering both time window and skill set constraints.

5.1.3 Travel time

The travel distance is regarded as straight-line distance in this problem for simplicity reasons. Given the latitude and longitude of all the nodes, the distances between every two nodes can be calculated following Haversine formula. The haversine formula is extensively used in navigation research and it calculates the shortest distance between two points given their longitudes and latitudes.

According to the Haversine formula, the first step is to convert the degree difference of longitudes and latitudes into radians. The haversine of the central angle can be calculated by the following equation:

$$hav\left(\frac{d}{r}\right) = hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)hav(\lambda_2 - \lambda_1) \quad (5.2)$$

, where hav is the haversine function:

$$hav(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2} \quad (5.3)$$

In the Haversine formula, d is the spherical distance between two points; r is the radius of earth, 6378.7 in kilometers. φ_1, φ_2 denote the Latitudes of two points in radians, and λ_1, λ_2 denote the Longitudes.

After some mathematical transformations, distance between two points can be written as:

$$d = 2r \arcsin(\sqrt{hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)hav(\lambda_2 - \lambda_1)})$$

$$= 2r \arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + cos(\varphi_1)\, cos(\varphi_2)\, sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \qquad (5.4)$$

The haversine formula provides computationally exact results even at small distances according to Cassa et al. (2005). Therefore, this formula can be used in this problem with small distances. The travel time between every two nodes is calculated by dividing the distance by the speed.

In our problem, maximum travel time between two arbitrary nodes in this problem is designed to be less than 80 minutes. The average speed is estimated to be 40 kilometers per hour in consideration of the complex road conditions and speed limit in cities. Thus, the maximum distance between every two nodes should be less than 53.33 kilometers. In our examples, two points are randomly selected with the straight-line distance to be around 53.33 kilometers. These two nodes are regarded as the two opposite corners of a rectangular. Based on these two nodes, the longitudes and latitudes of all customers and the depot are randomly selected within this rectangular. This guarantees travel time between every two nodes to be within 80 minutes.

In data simulation phase, because of the randomness of data, infeasible data sets might exist. For example, suppose two customers with the same time windows can only be served by one technician. After one customer being allocated in the route of the technician, the other customer may not be served by the technician. Under this condition, the process will terminate and new data sets will be generated.

In the data simulation phase, we assume the time windows of customers do not partially overlapped. In a more general case in which customers have the freedom to select any time periods as their time windows, the MILP model will not change. As for the heuristic algorithm, the sub-tour reversal intra-route improvement heuristic will be changed, while other improvement heuristics remain the same as the current algorithm. The sub-tour reversal will reverse sequences of consecutive customers with the same overlapping time windows instead of only reversing customers with the exactly same time windows. The selection of all other data will not influence the algorithms developed in this research.

Thus, the algorithms in this research can be applied to other VRPTWSS problems with only slight modifications.

5.2 Selection of parameters for metaheuristic algorithms

Before comparing different methods, several parameters for both tabu search and simulated annealing should be determined. The selection of parameters exerts a significant influence on solutions and computation time. Normally, a larger number of iterations has higher opportunity of generating better solutions but will take more computation time. In this section, we select parameters by investigating several examples.

5.2.1 Tabu Search parameters

Tabu search parameters to be determined in this problem include the length of tabu list, the number of non-improved consecutive iterations for transfer and swap neighbourhood heuristics, and the total number of non-improving consecutive iterations.

We tend to select a large number of iterations to search for larger areas. In the meanwhile, it is necessary to take into account the complexity of our problem and the goal of saving computation time.

To determine the number of iterations, we generate 10 sets of data for each scale of problems with 10, 15, 20, and 25 customers and use the combinations of (10, 40), (20, 80), and (50, 400) as the number of non-improved consecutive iterations for each neighborhood heuristic and the total number of non-improved consecutive iterations to study the influence of different parameter combinations on optimality and computation time. The results of employing different parameters on different scales of problems are shown in Table 5.1.

Table 5.1 Comparison of Tabu Search parameter combinations

			Tabu Search								
	Param		10, 40			20, 80			50, 400		
	Examples	Optimal	Cost	Time	optimality%	Cost	Time	optimality%	Cost	Time	optimality
10 Customers	1	288.30	288.30	2.04	100.00%	288.30	2.84	100.00%	288.30	5.36	100.00%
	2	384.24	384.24	2.35	100.00%	384.24	3.89	100.00%	384.24	11.15	100.00%
	3	440.43	445.88	2.69	98.78%	445.88	3.03	98.78%	445.88	8.29	98.78%
	4	509.11	509.11	0.26	100.00%	509.11	0.26	100.00%	509.11	0.26	100.00%

Table 5.1 Comparison of Tabu Search parameter combinations (Continued)

10 Customers	5	453.49	459.95	0.27	98.60%	459.95	0.27	98.60%	459.95	0.28	98.60%
	6	330.75	330.75	1.92	100.00%	330.75	3.17	100.00%	330.75	11.47	100.00%
	7	665.74	665.74	0.21	100.00%	665.74	0.18	100.00%	665.74	0.64	100.00%
	8	432.51	432.51	1.99	100.00%	432.51	2.47	100.00%	432.51	5.51	100.00%
	9	381.31	381.31	1.27	100.00%	381.31	1.44	100.00%	381.31	4.45	100.00%
	10	533.63	533.63	0.69	100.00%	533.63	1.30	100.00%	533.63	2.16	100.00%
	Avg	441.95	443.14	1.37	99.74%	443.14	1.89	99.74%	443.14	4.96	99.74%
15 Customers	1	333.27	333.27	5.50	100.00%	333.27	10.03	100.00%	333.27	32.80	100.00%
	2	405.78	405.78	4.82	100.00%	405.78	19.20	100.00%	405.78	53.97	100.00%
	3	389.07	389.07	7.22	100.00%	389.07	12.63	100.00%	389.07	112.06	100.00%
	4	444.52	444.52	4.89	100.00%	444.52	8.11	100.00%	444.52	25.94	100.00%
	5	381.53	381.53	5.97	100.00%	381.53	9.73	100.00%	381.53	31.03	100.00%
	6	410.98	420.19	4.48	97.81%	420.19	7.75	97.81%	410.98	36.79	100.00%
	7	571.12	571.12	2.27	100.00%	571.12	4.45	100.00%	571.12	23.04	100.00%
	8	734.26	734.26	4.23	100.00%	734.26	6.98	100.00%	734.26	19.46	100.00%
	9	858.24	858.24	2.36	100.00%	858.24	4.68	100.00%	858.24	15.81	100.00%
	10	656.70	689.93	5.20	95.18%	656.70	8.25	100.00%	656.70	23.13	100.00%
	Avg	518.55	522.79	4.69	99.30%	519.47	9.18	99.78%	518.55	37.40	100.00%
20 Customers	1	968.07	971.85	3.46	99.61%	971.85	6.00	99.61%	971.85	18.85	99.61%
	2	633.24	646.99	6.20	97.87%	646.99	10.73	97.87%	646.99	34.98	97.87%
	3	609.21	611.82	7.65	99.57%	609.30	14.58	99.98%	609.30	41.09	99.98%
	4	931.65	931.65	2.25	100.00%	931.65	4.69	100.00%	931.65	12.25	100.00%
	5	864.48	934.74	5.54	92.48%	865.43	18.11	99.89%	881.94	27.88	98.02%
	6	867.01	872.51	4.43	99.37%	872.51	10.46	99.37%	867.01	42.65	100.00%
	7	826.32	833.55	7.54	99.13%	833.55	13.37	99.13%	826.32	43.73	100.00%
	8	765.06	772.92	11.38	98.98%	783.06	12.82	97.70%	783.06	28.47	97.70%
	9	710.30	710.30	7.09	100.00%	710.30	12.38	100.00%	710.30	35.58	100.00%
	10	1022.99	1045.68	4.29	97.83%	1045.68	7.30	97.83%	1022.99	35.47	100.00%
	Avg	819.83	833.20	5.98	98.40%	827.03	11.04	99.13%	825.14	32.10	99.36%
25 Customers	1	535.32	543.95	35.88	98.41%	539.34	65.75	99.25%	538.29	177.39	99.45%
	2	1119.12	1154.00	15.47	96.98%	1119.12	26.58	100.00%	1119.12	87.98	100.00%
	3	1103.00	1177.75	8.85	93.65%	1117.34	28.32	98.72%	1103.00	103.49	100.00%
	4	914.23	939.89	15.77	97.27%	915.11	28.19	99.90%	914.23	88.01	100.00%
	5	982.89	982.89	10.72	100.00%	982.89	27.91	100.00%	982.89	45.75	100.00%
	6	1239.41	1254.37	9.17	98.81%	1254.37	14.98	98.81%	1254.37	33.51	98.81%
	7	947.35	965.85	8.34	98.08%	965.85	12.35	98.08%	965.85	85.61	98.08%
	8	929.17	1000.44	12.65	92.88%	929.17	51.70	100.00%	929.17	121.44	100.00%
	9	1200.62	1231.41	6.73	97.50%	1231.41	19.19	97.50%	1231.41	61.14	97.50%
	10	975.48	1038.21	5.05	93.96%	975.48	18.22	100.00%	975.48	43.18	100.00%
	Avg	994.66	1028.87	12.86	96.75%	1003.01	29.32	99.23%	1001.38	84.75	99.33%

Table 5.1 shows that with the increase of iterations, the number of optimal solutions and the percentage of costs solved to optimality increase. However, the influence of iteration numbers on total cost is less significant for small size examples. In Table 5.1, all parameter sets lead to the same optimality percentage in all 10-customer examples, while in other examples, the (50, 400) parameter set performs better than other two sets, leading to 10 optimal solutions in 15-customer examples, 5 over 10 optimal solutions in 20-

customer examples and 6 over 10 optimal solutions in 25-customer examples. However, more iterations do not guarantee a better solution.

In all the examples with 10 to 25 customers, the parameter combination of (50, 400) achieves an average of more than 99% optimality. Higher values of iteration numbers are not tested due to limited improvement in the solution and the increase in computation time especially for large-scale examples. Thus, we set the number of non-improved consecutive iterations for transfer and swap neighbourhood heuristics to be 50 and the total number of non-improved consecutive iterations to be 400 in all the following examples.

Given the small scale of our examples, we select the length of tabu list from the set of (4, 6, 8). We test 10 examples in Table 5.2 to determine the tabu size and verify the rationality of these parameters. The table illustrates that tabu size 8 generates the lowest means for these ten examples, while tabu size of 4 has the lowest standard deviations.

Table 5.2 Comparison of different Tabu Size

Tabu size	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6	Example 7	Example 8	Example 9	Example 10	Mean	Standard Deviation
4	543.95	1119.12	1137.28	914.23	982.89	1254.37	965.85	957.60	1231.41	975.48	1008.22	192.30
6	538.29	1119.12	1117.34	914.23	982.89	1254.37	965.85	929.17	1231.41	975.48	1002.81	193.32
8	538.29	1119.12	1103.00	914.23	982.89	1254.37	965.85	929.17	1231.41	975.48	1001.38	192.52
optimal solution	535.32	1119.12	1103.00	914.23	982.89	1239.41	947.35	929.17	1200.62	975.48	994.66	188.16

According to the above results, we adopt (50, 400, 8) as the parameter set so as to find close to optimal solution as well as to control the computation time.

5.2.2 Simulated Annealing parameters

In this research, the starting temperature of the temperature schedule is designed to be the objective function value of the initial solution multiplies 0.2, and the cooling parameter is 0.95, where

$T_1 = 0.2Z_c$, Z_c is the objective function value of the initial solution

$T_2 = 0.95T_1$,

$T_n = 0.95T_{n-1}$.

Other simulated annealing parameters to be determined involve the number of temperatures and the number of iterations on each temperature. Besides, the randomness of simulated annealing algorithms requires several runs to eliminate the effect of the algorithm randomness on the result.

The simulated annealing algorithm randomly selects one neighbor of the current trial solution in each iteration. This randomness process often results in different solutions when running an example using the same parameters. The more replications we run, the more accurate it is to reflect the performance in obtaining optimal value of this algorithm. However, more replications require the increase in workload and computation time. Thus, we need to determine the number of replications for our problems.

In this thesis, we test 1, 3, 5 replications on five 25-customer examples with 20 temperatures and 50 iterations on each temperature. The average cost of running 1, 3, and 5 replications of these five examples can be found in Appendix B. Figure 5.2 summarizes the mean and standard deviation of running 1, 3, 5 replications of five examples. According to Figure 5.2, the average cost running different numbers of replications are similar. Running 1 replication generates the highest standard deviation, while 3 and 5 replications lead to similar standard deviations. Thus, 3 replications are conducted in all the examples using simulated annealing in this research.

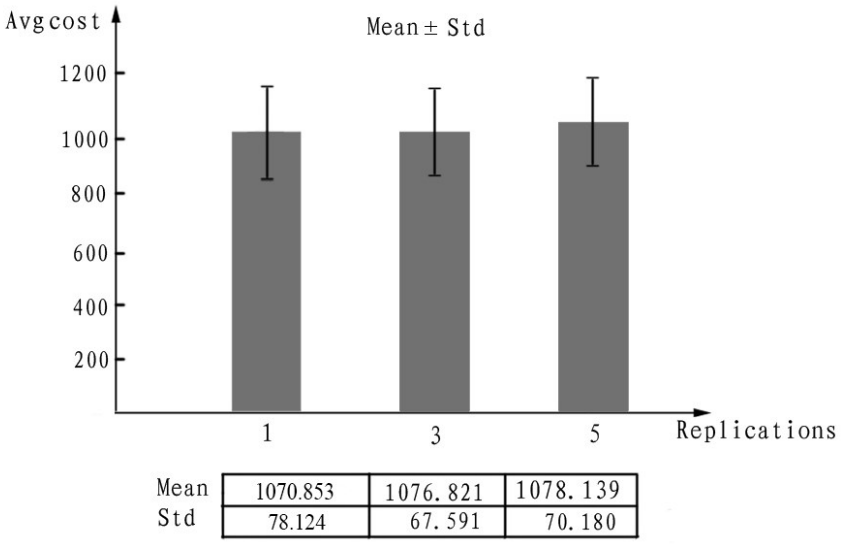


Figure 5.2 One standard of the Mean for 1, 3, 5 replications

Another two parameters to be determined are the number of temperatures and the number of iterations on each temperature. Similar to the process of selecting parameters in Tabu Search, here we compare three different parameter combinations: (20, 20), (50, 50) and (50, 100) in Table 5.3.

Table 5.3 Comparison of Simulated Annealing parameter combinations

			SA								
		Param	20, 20			50, 50			50, 100		
	Examples	Optimal	Cost	Time	optimality	Cost	Time	optimality	Cost	Time	optimality
10 Customers	1	288.30	288.30	2.31	100.00%	288.30	17.54	100.00%	288.30	34.271	100.00%
	2	384.24	384.24	2.53	100.00%	384.24	13.33	100.00%	384.24	26.99	100.00%
	3	440.43	440.43	2.12	100.00%	440.43	11.26	100.00%	440.43	22.39	100.00%
	4	509.11	509.11	2.10	100.00%	509.11	10.11	100.00%	509.11	19.88	100.00%
	5	453.49	453.49	4.51	100.00%	453.49	23.17	100.00%	453.49	48.35	100.00%
	6	330.75	330.75	2.65	100.00%	330.75	17.77	100.00%	330.75	31.32	100.00%
	7	665.74	665.74	0.88	100.00%	665.74	1.14	100.00%	665.74	1.67	100.00%
	8	432.51	432.51	2.79	100.00%	432.51	14.44	100.00%	432.51	28.33	100.00%
	9	381.31	381.31	2.12	100.00%	381.31	12.05	100.00%	381.31	24.76	100.00%
	10	533.63	533.63	1.88	100.00%	533.63	15.03	100.00%	533.63	27.49	100.00%
	Avg	441.95	441.95	2.39	100.00%	441.95	13.58	100.00%	441.95	26.55	100.00%
15 Customers	1	333.27	333.27	4.63	100.00%	333.27	23.74	100.00%	333.27	46.44	100.00%
	2	405.78	405.78	4.31	100.00%	405.78	24.98	100.00%	405.78	58.26	100.00%
	3	389.07	389.07	5.83	100.00%	389.07	22.52	100.00%	389.07	42.18	100.00%
	4	444.52	444.52	5.18	100.00%	444.52	18.70	100.00%	444.52	37.85	100.00%
	5	381.53	381.53	5.03	100.00%	381.53	18.84	100.00%	381.53	36.98	100.00%
	6	410.98	410.98	3.13	100.00%	410.98	13.45	100.00%	410.98	25.38	100.00%

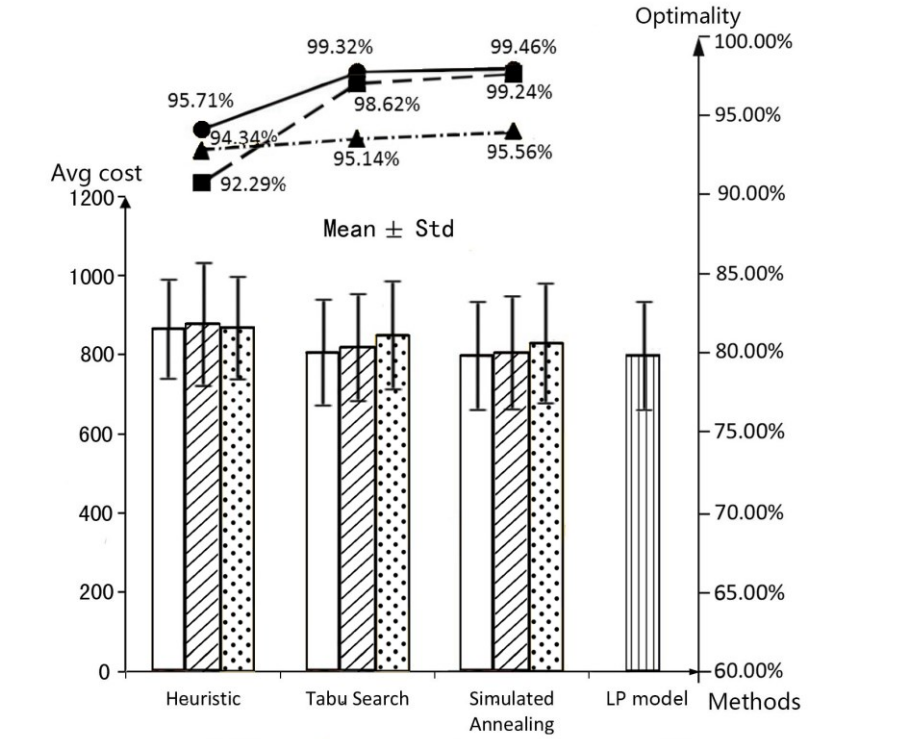
Table 5.3 Comparison of Simulated Annealing parameter combinations (Continued)

15 Customers	7	571.12	571.12	3.20	100.00%	571.12	14.92	100.00%	571.12	30.60	100.00%
	8	734.26	734.26	3.20	100.00%	734.26	12.15	100.00%	734.26	22.64	100.00%
	9	858.24	858.24	3.47	100.00%	858.24	41.68	100.00%	858.24	81.87	100.00%
	10	656.70	688.97	3.46	95.32%	656.70	10.61	100.00%	656.70	19.09	100.00%
	Avg	518.55	521.78	4.14	99.38%	518.55	20.16	100.00%	518.55	40.13	100.00%
20 Customers	1	968.07	971.85	4.85	99.61%	971.85	19.20	99.61%	971.85	39.69	99.61%
	2	633.24	644.42	5.30	98.26%	639.64	19.93	99.00%	633.24	39.41	100.00%
	3	609.21	623.69	7.19	97.68%	615.65	19.60	98.95%	613.54	42.50	99.29%
	4	931.65	931.65	3.75	100.00%	931.65	17.31	100.00%	931.65	34.45	100.00%
	5	864.48	910.90	3.43	94.90%	880.47	12.49	98.18%	864.48	23.64	100.00%
	6	867.01	875.42	5.29	99.04%	867.01	17.31	100.00%	867.01	34.60	100.00%
	7	826.32	861.68	4.96	95.90%	841.86	17.29	98.15%	830.54	33.30	99.49%
	8	765.06	780.89	5.64	97.97%	784.27	20.91	97.55%	783.06	41.10	97.70%
	9	710.30	713.38	4.91	99.57%	711.83	18.64	99.79%	712.13	36.35	99.74%
	10	1022.99	1029.8	4.54	99.33%	1033.4	18.36	98.99%	1025.1	32.54	99.79%
	Avg	819.83	834.38	4.98	98.23%	827.76	18.10	99.02%	823.27	35.76	99.56%
25 Customers	1	535.32	543.95	18.23	98.41%	541.58	47.31	98.84%	539.43	107.17	99.24%
	2	1119.12	1154.0	7.62	96.98%	1143.4	26.62	97.87%	1130.8	50.25	98.96%
	3	1103.00	1180.7	6.16	93.41%	1146.7	22.46	96.18%	1121.6	42.34	98.34%
	4	914.23	1030.6	8.69	88.70%	982.48	24.28	93.05%	968.00	26.14	94.45%
	5	982.89	1036.1	6.77	94.86%	1010.8	19.96	97.23%	1003.8	38.74	97.91%
	6	1239.41	1254.7	4.86	98.78%	1254.6	17.75	98.79%	1254.7	34.69	98.78%
	7	947.35	965.85	8.79	98.08%	962.68	30.27	98.41%	964.00	57.31	98.27%
	8	929.17	1000.4	8.09	92.88%	944.74	30.08	98.35%	940.44	56.82	98.80%
	9	1200.62	1231.4	5.61	97.50%	1231.4	22.47	97.50%	1231.4	45.05	97.50%
	10	975.48	1031.0	5.51	94.61%	984.84	18.64	99.05%	984.10	35.87	99.12%
	Avg	994.66	1042.9	8.03	95.42%	1020.3	25.98	97.53%	1013.8	49.44	98.14%

Table 5.3 shows that all these three parameter sets accomplish 100% optimality on 10-customer examples in simulated annealing metaheuristic. However, in larger scale problems, as expected, parameter set (50, 100) performs better than other parameter sets and results in the highest optimality percentage among all parameter sets. For the 10 examples with 20 customers in Table 5.3, the parameter set (20, 20), (50, 50) and (50, 100) of SA obtain 1, 2, 4 optimal solutions over 10 problems, respectively. And the average optimality percentages for these 10 problems are 98.23%, 99.02%, and 99.56%, separately. Even if none of these parameter sets reaches the optimum in some examples, the parameter set with more iterations usually has more chance of generating a better solution. Again, higher values are not tested here considering computational efficiency and limited improvement. Thus, we set the number of temperatures to be 50, and the number of iterations in each Temperature to be 100.

5.3 Comparison between two neighborhood heuristics

In the methodology section, we proposed two neighborhood heuristics (transfer and swap to generate immediate neighbors) in heuristic and metaheuristic algorithms. The combination of transfer and swap neighbourhood heuristic is rarely applied in searching for neighbors in the literature. Thus, before applying these two neighborhood heuristics, we first validate the rationality of applying this combination. In this part, comparisons on the employment of transfer heuristic only, swap heuristic only, and the combination of them in 10 trials with 20 customers in heuristic, tabu search and simulated annealing algorithms are given separately. The parameter combination derived from the previous section is applied. The results of total cost, computation time, and optimality of each trial using different neighbourhood structure in different algorithms can be found in Appendix C. Figure 5.3 summarizes the mean, standard deviation and average optimality of these 10 trials.



	Mean	Std	Mean	Std	Mean	Std	Mean	Std
transfer&swap	856.39	131.41	825.14	130.49	823.27	131.27	819.83	131.98
transfer only	888.28	150.44	830.97	132.78	826.52	135.35		
swap only	869.05	123.91	859.65	122.75	857.03	128.65		

Figure 5.3 Mean, Std, and Optimality percentage of different neighborhood structures by algorithms

Looking at the means from Figure 5.3, the combination of transfer and swap heuristics behaves the best in all three algorithms, achieving an average of 95.71% optimality in heuristic algorithm and more than 99% optimality in tabu search and simulated annealing algorithms. In heuristic algorithm, swap heuristic behaves a little better than transfer heuristic with a slight gap. In tabu search and simulated annealing algorithms, transfer heuristic performs better than swap heuristic with larger differences. Overall, transfer heuristic is more effective in leading to a better solution according to the means in tables.

Figure 5.3 also plots the one standard deviation away from the mean. According to the figure, the standard deviations of applying different neighbourhood structures are very close and are relatively small compared to mean. Swap heuristic generates the lowest standard deviations in all these three algorithms, transfer heuristic leads to the highest standard deviation, and the combination of swap and transfer is way in between.

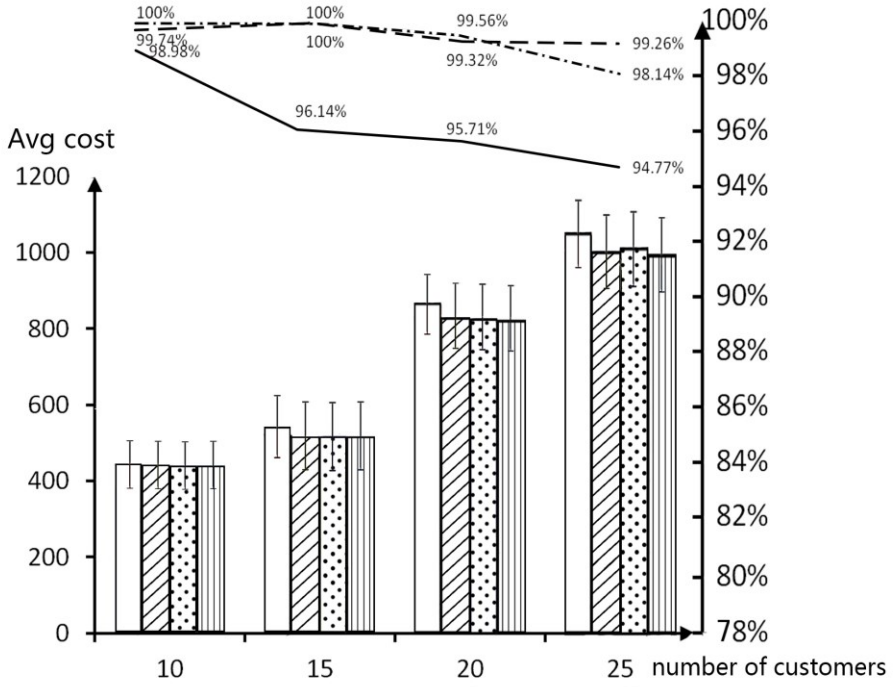
According to the analysis of means and standard deviations, the combination of transfer and swap neighborhood heuristic is the most effective among these three neighborhood structures, which proves the appropriate selection of the neighborhood structure in this research.

5.4 Comparison among different methods

In this section, we apply all the methods employed in this research on different scales of problems from 10 to 25 customers and compare their performance on total cost and computational time. GLPK and Gurobi are both applied in testing the MILP model. The GLPK package is intended for solving linear programming, mixed integer programming and other related problems with poor performance in relatively large or complex problems, while Gurobi optimizer is a commercial optimization solver for math programming with outstanding solver performance. The same parameters determined before are applied since the more iterations can explore lager areas and therefore increasing the chance of reaching better solutions. In tabu search algorithm, tabu size of 8 is applied to different scales of examples since these examples are all in small scale. The transfer heuristic and swap heuristic are applied together as the neighborhood structure in heuristic and metaheuristic algorithms. 10 examples are tested in each scale of problem.

The cost, computation time, and optimality percentage of applying different algorithms in each trial can be found in Appendix D.

The comparisons on average cost, standard deviation, and optimality percentage of different approaches in different scales of examples are demonstrated in Figure 5.4. The number of optimal solutions and the computation time of each approach in different scale of examples are shown in Table 5.4 and 5.5, respectively.



Mean	Std	Mean	Std	Mean	Std	Mean	Std
445.54	100.00	543.08	186.17	856.39	131.41	1051.32	197.88
443.14	103.15	518.55	168.35	825.14	130.49	1002.81	193.32
441.95	103.06	518.55	168.35	823.27	131.27	1013.84	191.49
441.95	103.06	518.55	168.35	819.83	131.98	994.66	188.16

Figure 5.4 Mean, Std, and Optimality percentage of four methods by different problem scales

Table 5.4 Number of optimal solutions of four methods

	Number of optimal solutions			
Number of customers	MILP model	Tabu Search	Simulated Annealing	Heuristic
10 customers	10	8	10	6
15 customers	10	10	10	2
20 customers	10	5	4	0
25 customers	10	5	0	0

Table 5.5 Average computation time of four methods

Computation time (s)	MILP on GLPK	MILP on Gurobi	Tabu Search	Simulated Annealing	Heuristic
10 customers	0.19	0.47	4.53	20.49	0.18
15 customers	36.21	1.56	29.25	39.42	0.64
20 customers	2212.57	4.76	37.54	37.42	0.81
25 customers	11912.26	34.48	77.78	46.76	1.42

According to the figure and tables, MILP model has the best performance on both cost and computation time in 10-customer examples using GLPK and Gurobi. The average computation time on these 10 examples using GLPK is only 0.19 seconds and 0.47 on Gurobi. Simulated annealing works the best among heuristic and metaheuristic algorithms on total cost, arriving at optimum in these 10 examples with the average computation time of 20.49 seconds. Tabu search algorithm generates 8 optimum solutions over these 10 examples with optimality of 99.74% and average computation time of 4.53. The performance of tabu search on total cost is highly close to optimum. Heuristic algorithm achieves 6 optimal solutions with the minimum average computation time of 0.18 seconds among all these methods.

For 15-customer problems, MILP model based on GLPK performs still well on computation time with an average of 36.21 seconds, while Gurobi only takes 1.56 seconds on average. Only two examples using the heuristic algorithm generate optimal solutions. The average optimality percentage is 96.14% with time of 0.64 seconds. Both tabu search and simulated annealing algorithms achieve optimum in these 10 examples with the average time of 29.25 and 39.42 seconds, respectively.

For 20-customer problems, the GLPK solver becomes time consuming and the average computation time is 2212.57 seconds, while Gurobi only takes 4.76 seconds averagely. Heuristic algorithm spends only 0.81 seconds but can only produce non-optimal solutions with 95.71% optimality averagely. Tabu search algorithm provides 5 optimal solutions with an average of 99.32% optimality and 37.54 seconds on computation time. Similarly, simulated annealing generates 4 optimal solutions with 99.56% optimality and 37.42 seconds computation time.

As for 25-customer problems, the GLPK solver is extremely time consuming with the average computation time of 11912.26 seconds, while Gurobi takes 34.48 seconds. The actual average computation time might be even longer since the memory is not enough after several hours' computation in some attempts. Heuristic algorithm spends 1.42 seconds and reaches 94.77% of optimality on average. Tabu search method generates 5 optimal solutions over these 10 problems with 77.78 seconds of computation time on average. The optimality percentage is 99.26%. Simulated annealing spends 46.76 seconds averagely. Since the result in each example is the average of three replications, there is no optimal solution. It arrives at 98.14% of optimality.

In summary, MILP model tested on Gurobi generates optimal solutions within one minute in all the examples as expected. The average computation time spent using Gurobi is only a little more than the heuristic algorithm in all scales of problems. GLPK performs well for problems with 10 and 15 customers. However, it cannot solve 20 and 25 customer problems within reasonable computation time. With the increase of customer quantities, the difference in computation time between these two solvers becomes large. Heuristic method has the worst performance on average cost but spends the least time among all the algorithms. This is reasonable since the heuristic method applied in this thesis is a local search process and sometimes can only reach a local optimum. In this thesis, the result generated from this heuristic method is applied as the initial solution in tabu search and simulated annealing algorithms. Looking at the means of the objective values in Figure 5.4, simulated annealing and tabu search all perform very well and the results generated by these two algorithms are very close. Simulated annealing seems to perform better than tabu search in small-scale problems with 10 to 20 customers, while

tabu search behaves a little better on 25-customer examples according to the means of cost. However, the difference is very limited. Performance on computation time is also similar for these two methods. Besides, tabu search and simulated annealing algorithms can both achieve more than 98% of optimality compared to the exact algorithm with much less computation time, usually less than one minute. Thus, our methods of tabu search and simulated annealing based on transfer and swap neighborhood structures prove to be very effective as well as efficient. According to Figure 5.4, the standard deviations of applying different methods are very close and are relatively small compared to mean in all scales of problems.

The heuristic algorithm proposed in this research is very efficient on small-scale examples according to the optimality percentage and computation time, which validates the appropriate selection of initialization and improvement algorithms.

Trend: According to Figure 5.4, with the increase of customer quantities, the difference among heuristic algorithm and the two metaheuristic algorithms on optimality percentage also increases. The optimality percentages of metaheuristic algorithms tested on different problem scales decreases slowly and the trend is very stable, while the optimality percentage of heuristic tested on different problem scales decreases dramatically. Thus, metaheuristic algorithms are worthwhile to be developed because they can lead to close to optimal solutions in large-scale examples according to the trend.

After testing problems with 10 to 25 customers, we also apply Gurobi to solve 50-customer problems. However, the computer is out of memory and cannot reach a solution. Thus, we will deal with large-scale problems only using heuristic and metaheuristic algorithms.

5.5 Tests on Large-Scale Examples

Finally, we apply heuristic, tabu search and simulated annealing methods on examples with 100, 150, and 200 customers to further demonstrate the performance of these methods. For tabu search and simulated annealing algorithms, different computation time (1000, 2000, and 3000 seconds) are implemented to further compare these two methods.

In tabu search algorithm, we test 8, 20, 30 as tabu size and the results of tabu search using different tabu sizes in examples with 100 and 150 customers are given in Table 5.6. The 200-customer example is not tested using different tabu size since it is very time consuming. From the table, the difference in cost using different tabu size is not large. Thus, we decide to still use 8 as tabu size.

Table 5.6 Results of tabu search with different tabu size in 100- and 150-customer examples

	tabu size 8		tabu size 20		tabu size 30	
Number of customers	Cost	Time	Cost	Time	Cost	Time
100	3769.12	4160.35	3771.54	3607.46	3752.12	2622.8
150	5732.36	3167.63	5706.21	4537.53	5737.62	3127.52

The results of heuristic and metaheuristic algorithms on large-scale problems are shown in Table 5.7. Heuristic algorithm is very efficient in computation time and it leads to a cost similar to that generated by simulated annealing algorithm. Tabu search leads to improvements by running for more computation time in each problem, while increasing the computation time do not improve the result of simulated annealing notably. Among these two metaheuristics, tabu search performs better in the cost under the same computation time.

Table 5.7 Comparison among heuristic and metaheuristic algorithms on large-scale examples

	Heuristic		Tabu Search		Simulated Annealing	
Number of customers	Cost	Time	Cost	Time	Cost	Time
100	3905.90	65.57	3905.90	1000	3905.90	1000
			3784.38	2000	3871.31	2000
			3787.81	3000	3874.61	3000
150	6088.95	184.55	5999.10	1000	6088.95	1000
			5990.20	2000	6088.95	2000
			5968.43	3000	6088.95	3000
200	7729.83	466.23	7657.31	1000	7729.83	1000
			7656.21	2000	7729.83	2000
			7628.32	3000	7729.83	3000

5.6 Conclusion and Future Research

Based on existing research on vehicle routing problem with time windows and vehicle routing problem with skill sets, we developed four methods to solve this VRPTWSS. To the best of our knowledge, this is the first time the VRPTWSS is studied thoroughly in literature. The goal in this problem is to allocate all customers to eligible technicians considering time window and skill set constraints in a short planning time and reduce total cost.

The MILP model is built up to solve this problem optimally and tested on Gusek and Gurobi solvers. However, the MILP model is not capable of solving large-scale problems. Heuristic and metaheuristic algorithms are developed to speed up computations. Heuristic algorithm aims at reaching a local optimum, while tabu search and simulated annealing algorithms are applied to jump out of the local optimum and obtain a close to optimum solution. The combination of transfer and swap neighbourhood structure is applied in heuristic and metaheuristic algorithms to search for neighbors. Another main contribution in this research is our development of the intra-route improvement heuristics in which the sub-tour reversal method is applied to reduce the travel time and the waiting time improvement method is applied to reduce the waiting time. All data used in this thesis are generated by a simulation process.

The performance of transfer only, swap only, and the combination of these two neighborhood heuristics is examined. The combination of two heuristics behave the best in total cost. Thus, transfer and swap heuristics are applied together as the neighborhood structure in heuristic and metaheuristic algorithms.

All the approaches are examined on small-scale examples with 10 to 25 customers since neither GLPK nor Gurobi can solve larger problems. Gurobi can yield optimal solutions for these small-scale problems very fast, while GLPK takes much more computation time in solving examples with 20 and 25 customers. Looking at means, tabu search and simulated annealing algorithms have similar performance in these small-scale examples and the difference in cost between MILP model and these two metaheuristics is reasonably small. Heuristic algorithm spends the least time in approximation algorithms while behaves the worst in improving the objective value.

Heuristic and metaheuristic algorithms are tested on large-scale examples as well. In these examples, tabu search algorithm produces the best results, while being less efficient in computation time. Heuristic and simulated annealing have similar performance on the objective value, while heuristic is more time saving.

5.6.1 Future Research

- Intra-route improvement: In this research, we apply sub-tour reversal algorithm to change the sequence of a route and thereby improving the route internally. Since the time windows of customers are not overlapping in our problem, we only exchange customers with the same time windows. Future research can put efforts in exchanges between customers with overlapping time windows. Besides, MILP model can be applied to explore the optimal sequence of each route.
- Neighborhood Structure: In this research, we apply transfer and swap heuristics as our neighborhood structure. Future research can apply other neighborhood heuristics in the literature into this VRPTWSS.
- Metaheuristics: Future research can investigate genetic algorithm and evaluate its performance in solving this particular problem.
- Size of tabu list: In this thesis, we test several tabu sizes and apply 8 to all our examples. In future research, the tabu size can be studied further and different number of tabu sizes can be applied based on the scale of problems.
- Practical application: In this thesis, we apply four methods on several simulated data set. In future research, realistic data might be collected and tested to further evaluate our methods.
- Tests on large-scale problems: The algorithms developed in this research can be evaluated on large-scale problems by implementing the MILP model and the two metaheuristic algorithms for the same computation time and comparing the results of different algorithms.

Bibliography

[1] Golden, B. L., Raghavan, S., & Wasil, E. A. (Eds.). (2008). The vehicle routing problem: latest advances and new challenges (Vol. 43). Springer Science & Business Media.

[2] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), 80-91.

[3] Clarke, G. U., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12(4), 568-581.

[4] Ropke, S. (2005). Heuristic and exact algorithms for vehicle routing problems.

[5] Unpublished PhD thesis, Computer Science Department, University of Copenhagen.

[6] Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. In The vehicle routing problem: latest advances and new challenges (pp. 3-27). Springer US.

[7] Campbell, A., Clarke, L., Kleywegt, A., & Savelsbergh, M. (1998). The inventory routing problem. In Fleet management and logistics (pp. 95-113). Springer US.

[8] Dror, M., & Trudeau, P. (1990). Split delivery routing. Naval Research Logistics (NRL), 37(3), 383-402

[9] Lin, C., Choy, K. L., Ho, G. T., Chung, S. H., & Lam, H. Y. (2014). Survey of green vehicle routing problem: past and future trends. Expert Systems with Applications, 41(4), 1118-1138.

[10] Pillac, V., Gueret, C., & Medaglia, A. L. (2013). A parallel matheuristic for the technician routing and scheduling problem. Optimization Letters, 7(7), 1525-1535.

[11] Lenstra, J. K., & Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. Networks, 11(2), 221-227.

[12] Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(3), 345-358.

[13] Desrochers, M., Lenstra, J. K., & Savelsbergh, M. W. (1990). A classification scheme for vehicle routing and scheduling problems. European Journal of Operational Research, 46(3), 322-332.

[14] Desrochers, M., Jones, C. V., Lenstra, J. K., Savelsbergh, M. W., & Stougie, L. (1999). Towards a model and algorithm management system for vehicle routing and scheduling problems. Decision Support Systems, 25(2), 109-133.

[15] Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering, 57(4), 1472-1483.

[16] Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. North-Holland Mathematics Studies, 132, 147-184.

[17] Magnanti, T. L. (1981). Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. Networks, 11(2), 179-213.

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

[19] Laporte, G., Mercure, H., & Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. Networks, 16(1), 33-46.

[20] Kumar, Y., & Jain, S. (2015, September). School bus routing based on branch and bound approach. In Computer, Communication and Control (IC4), 2015 International Conference on (pp. 1-4). IEEE.

[21] Eilon, S., Watson-Gandy, C. D. T., Christofides, N., & de Neufville, R. (1974). Distribution Management-Mathematical Modelling and Practical Analysis. IEEE Transactions on Systems, Man, and Cybernetics, (6), 589-589.

[22] Christofides, N., Mingozzi, A., & Toth, P. (1981). State-space relaxation procedures for the computation of bounds to routing problems. Networks, 11(2), 145-164.

[23] Desrosiers, J., Dumas, Y., & Soumis, F. (1984). A Dynamic Programming Method for the Large Scale Single Vehicle Dial-a-ride Problem with Time Windows. Cahier du GERAD, 84, 12.

[24] Toth, P., & Vigo, D. (Eds.). (2014). Vehicle routing: problems, methods, and applications (Vol. 18). Siam.

[25] Balinski, M. L., & Quandt, R. E. (1964). On an integer program for a delivery problem. Operations Research, 12(2), 300-304.

[26] Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. Journal of the Operational Research Society, 27(2), 367-384.

[27] Agarwal, Y., Mathur, K., & Salkin, H. M. (1989). A set-partitioning-based exact algorithm for the vehicle routing problem. Networks, 19(7), 731-749.

[28] Naddef, D., & Rinaldi, G. (2001, January). Branch-and-cut algorithms for the capacitated VRP. In The vehicle routing problem (pp. 53-84). Society for Industrial and Applied Mathematics.

[29] Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. Networks, 11(2), 109-124.

[30] Baldacci, R., Hadjiconstantinou, E., & 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.

[31] Laporte, G., Ropke, S., & Vidal, T. (2014). Heuristics for the vehicle routing problem. Vehicle Routing: Problems, Methods, and Applications, 18, 87.

[32] Laporte, G., Gendreau, M., Potvin, J. Y., & Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research, 7(4-5), 285-300.

[33] Laporte, G. (2007). What you should know about the vehicle routing problem. Naval Research Logistics (NRL), 54(8), 811-819.

[34] Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In Logistics systems: design and optimization (pp. 279-297). Springer US.

[35] Ioannou, G., Kritikos, M., & Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. Journal of the Operational Research Society, 52(5), 523-537.

[36] Lu, Q., & Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. European Journal of Operational Research, 175(2), 672-687.

[37] Nelson, M. D., Nygard, K. E., Griffin, J. H., & Shreve, W. E. (1985). Implementation techniques for the vehicle routing problem. Computers & Operations Research, 12(3), 273-283.

[38] Paessens, H. (1988). The savings algorithm for the vehicle routing problem. European Journal of Operational Research, 34(3), 336-344.

[39] Altınel, İ. K., & Öncan, T. (2005). A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. Journal of the Operational Research Society, 56(8), 954-961.

[40] Laporte, G., & Semet, F. (2001, January). Classical heuristics for the capacitated VRP. In The vehicle routing problem (pp. 109-128). Society for Industrial and Applied Mathematics.

[41] Liu, F. H., & Shen, S. Y. (1999). A method for vehicle routing problem with multiple vehicle types and time windows. Proceedings of Natural Science Council, 23.

[42] Gronalt, M., Hartl, R. F., & Reimann, M. (2003). New savings based algorithms for time constrained pickup and delivery of full truckloads. European Journal of Operational Research, 151(3), 520-535.

[43] Wren, A., & Carr, J. D. (1971). Computers in transport planning and operation.

[44] Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. Operations Research, 22(2), 340-349.

[45] Lin, S. (1965). Computer solutions of the traveling salesman problem. The Bell System Technical Journal, 44(10), 2245-2269.

[46] Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations Research, 21(2), 498-516.

[47] Or, I. (1976). Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Xerox University Microfilms.

[48] Renaud, J., Boctor, F. F., & Laporte, G. (1996). A fast composite heuristic for the symmetric traveling salesman problem. INFORMS Journal on Computing, 8(2), 134-143.

[49] Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. Local Search in Combinatorial Optimization, 1, 215-310.

[50] Van Breedam, A. (1994). An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints. RUCA.

[51] Thompson, P. M., & Psaraftis, H. N. (1993). Cyclic transfer algorithm for multivehicle routing and scheduling problems. Operations Research, 41(5), 935-946.

[52] Wassan, N. A. (2006). A reactive tabu search for the vehicle routing problem. Journal of the Operational Research Society, 57(1), 111-116.

[53] Derigs, U., & Kaiser, R. (2007). Applying the attribute based hill climber heuristic to the vehicle routing problem. European Journal of Operational Research, 177(2), 719-732.

[54] Lai, D. S., Demirag, O. C., & Leung, J. M. (2016). A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. Transportation Research Part E: Logistics and Transportation Review, 86, 32-52.

[55] Zeng, L., Ong, H. L., & Ng, K. M. (2005). An assignment-based local search method for solving vehicle routing problems. Asia-Pacific Journal of Operational Research, 22(01), 85-104.

[56] Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research, 41(4), 421-451.

[57] Baker, B. M., & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. Computers & Operations Research, 30(5), 787-800.

[58] Berger, J., & Barkaoui, M. (2003). A new hybrid genetic algorithm for the capacitated vehicle routing problem. Journal of the Operational Research Society, 54(12), 1254-1262.

[59] Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31(12), 1985-2002.

[60] Bell, J. E., & McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. Advanced Engineering Informatics, 18(1), 41-48.

[61] Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research, 31(4), 563-591.

[62] Doerner, K. F., Hartl, R. F., & Lucka, M. (2005). A parallel version of the d-ant algorithm for the vehicle routing problem. Parallel Numerics, 5, 109-118.

[63] Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(2), 231-247.

[64] Hoffman, K. L., Padberg, M., & Rinaldi, G. (2013). Traveling salesman problem. In Encyclopedia of Operations Research and Management Science (pp. 1573-1578). Springer US.

[65] Letchford, A. N., Nasiri, S. D., & Theis, D. O. (2013). Compact formulations of the Steiner traveling salesman problem and related problems. European Journal of Operational Research, 228(1), 83-92.

[66] Lawler, E. L., Lenstra, J. K., Rinnooy-Kan, A. G., & Shmoys, D. B. (1985). Traveling salesman problem.

[67] Gutin, G., & Punnen, A. P. (Eds.). (2006). The traveling salesman problem and its variations (Vol. 12). Springer Science & Business Media.Johnson, D. S.

[68] Potvin, J. Y. (1996). Genetic algorithms for the traveling salesman problem. Annals of Operations Research, 63(3), 337-370

[69] Bryant, K., & Benjamin, A. (2000). Genetic algorithms and the traveling salesman problem. Department of Mathematics, Harvey Mudd College, 10-12.

[70] Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. European Journal of Operational Research, 228(1), 72-82.

[71] Aarts, E. H., Korst, J. H., & van Laarhoven, P. J. (1988). A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem. Journal of Statistical Physics, 50(1-2), 187-206.

[72] Wang, Y., Tian, D., & Li, Y. (2013). An improved simulated annealing algorithm for traveling salesman problem. In Proceedings of the 2012 International Conference on Information Technology and Software Engineering (pp. 525-532). Springer Berlin Heidelberg.

[73] Fiechter, C. N. (1994). A parallel tabu search algorithm for large traveling salesman problems. Discrete Applied Mathematics, 51(3), 243-267.

[74] Xu, D., Weise, T., Wu, Y., Lässig, J., & Chiong, R. (2015, September). An investigation of hybrid tabu search for the traveling salesman problem. In Bio-Inspired Computing-Theories and Applications (pp. 523-537). Springer Berlin Heidelberg.

[75] Achuthan, N. R., & Caccetta, L. (1991). Integer linear programming formulation for a vehicle routing problem. European Journal of Operational Research, 52(1), 86-89.

[76] Kara, I., Laporte, G., & Bektas, T. (2004). A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. European Journal of Operational Research, 158(3), 793-795.S

[77] Baldacci, R., Mingozzi, A., & 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.

[78] Semet, F., Toth, P., & Vigo, D. (2014). Classical exact algorithms for the capacitated vehicle routing problem. Vehicle Routing: Problems, Methods, and Applications, 18, 37.

[79] Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., & Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem. Rapport de recherche- IMAG.

[80] Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming, 100(2), 423-445.

[81] Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., & Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical programming, 106(3), 491-511.

[82] Baldacci, R., Christofides, N., & Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming, 115(2), 351-385.

[83] Toth, P., & Vigo, D. (2001, January). An overview of vehicle routing problems. In The vehicle routing problem (pp. 1-26). Society for Industrial and Applied Mathematics.

[84] Gillett, B. E., & Johnson, J. G. (1976). Multi-terminal vehicle-dispatch algorithm. Omega, 4(6), 711-718.

[85] Robust, F., Daganzo, C. F., & Souleyrette, R. R. (1990). Implementing vehicle routing models. Transportation Research Part B: Methodological, 24(4), 263-286.

[86] Tavakkoli-Moghaddam, R., Safaei, N., & Gholipour, Y. (2006). A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length. Applied Mathematics and Computation, 176(2), 445-454.

[87] Xiao, Y., Zhao, Q., Kaku, I., & Mladenovic, N. (2014). Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems. Engineering Optimization, 46(4), 562-579.

[88] Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. Management Science, 40(10), 1276-1290.

[89] Jin, J., Crainic, T. G., & Løkketangen, A. (2012). A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. European Journal of Operational Research, 222(3), 441-451.

[90] Nazif, H., & Lee, L. S. (2012). Optimised crossover genetic algorithm for capacitated vehicle routing problem. Applied Mathematical Modelling, 36(5), 2110-2117.

[91] Lin, S. W., Lee, Z. J., Ying, K. C., & Lee, C. Y. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. Expert Systems with Applications, 36(2), 1505-1512.

[92] Speranza, M. G., & Ukovich, W. (1994). Minimizing transportation and inventory costs for several products on a single link. Operations Research, 42(5), 879-894.

[93] Speranza, M. G., & Ukovich, W. (1996). An algorithm for optimal shipments with given frequencies. Naval Research Logistics (NRL), 43(5), 655-671.

[94] Bertazzi, L., Speranza, M. G., & Ukovich, W. (2000). Exact and heuristic solutions for a shipment problem with given frequencies. Management Science, 46(7), 973-988.

[95] Bertazzi, L., Chan, L. M. A., & Speranza, M. G. (2007). Analysis of practical policies for a single link distribution system. Naval Research Logistics (NRL), 54(5), 497-509.

[96] Archetti, C., Bertazzi, L., Laporte, G., & Speranza, M. G. (2007). A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation Science, 41(3), 382-391.

[97] Bertazzi, L., Paletta, G., & Speranza, M. G. (2002). Deterministic order-up-to level policies in an inventory routing problem. Transportation Science, 36(1), 119-132.

[98] Bertazzi, L., Paletta, G., & Speranza, M. G. (2005). Minimizing the total cost in an integrated vendor—managed inventory system. Journal of heuristics, 11(5-6), 393-419.

[99] Cousineau-Ouimet, K. (2002, November). A tabu search heuristic for the inventory routing problem. In Proceedings of 37th Annual ORSNZ Conference.

[100] Berman, O., & Larson, R. C. (2001). Deliveries in an inventory/routing problem using stochastic dynamic programming. Transportation Science, 35(2), 192-213.

[101] Savelsbergh, M., & Song, J. H. (2007). Inventory routing with continuous moves. Computers & Operations Research, 34(6), 1744-1763.

[102] Tan, C. C. R., & Beasley, J. E. (1984). A heuristic algorithm for the period vehicle routing problem. Omega, 12(5), 497-504.

[103] Russell, R. A., & Gribbin, D. (1991). A multiphase approach to the period routing problem. Networks, 21(7), 747-765.

[104] Chao, I., Golden, B. L., & Wasil, E. (1995). An improved heuristic for the period vehicle routing problem. Networks, 26(1), 25-44.

[105] Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for

periodic and multi-depot vehicle routing problems. Networks, 30(2), 105-119.

[106] Drummond, L. M., Ochi, L. S., & Vianna, D. S. (2001). An asynchronous parallel metaheuristic for the period vehicle routing problem. Future Generation Computer Systems, 17(4), 379-386.

[107] Archetti, C., Savelsbergh, M. W., & Speranza, M. G. (2006). Worst-case analysis for split delivery vehicle routing problems. Transportation Science, 40(2), 226-234.

[108] Lee, C. G., Epelman, M. A., White, C. C., & Bozer, Y. A. (2006). A shortest path approach to the multiple-vehicle routing problem with split pick-ups. Transportation Research Part B: Methodological, 40(4), 265-284.

[109] Jin, M., Liu, K., & Bowden, R. O. (2007). A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. International Journal of Production Economics, 105(1), 228-242.

[110] Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2006). Vehicle routing with time windows and split deliveries. Technical Paper, 851.

[111] Frizzell, P. W., & Giffin, J. W. (1995). The split delivery vehicle scheduling problem with time windows and grid network distances. Computers & Operations Research, 22(6), 655-667.

[112] Archetti, C., Speranza, M. G., & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science, 40(1), 64-73.

[113] Archetti, C., Speranza, M. G., & Savelsbergh, M. W. (2008). An optimization-based heuristic for the split delivery vehicle routing problem. Transportation Science, 42(1), 22-31.

[114] Kallehauge, B., Larsen, J., Madsen, O. B., & Solomon, M. M. (2005). Vehicle routing problem with time windows. In Column generation (pp. 67-98). Springer US.

[115] Kohl, N., & 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.

[116] Desrosiers, J., Soumis, F., & Desrochers, M. (1984). Routing with time windows by column generation. Networks, 14(4), 545-565.

[117] Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. Operations Research, 40(2), 342-354.

[118] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, 35(2), 254-265.

[119] Potvin, J. Y., & Rousseau, J. M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research, 66(3), 331-340

[120] Russell, R. A. (1995). Hybrid heuristics for the vehicle routing problem with time windows. Transportation Science, 29(2), 156-166.

[121] Potvin, J. Y., & Rousseau, J. M. (1995). An exchange heuristic for routing problems with time windows. Journal of the Operational Research Society, 46(12), 1433-1446.

[122] Prosser, P., & Shaw, P. (1996). Study of greedy search with multiple improvement

heuristics for vehicle routing problems.

[123] Kontoravdis, G., & Bard, J. F. (1995). A GRASP for the vehicle routing problem with time windows. ORSA Journal on Computing, 7(1), 10-23.

[124] Bräysy, O. (2002). Fast local searches for the vehicle routing problem with time windows. INFOR: Information Systems and Operational Research, 40(4), 319-330.

[125] Chiang, W. C., & Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. Annals of Operations Research, 63(1), 3-27.

[126] Afifi, S., Dang, D. C., & Moukrim, A. (2013). A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints. In Learning and Intelligent Optimization (pp. 259-265). Springer Berlin Heidelberg.

[127] Potvin, J. Y., Kervahut, T., Garcia, B. L., & Rousseau, J. M. (1996). The vehicle routing problem with time windows part I: tabu search. INFORMS Journal on Computing, 8(2), 158-164.

[128] Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science, 31(2), 170-186.

[129] Chiang, W. C., & Russell, R. A. (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. INFORMS Journal on Computing, 9(4), 417-430.

[130] Badeau, P., Guertin, F., Gendreau, M., Potvin, J. Y., & Taillard, E. (1997). A parallel tabu search heuristic for the vehicle routing problem with time windows. Transportation Research Part C: Emerging Technologies, 5(2), 109-122.

[131] Schulze, J., & Fahle, T. (1999). A parallel algorithm for the vehicle routing problem with time window constraints. Annals of Operations Research, 86, 585-607.

[132] Thangiah, S. R., Nygard, K. E., & Juell, P. L. (1991, February). Gideon: A genetic algorithm system for vehicle routing with time windows. In Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on (Vol. 1, pp. 322-328). IEEE.

[133] Thangiah, S. R., Osman, I. H., & Sun, T. (1994). Hybrid genetic algorithm simulated annealing and tabu search methods for vehicle routing problems with time windows. Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27, 69.

[134] Cappanera, P., Gouveia, L., & Scutellà, M. G. (2011). The skill vehicle routing problem. In Network Optimization (pp. 354-364). Springer Berlin Heidelberg.

[135] Krishnamurti, R., Iranmanesh, E., & Sun, W. (2012). The Vehicle Routing Problem with Skill Sets.

[136] Xu, J., & Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem. Journal of Heuristics, 7(5), 495-509.

[137] Tang, H., Miller-Hooks, E., & Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. Transportation Research Part E: Logistics and Transportation Review, 43(5), 591-609.

[138] Pillac, V., Guéret, C., & Medaglia, A. (2012). On the dynamic technician routing and

scheduling problem.

[139] Yang, C. E. (1989). A dynamic programming algorithm for the travelling repairman problem. ASIA-PACIFIC J. OPER. RES., 6(2), 192-206.

[140] Simchi-Levi, D., & Berman, O. (1991). Minimizing the total flow time of n jobs on a network. IIE TRANSACTIONS, 23(3), 236-244.

[141] Fischetti, M., Laporte, G., & Martello, S. (1993). The delivery man problem and cumulative matroids. Operations Research, 41(6), 1055-1064.

[142] Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., & Sudan, M. (1994, May). The minimum latency problem. In Proceedings of the twenty-sixth annual ACM symposium on Theory of computing (pp. 163-171). ACM.

[143] Chaudhuri, K., Godfrey, B., Rao, S., & Talwar, K. (2003, October). Paths, trees, and minimum latency tours. In Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on (pp. 36-45). IEEE.

[144] Archer, A., & Blasiak, A. (2010, January). Improved approximation algorithms for the minimum latency problem via prize-collecting strolls. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (pp. 429-447). Society for Industrial and Applied Mathematics.

[145] Salehipour, A., Sörensen, K., Goos, P., & Bräysy, O. (2011). Efficient GRASP+ VND and GRASP+ VNS metaheuristics for the traveling repairman problem. 4OR, 9(2), 189-209.

[146] Silva, M. M., Subramanian, A., Vidal, T., & Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. European Journal of Operational Research, 221(3), 513-520.

[147] Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C., & Vansteenwegen, P. (2013). Heuristics for the traveling repairman problem with profits. Computers & Operations Research, 40(7), 1700-1707.

[148] Hillier, F. S. (2012). Introduction to operations research. Tata McGraw-Hill Education.

[149] Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM), 7(4), 326-329.

[150] Danvildanvil (2014), Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=31096324

[151] Cassa, C. A., Iancu, K., Olson, K. L., & Mandl, K. D. (2005). A software tool for creating simulated outbreaks to benchmark surveillance systems. BMC Medical Informatics and Decision Making, 5(1), 1.

[152] Lesaint, D., Voudouris, C., Azarmi, N., Alletson, I., & Laithwaite, B. (2003). Field workforce scheduling. BT Technology Journal, 21(4), 23-26.

[153] Petrakis, I., Hass, C., & Bichler, M. (2012). On the impact of real-time information on field service scheduling. Decision Support Systems, 53(2), 282-293.

Appendix A Solomon's Insertion Heuristic

Solomon's insertion heuristic (1987) provided a method of selecting the new customer to be inserted into a route using two criteria. The first criterion c_1 is applied to calculate the best feasible insertion place in the current route for each unrouted customer u that can be served by the technician. The second criterion c_2 is then applied to select the new inserted customer.

The criterion c_1 considers two factors: the increase in total distance of the current route after the insertion (c_{11}), and the delay of service start time of the customer following the new inserted customer (c_{12}). To be specific, $c_1(i,u,j)$ is calculated as:

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j), \alpha_1 + \alpha_2 = 1;\ \alpha_1, \alpha_2 \geq 0; \quad \text{(A.1)}$$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, u \geq 0; \quad \text{(A.2)}$$

$$c_{12}(i,u,j) = b_{j_u} - b_j, \quad \text{(A.3)}$$

Equation A.2 calculates the increase in total distance of the entire route. Considering the insertion of customer u between nodes i and j, $d_{iu} + d_{uj}$ is the new distance between two nodes i and j after the insertion, and d_{ij} is the old distance between i and j. In A.3, b_{j_u} is the new service start time of customer j after the insertion of customer u and b_j is the previous service start time. The measurement of an insertion place is based on these two factors as shown in equation A.1.

Suppose the current route to be $(i_0, i_1, i_2, \dots, i_m)$, the best feasible insertion place c_1 in the current route for each unrouted customer u is calculated as:

$$c_1(i(u), u, j(u)) = \min\left[c_1\left(i_{p-1}, u, i_p\right)\right], p = 1, \dots, m. \quad \text{(A.4)}$$

Then the second criterion for each customer is calculated as:

$$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j), \lambda \geq 0, \quad \text{(A.5)}$$

Where d_{0u} is the distance between the depot and customer to be inserted. The next customer to be inserted in the current route as the one with the minimal value of c_2:

$$c_2(i^*, u^*, j^*) = \text{optimum}[c_2(i, u, j)] \quad \text{(A.6)}$$

And the insertion place for this new customer u^* is between i^* and j^* with the minimum value of c_1.

Appendix B Average cost of running different numbers of replications in SA algorithm

The following table provides the average cost of running 1, 3, and 5 replications on five examples with 20 customers.

Table 1 Results for different numbers of replications in SA algorithm

	Avg cost of 1 rep	Avg cost of 3 rep	Avg cost of 5 rep
Example 1	1170.75	1167.00	1175.18
Example 2	966.63	1005.14	1004.69
Example 3	1026.40	1028.79	1028.39
Example 4	1038.21	1030.53	1032.06
Example 5	1152.29	1149.47	1150.37

Appendix C Comparisons of different neighborhood structure by algorithms

The following tables provide the total cost, computation time, and optimality of each trial with 20 customers using different neighbourhood structure in different algorithms.

Table 1 Results of different neighbourhood structure in heuristic algorithm

			Heuristic								
			transfer & swap			transfer only			swap only		
	Examples	Optimum	cost	time	optimality	cost	time	optimality	cost	time	optimality%
20 Customers	1	968.07	971.85	0.44	99.61%	1004.82	0.11	96.34%	971.85	0.88	99.61%
	2	633.24	646.99	0.88	97.87%	646.99	0.16	97.87%	700.15	0.34	90.44%
	3	609.21	623.69	1.83	97.68%	639.24	0.32	95.30%	649.13	0.44	93.85%
	4	931.65	961.81	0.15	96.86%	961.81	0.12	96.86%	956.51	0.15	97.40%
	5	864.48	934.74	0.47	92.48%	1010.50	0.15	85.55%	928.44	0.22	93.11%
	6	867.01	875.42	1.14	99.04%	929.19	0.32	93.31%	905.52	0.33	95.75%
	7	826.32	897.36	0.66	92.08%	946.35	0.21	87.32%	912.21	0.32	90.59%
	8	765.06	819.85	0.74	93.32%	831.44	0.22	92.02%	809.81	0.60	94.47%
	9	710.30	786.48	1.21	90.31%	790.96	0.26	89.80%	785.02	0.32	90.48%
	10	1022.99	1045.68	0.56	97.83%	1121.51	0.13	91.22%	1071.89	0.39	95.44%
	Avg	819.83	856.39	0.81	95.71%	888.28	0.20	92.29%	869.05	0.40	94.34%

Table 2 Results of different neighbourhood structure in Tabu Search algorithm

			Tabu Search								
			transfer & swap			transfer only			swap only		
	Examp	Optimum	cost	time	optimality	cost	time	optimality	cost	time	optimality
20 Customers	1	968.07	971.85	18.85	99.61%	999.89	0.84	96.82%	971.85	7.05	99.61%
	2	633.24	646.99	34.98	97.87%	646.99	6.83	97.87%	700.15	8.08	90.44%
	3	609.21	609.30	41.09	99.98%	639.24	9.09	95.30%	649.13	10.62	93.85%
	4	931.65	931.65	10.90	100.00%	931.65	0.99	100.00%	956.74	3.52	97.38%
	5	864.48	881.94	24.08	98.02%	865.43	2.50	99.89%	914.34	5.07	94.55%
	6	867.01	867.01	36.08	100.00%	872.78	2.25	99.34%	899.98	12.93	96.34%
	7	826.32	826.32	53.94	100.00%	826.32	3.50	100.00%	912.21	8.32	90.59%
	8	765.06	783.06	55.90	97.70%	770.33	2.70	99.32%	809.62	8.85	94.50%
	9	710.30	710.30	46.97	100.00%	711.40	4.82	99.85%	737.93	7.54	96.26%
	10	1022.99	1022.99	52.59	100.00%	1045.68	1.03	97.83%	1044.55	6.90	97.94%
	Avg	819.83	825.14	37.54	99.32%	830.97	3.45	98.62%	859.65	7.89	95.14%

Table 3 Results of different neighbourhood structure in Simulated Annealing algorithm

			Simulated Annealing								
			transfer & swap			transfer only			swap only		
	Examp	Optimum	cost	time	optimality	cost	time	optimality	cost	time	optimality
20 Customers	1	968.07	971.85	39.69	99.61%	999.89	29.97	96.82%	971.85	10.64	99.61%
	2	633.24	633.24	39.41	99.00%	633.24	23.53	100.00%	674.30	7.12	93.91%
	3	609.21	613.54	42.50	99.29%	618.13	25.83	98.56%	623.69	19.38	97.68%
	4	931.65	931.65	34.45	100.00%	931.65	27.96	100.00%	956.35	6.23	97.42%
	5	864.48	864.48	23.64	100.00%	871.90	18.95	99.15%	914.34	8.52	94.55%
	6	867.01	867.01	45.02	100.00%	870.29	24.44	99.62%	900.91	10.37	96.24%
	7	826.32	830.54	33.30	99.49%	830.35	19.97	99.51%	912.21	13.82	90.59%
	8	765.06	783.06	44.91	97.70%	767.68	30.72	99.66%	809.81	15.23	94.47%
	9	710.30	712.13	36.35	99.74%	713.05	24.24	99.61%	762.38	15.60	93.17%
	10	1022.99	1025.15	34.89	99.79%	1028.98	21.87	99.42%	1044.55	12.60	97.94%
	Avg	819.83	823.27	37.42	99.46%	826.52	24.75	99.24%	857.04	11.95	95.56%

Appendix D Comparisons of applying different methods

The following table provides the cost, computation time, and optimality percentage of applying different algorithms in examples with10, 15, 20, and 25 customers.

Table 1 Results of applying different methods in different scales of problems

		MILP model			Heuristic			Tabu Search			Simulated Annealing		
	Examples	Cost	Time-GLPK	Time-guro	Cost	Time	optimality %	Cost	Time	optimality %	Cost	Time	optimality %
10 Customers	1	288.30	0.20	0.45	302.99	0.66	95.15%	288.30	5.03	100.00%	288.30	9.25	100.00%
	2	384.24	0.00	0.33	384.24	0.22	100.00%	384.24	10.69	100.00%	384.24	17.55	100.00%
	3	440.43	1.00	1.73	445.88	0.04	98.78%	445.88	6.84	98.78%	440.43	23.75	100.00%
	4	509.11	0.00	0.08	509.11	0.12	100.00%	509.11	0.76	100.00%	509.11	18.92	100.00%
	5	453.49	0.10	0.47	459.95	0.03	98.60%	459.95	0.27	98.60%	453.49	24.51	100.00%
	6	330.75	0.10	0.51	340.01	0.25	97.28%	330.75	13.98	100.00%	330.75	28.92	100.00%
	7	665.74	0.20	0.05	665.74	0.11	100.00%	665.74	0.24	100.00%	665.74	1.28	100.00%
	8	432.51	0.00	0.09	432.51	0.11	100.00%	432.51	2.47	100.00%	432.51	26.89	100.00%
	9	381.31	0.10	0.45	381.31	0.11	100.00%	381.31	3.71	100.00%	381.31	27.69	100.00%
	10	533.63	0.20	0.56	533.63	0.11	100.00%	533.63	1.30	100.00%	533.63	26.10	100.00%
	Avg	441.95	0.19	0.47	445.54	0.18	98.98%	443.14	4.53	99.74%	441.95	20.49	100.00%
15 Customers	1	333.27	0.20	0.83	333.77	1.01	99.85%	333.27	34.05	100.00%	333.27	46.44	100.00%
	2	405.78	7.00	1.40	405.78	0.38	100.00%	405.78	29.11	100.00%	405.78	58.26	100.00%
	3	389.07	275.20	2.65	398.00	1.40	97.76%	389.07	60.92	100.00%	389.07	42.18	100.00%
	4	444.52	7.20	2.22	495.10	0.91	89.78%	444.52	29.81	100.00%	444.52	37.85	100.00%
	5	381.53	0.20	0.48	381.53	0.86	100.00%	381.53	39.39	100.00%	381.53	36.98	100.00%
	6	410.98	3.50	2.26	424.64	0.55	96.78%	410.98	23.45	100.00%	410.98	25.38	100.00%
	7	571.12	0.20	0.17	602.61	0.15	94.77%	571.12	19.93	100.00%	571.12	30.60	100.00%
	8	734.26	9.40	1.75	764.62	0.37	96.03%	734.26	15.75	100.00%	734.26	22.64	100.00%
	9	858.24	42.80	2.09	919.17	0.23	93.37%	858.24	17.67	100.00%	858.24	74.78	100.00%
	10	656.70	16.40	1.76	705.55	0.55	93.08%	656.70	22.46	100.00%	656.70	19.09	100.00%
	Avg	518.55	36.21	1.56	543.08	0.64	96.14%	518.55	29.25	100.00%	518.55	39.42	100.00%
20 Customers	1	968.07	6830.60	2.39	971.85	0.44	99.61%	971.85	18.85	99.61%	971.85	39.69	99.61%
	2	633.24	2385.46	1.47	646.99	0.88	97.87%	646.99	34.98	97.87%	633.24	39.41	100.00%
	3	609.21	864.32	3.50	623.69	1.83	97.68%	609.30	41.09	99.98%	613.54	42.50	99.29%
	4	931.65	2175.20	2.90	961.81	0.15	96.86%	931.65	10.90	100.00%	931.65	34.45	100.00%
	5	864.48	192.50	2.48	934.74	0.47	92.48%	881.94	24.08	98.02%	864.48	23.64	100.00%
	6	867.01	1224.40	3.28	875.42	1.14	99.04%	867.01	36.08	100.00%	867.01	45.02	100.00%
	7	826.32	425.10	5.07	897.36	0.66	92.08%	826.32	53.94	100.00%	830.54	33.30	99.49%
	8	765.06	1685.10	8.66	819.85	0.74	93.32%	783.06	55.90	97.70%	783.06	44.91	97.70%
	9	710.30	447.10	4.10	786.48	1.21	90.31%	710.30	46.97	100.00%	712.13	36.35	99.74%
	10	1022.99	5895.90	13.77	1045.68	0.56	97.83%	1022.99	52.59	100.00%	1025.15	34.89	99.79%
	Avg	819.83	2212.57	4.76	856.39	0.81	95.71%	825.14	37.54	99.32%	823.27	37.42	99.56%

Table 1 Results of applying different methods in different scales of problems (Continued)

25 Customers	*1*	535.32	19210.50	148.00	543.95	3.21	98.41%	538.29	177.30	99.45%	539.43	107.17	99.24%
	2	1119.12	17892.10	24.73	1154.00	1.70	96.98%	1119.12	87.98	100.00%	1130.83	50.25	98.96%
	3	1103.00	15224.50	56.87	1249.44	0.61	88.28%	1117.34	103.40	98.72%	1121.61	42.34	98.34%
	4	914.23	31619.70	22.78	1031.79	2.18	88.61%	914.23	88.01	100.00%	968.00	26.14	94.45%
	5	982.89	144.30	2.50	1043.34	0.98	94.21%	982.89	45.75	100.00%	1003.88	38.74	97.91%
	6	1239.41	595.10	2.96	1254.75	0.68	98.78%	1254.37	33.51	98.81%	1254.75	34.69	98.78%
	7	947.35	11468.00	35.01	965.85	1.75	98.08%	965.85	85.61	98.08%	964.00	57.31	98.27%
	8	929.17	7932.40	22.66	1000.44	1.52	92.88%	929.17	51.70	100.00%	940.44	30.08	98.80%
	9	1200.62	12654.60	17.80	1231.41	0.76	97.50%	1231.41	61.14	97.50%	1231.41	45.05	97.50%
	10	975.48	2381.40	11.47	1038.21	0.87	93.96%	975.48	43.18	100.00%	984.10	35.87	99.12%
	Avg	994.66	11912.26	34.48	1051.32	1.42	94.77%	1002.81	77.78	99.26%	1013.84	46.76	98.14%