A MULTI-DEPOT CREW ROUTING OPTIMIZATION FOR POST-DISASTER SERVICE RESTORATION

by

Parth Pancholi

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

at

Dalhousie University
Halifax, Nova Scotia
April 2017

Table of Contents

List of Figures

List of Tables

Abstract

After any natural disaster, thousands of customers may get disconnected from the supply of electricity. To deal with the after effect of such calamity, the power utility companies will need to allocate the PLTs to the depots which are responsible to maintain power grids of certain areas, and schedule the sequence in which PLTs will fix the outages. The goal is to restore the service of all the customers as soon as possible while reducing the total weighted wait times across all the customers.

In this research, a Mixed Integer Linear Programming model is proposed to solve this problem optimally, as well as a Multi-Stage Metaheuristic method is proposed to achieve a near optimum solution. Additionally, a Hybrid method is proposed which uses a combination of the optimum method and metaheuristic method to solve the problem. The performances of the proposed methods are compared with each other based on simulated data.

List of Abbreviations Used

AOC	Ant Colony Optimization
CCVRP	Cumulative Capacitated Vehicle Routing Problem
CVRP	Capacitated Vehicle Routing Problem
DOM	Disaster Operations Management
GA	Genetic Algorithm
ILP	Integer Linear Programming
ILS	Iterated Local Search
k-DCT	k-Degree Centre Trees
LP	Linear Programming
MA-MCPTDR	Multiple Agents Maximum Collection Problem with Time-Dependent Reward
MCPTDR	Maximum Collection Problem with Time-Dependent Reward
MDVRP	Multi-Depot Vehicle Routing Problem
MDVRPI	Multi-Depot Vehicle Routing Problem with Inter-depot routes
MILP	Mixed Integer Linear Programming
MLP	Minimum Latency Problem
MT-CCVRP	Multi Trip Cumulative Capacitated Vehicle Routing Problem
MTMCP	Multiple Tour Maximum Collection Problem
NP	Non-Polynomial
OR/MS	Operations Research and Management Science
PDVRP	Pick-up and Delivery Vehicle Routing Problem
PLT	Power Line Technician
PVRP	Periodic Vehicle Routing Problem
SDVRP	Split Delivery Vehicle Routing Problem
TRP	Traveling Repairman Problem
TRPP	Traveling Repairman Problem with Profits
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

VRPB	Vehicle Routing problem with Backhaul
VRPM	Vehicle Routing Problem with Multiple uses of the vehicles
VRPTW	Vehicle Routing Problem with Time Window

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Alireza Ghasemi for his excellent guidance and motivation throughout this research. This thesis would not have been possible without his invaluable advice and continuous cooperation.

I would also like to thank Dr. Pemberton Cyrus and Dr. Nader Azad for offering their time and knowledge in evaluating my thesis. Moreover, I am immensely grateful to all my teachers for their academic guidance in my graduate courses.

I do not know how to thank my parents who have always believed in me and supported me in all aspects of my life. Without them, I would not be able to pursue advanced degrees. Last but not the least, a very special thanks to my friends for their continuous encouragement throughout this thesis.

Chapter 1 Introduction

Transportation costs billions of dollars to customers and manufacturers in today's distribution and logistics field. An increasing growth of e-commerce, competitive environment between the manufacturers, and increasing consumers' demand call for an effective planning of the distribution network. Hence, it is of crucial importance to design these distribution routes in an efficient manner to minimize the overall cost as well as the overall time for distribution. To cope up with the high demand of consumers, it is also pivotal to allocate the available resources in an optimal manner.

Besides the logistics field, optimum distribution networks are also important for the service industry. For instance, a utility company has several service demands from the customers distributed in different areas of the city. When the company assigns these jobs to a salesman, he must decide the order in which he will visit the customers. This must be done in the most efficient way to minimize the distance and the associated costs. By doing so, he will also be able to provide a quick service to the customers and serve more customers in a given time, which in turn will result in more profit. This problem of visiting a maximum number of locations in minimum time or traveling minimum distance is regarded as the Traveling Salesman Problem (TSP) in literature (Flood, 1956).

In a TSP, a salesman must begin his journey from one point and visit N customers (cities) exactly once before ending his journey at the same point. He should complete his journey traveling minimum distance or incurring a minimum cost. The generalized form of the TSP is called the Vehicle Routing Problem (VRP) (Laporte & Nobert, 1987). VRP can be defined as the problem of finding an optimal route for the fleet of vehicles from single or multiple depots to several customers scattered in a geographic area (Laporte G. , 1992). The purpose of the routing can be pick-up or delivery of goods or services.

In addition to logistics and service industry, VRP is also applicable for disaster management. Natural disasters, being devastating for people's lives and governments, demand for quick and efficient decision making and response in post-disaster activities. These activities are simply the restoration of the services interrupted by the disaster or delivering help such as communication, medical supply, and goods distribution to the impacted people. Delivering these services at the minimum possible time is of utmost importance in these situations.

Over the past several decades, many variants of VRP have been extensively studied which incorporate several real-life scenarios. These scenarios may include the capacity of the vehicles, the time-window for which vehicles are available or the time-window in which the locations must be visited, split deliveries, pick-up and delivery, etc. As per Lenstra & Rinnooy Kan (1987), most VRPs are NP-hard which are unlikely to be solved in a polynomial time. Therefore, it needs great attention to develop different algorithms which can solve VRP in less time. In the last few decades, several exact and approximation algorithms are developed. But because of NP-hardness of VRP, exact algorithms can only solve the small problems in many cases. On the other hand, the approximation algorithms can yield comparatively good results in less time.

In this research, the problem of post-disaster service restoration is considered. An optimum and approximation methods are proposed to solve the problem. Moreover, a hybrid technique using a combination of both optimum and approximate algorithms is developed to solve the problem. In section 1.1, the exact definition of the problem is provided, followed by section 1.2 which enlists and explains the assumptions made while solving the problem.

1.1 Problem Summary

Under any natural calamity, significant damage to the power grid can occur, disconnecting thousands of customers from the supply of electricity that supports many critical activities. In order to meet customers' expectations, an electricity supplier organization must have an effective plan for restoring the power outages as soon as possible.

In the problem under investigation, a certain number of Power Line Technicians (PLT) are present across all the depots governed by a power supplier organization, each depot is responsible for maintaining the grid for a specific geographical area during regular maintenance and repair of the system. After a disaster, the power supplier organization needs to allocate the PLTs to the depots depending on the number of outages in the whole region and the order in which PLTs will fix the outages. The PLTs spend time while traveling from one outage to another (travel time), as well as while fixing the outages (service time). The primary objective of this problem is to minimize the overall restoration time of all the customers while minimizing the total weighted wait time (secondary objective) across all the customers.

Before moving further, it is important to clearly distinguish the restoration time and the weighted wait time. The restoration time depends on the travel time and service time of the outages, whereas the weighted wait time depends on the number of customers associated with each outage (weight) in addition to the travel time and service time. For any *outage*, the restoration time is the sum of the time taken to arrive at that outage and time taken to fix that outage, whereas the weighted wait time is the restoration time of that outage multiplied by the number of customers associated with that outage. The restoration time and weighted wait time of an entire *route* traveled by one PLT are respectively the sum of the restoration times and the sum of the weighted wait times of all the outages in that route. The *overall* restoration time of the problem is the restoration time of the last outage restored, whereas the *overall* (*total)* weighted wait time is the *sum* of the weighted wait times of all the routes.

Figure 1 gives a good idea to understand both the terms. It shows the route traveled by two PLTs who start their journeys at the same time. PLT-1 follows route-A visiting two nodes, whereas PLT-2 follows route-B visiting three nodes. The time taken (in hours) to travel from one point to another is mentioned near the arrows. Let's assume that the time taken to fix any outage is 2 hours and the number of customers associated with each outage is 10 (both values may not be same for all the nodes in a real life problem).

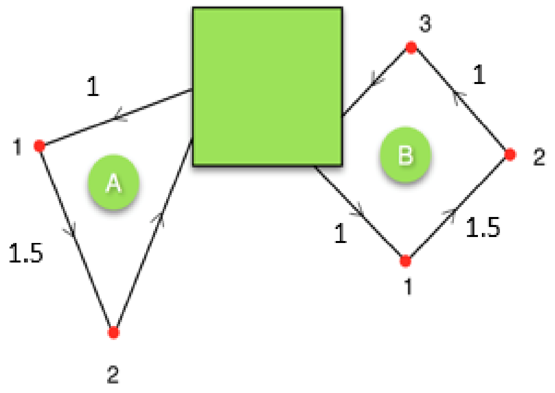


Figure 1. An instance involving one depot, five outages and two PLTs

From Figure 1:

- Restoration time (Route A) = (Time taken to travel from depot to outage 1 + Time taken to restore outage 1 + Time taken to travel from outage 1 to outage 2 + Time taken to restore outage 2)

  $\therefore$ Restoration time (A) = $(1 + 2 + 1.5 + 2) = 6.5$ hours

  $\therefore$ Restoration time (B) = $(1 + 2 + 1.5 + 2 + 1 + 2) = 9.5$ hours

- $\text{Weighted wait time (Route A)} = (\text{Restoration time of outage 1} * \text{Number of customers associated with outage 1}) + (\text{Restoration time of outage 2} * \text{Number of customers associated with outage 2})$

  $\therefore \text{Weighted wait time (A)} = (1+2) * 10 + (1+2+1.5+2) * 10 = 95 \text{ hours}$

  $\therefore \text{Weighted wait time (B)} = (1+2) * 10 + (1+2+1.5+2) * 10 + (1+2+1.5+2+1+2) * 10 = 190 \text{ hours}$

Therefore, the objective function values are as follows:

- $\text{Overall Restoration time} = \max(6.5, 9.5) = 9.5 \text{ hours}$
- $\text{Total Weighted Wait time} = (95 + 190) = 285 \text{ hours}$

One important thing to note is the contradictory behavior of primary and secondary objectives in some cases. The improvement in overall restoration time does not always guarantee the improvement in weighted wait time, in contrast, it may increase the weighted wait time in some instances. The reason behind this is *the number of customers* associated with each outage, which is considered in weighted wait time but not in restoration time. In the quest of improvement in restoration time, it is quite possible that an outage which is getting fixed first has very less number of customers than the outage getting fixed later. In that case, although the solution will improve the restoration time on one hand, it may increase the weighted wait time on the other hand.

1.2 Problem assumptions

In this section, the assumptions made in solving the problem are discussed.

- All the PLTs start their journeys at the same time from multiple depots.
- All the PLTs are assumed to have the same efficiency resulting in homogeneous fleet.

- The calculation of restoration and weighted wait time is started when the PLTs start their journeys *to fix* the outages from the depot. Therefore, the time spent for the allocation of PLTs to the depots is neglected.
- It is mandatory to fix all the outages.
- All the outages will be fixed on the first visit.
- The restoration time and the weighted wait time remain constant as soon as all the outages are restored by the PLTs. Therefore, any PLT's trip from the last visited outage back to the depot doesn't change the objective functions.
- The time to fix each of the outages is predetermined based on historical data of that outage or using industry standards.
- The factors which cannot be realized in advance are not considered in this problem. These factors may include the delay caused by weather and traffic, stochastic service delay due to lack of equipment, etc.

The remainder of this thesis is organized as follows. In chapter 2, a literature review related to the problem is presented. The proposed optimum method (Formulation), approximate method and hybrid method are described in chapter 3. Chapter 4 describes the computation experiments and comparison between the proposed methods using simulated data. Finally, conclusions and future research directions are discussed in chapter 5.

Chapter 2 Literature review and background

This chapter discusses the previous research work related to the problem under investigation. The evolution of the Vehicle Routing Problem (VRP) from its simplest form to the complex variants addressing the real-life scenarios, as well as the development of different techniques to solve the VRP and its variants are discussed in this chapter. Moreover, the classes of VRP which closely relate to the problem under investigation are discussed in detail.

2.1 Vehicle Routing Problem and its variants

Dantzig & Ramser (1959) were the first to propose a model to route a fleet of homogeneous trucks from the central hub to number of gas stations to fulfill their demand for gasoline. After five years, Clarke & Wright (1964) related this problem with the delivery of goods in the field of logistics and transportation. They converted this problem to a linear optimization problem which became popular as a 'Vehicle Routing Problem (VRP)'. Christofides (1976) seems to be the first one to use the term 'Vehicle Routing Problem' which is one of the most studied problems in the field of combinatorial optimization. The classic VRP is defined on the graph $G = (V, A)$, where $V = \{1, \dots, n\}$ is the set of the *cities* to be visited with the *depot* on vertex 1, and A is a set of arcs, $A = \{(i, j) : i, j \in V\}$. A *distance/cost matrix* $C = (c_{ij})$ is associated with every arc (i, j). In some conditions, c_{ij} may stand for the *travel time* or a *travel cost* depending on the problem. Initially, a fleet of m vehicles is located at the depot. We need to find m least cost optimal routes such that: (i) each vehicle start and end its journey at the depot (ii) each customer is visited exactly once and by one vehicle only. (iii) according to the problem, some side constraints such as capacity of vehicle, availability of customers and vehicles, etc. are satisfied (Laporte G. , 1992).

Over the past few decades, the burgeoning use of VRP in various real life scenarios resulted in the invention of many variants of VRP, which motivated the researchers to take more and more interest in this field. Some of the variants are discussed below:

Two of the most basic variants of VRP are *Capacitated Vehicle Routing Problem* (CVRP) and *Vehicle Routing Problem with Time Window* (VRPTW). In CVRP, the fleet of the vehicles is initially located at the central depot. The objective is to route these vehicles to fulfill the demands of all the customers incurring a minimum cost such that the total demand of any route does not exceed vehicle's capacity, and above-mentioned general VRP constraints are satisfied (Laporte, Mercure, & Nobert, 1986). CVRP can also be found in the work of (Toth & Vigo, 2002; Akpinar, 2016; Uchoa, Pecin, Pessoa, Vidal, & Subramanian, 2017). In VRPTW, the cities must be visited in their respective time window. Sometimes the vehicles are only available for particular time window; they have to start and end their journeys within that time period. Hence, a time-window constraint must be satisfied in addition to general VRP requirements (Solomon & Desrosiers, 1988; Kallehauge, 2008; Braaten, Gjønnes, Hvattum, & Tirado, 2017). When we have a small fleet of vehicles available to visit a large number of customers, or when a vehicle is available for a longer period of the day as compared to the average duration of a route, a vehicle can be assigned more than one route. This type of problem is called *Vehicle Routing Problem with Multiple Use of Vehicles* (VRPM) (Azi, Gendreau, & Potvin, 2010; Koç & Karaogˇlan, 2012). In the available fleet of vehicles, if the capacities of all the vehicles are same, the problem is called *VRP with a homogeneous* fleet of vehicle, and if different types of the vehicles are used, it is called *VRP with a heterogeneous* fleet of vehicles. Moreover. Instead of one-day planning, if the standard VRP's planning period extends for several days, it is called *Periodic Vehicle Routing Problem* (PVRP) (Dayarian, Crainic, Gendreau, & Rei, 2015; Ouaddi, Benadada, & Mhada, 2016)

In large distribution networks, several depots may be considered instead of one central depot. This type of problem is called *Multi-Depot Vehicle Routing Problem* (MDVRP). MDVRP first appeared in the work of (Kulkarni & Bhave, 1985) and (Laporte, Nobert, & Taillefer, 1988). In standard MDVRP, the vehicles start their respective journeys from different depots and end their journeys at the same depot from where they started, satisfying all the constraints of standard VRP (Montoya-Torres, López Franco, Nieto Isaza, Felizzola Jiménez, & Herazo-Padilla, 2015; Contardo & Martinelli, 2014). In MDVRP, vehicles cannot generally travel directly from one depot to another; if they can, it is called *Multi-Depot Vehicle Routing Problem with Inter-Depot Routes* (MDVRPI). MDVRPI can be found in the work of (Jordan & Burns, 1984; Crevier, Cordeau, & Laporte, 2007). In some cases, depots also work as a replenishment facility where vehicles may be replenished with the delivery product at intermediate depots along their route.

Another variant of VRP is *Pick-up and Delivery Vehicle Routing Problem* (PDVRP), in which goods are picked up from some customer's location and delivered to other customers (Li, Wang, Hei, Li, & Jiang, 2017). In contrast to PDVRP, in *Vehicle Routing problem with Backhaul* (VRPB), the goods are delivered from several depots or from one central depot to set of linehaul customers, and from backhaul customers to the depots (Goetschalckx & Jacobs-Blecha, 1989; Wassan, Wassan, Nagy, & Salhi, 2017). Another popular example of VRP is *Split Delivery Vehicle Routing Problem* (SDVRP) in which split deliveries to one customer by *more than one* vehicle are allowed, in contrast with the classical VRP where one customer can only be visited exactly once and by one vehicle only (Archetti & Speranza, 2012; Chen, Golden, Wang, & Wasil, 2017).

In addition to above variants, there exist more complex variants inspired by real life scenarios. For example, Oesterle & Bauernhansl (2016) proposed an exact method for a vehicle routing problem with mixed linehaul and backhaul customers, heterogeneous fleet, time window and manufacturing capacity. Gao, Wang, & Liu (2016) considered a VRPTW with multiple objectives of simultaneously minimizing cost, the number of vehicles used, and the traveled distance. Zhang et al. (2017) considered VRPTW with realistic pallet loading constraints in which goods of different sizes must be transported in standard sized boxes with limited capacity. They addressed the actual needs of the logistics industry for distribution of consumer goods and agricultural products. Some other examples of complex VRP variants are split delivery and pickup (Annouch, Bellabdaoui, & Minkhar, 2016), fuel consumption and stochastic travel speed (Feng, Zhang, & Jia, 2017), multi-objective vehicle routing problem for simultaneous delivery and pick up with time windows (Li, Wang, Hei, Li, & Jiang, 2017), etc. The increasing complexity of these variants demands efficient methods to solve VRP which are discussed in the next section.

2.2 Proposed methods to solve VRP and its variants

As far as complexity is concerned, VRP is considered NP-hard problem because of the augmented Traveling Salesman Problem (TSP) which is a popular NP-hard problem itself (Garey & Johnson, 1979). There exist exact algorithms which can solve TSP containing hundreds or thousands of vertices (Applegate, Bixby, Chvátal, & Cook, 2007), whereas VRP is much more difficult to solve in real life for which best exact algorithms can handle a couple of hundred customers only (Baldacci, Christofides, & Mingozzi, 2008) (Baldacci, Mingozzi, & Roberti, 2012). Over the past few decades, extensive research has been done in developing exact and approximation methods to solve VRP and its variants as discussed below:

2.2.1 Exact methods

Since the beginning of VRP era, the exact methods to solve VRP have transformed from basic branch-and-bound algorithms to highly advanced mathematical programming. Among them, the main algorithms are discussed here.

2.2.1.1 Linear Programming

Linear programming (LP) or linear optimization is a method to maximize or minimize a linear function. It uses a mathematical model in which all the requirements of the problem (constraints) are linear. If all the decision variables of a model are constrained to be integers, the method is called Integer Linear Programming (ILP). And if the mathematical model has some integer variables and some non-integer decision variables, the method is called Mixed Integer Linear Programming (MILP) (Hiller & Lieberman, 2015). Over the past few decades, the algorithms to solve LP models have evolved from a branch-and-bound algorithm (Davis, Kendrick, & Weitzman, 1971) to branch-and-cut algorithm (Finkel'shteyn, 1971), column generation (Og˜uz, 2002), set partitioning formulation (Marsten, 1974), vehicle flow formulation (Laporte, Nobert, & Desrochers, 1985), etc. Some of the popular algorithms to solve VRP optimally are discussed below.

Branch and bound algorithms are one of the most popular algorithms to solve the NP-hard problems in the field of combinatorial optimization. The basic concept of branch-and-bound is to divide a large problem into smaller and smaller sub-problems and solve them to obtain the solution of the original undivided problem. The algorithm systematically calculates the candidate solutions one by one forming a rooted tree. It then explores the *branches* of the tree which represent the subset of the solution set. Each candidate solution (branch) is tested against estimated lower and upper *bounds* on the optimum solution. The algorithm discards the branch if it cannot produce a better solution than the best one found so far. Little et al. (1963) used the branch-and-bound name for the first time in their work on the traveling salesman problem. For VRP, it seems to appear in the work of (Christofides & Eilon, 1969) for the first time. The performance of the algorithm was greatly improved by two effective lower bounds by *q-routes* and shortest spanning *k-Degree Centre Trees* (*k-DCT*) introduced by Christofides, Mingozzi, & Toth (1981) solving up to 25 customers. Kumar & Jain (2015) applied branch-and-bound algorithm in school bus routing problem and were able to optimally solve the instances up to 65 buses.

The *Branch-and-cut algorithm* combines the branch-and-bound algorithm with cutting planes to solve Integer Linear Program (ILP) in which the cutting planes restrict the non-integer solutions tightening the Linear Programming (LP) relaxation. These cuts reduce the number of branches. Fisher (1994) incorporated *k-DTC* lower bound within a branch-and-cut algorithm and could solve instances up to 134 vertices.

Balinski & Quandt (1964) were the first to provide a simple *set partitioning formulation* for VRP. The set-partitioning algorithm consists of the m×n binary matrix in which each row represents a customer and each column represents a feasible route. The objective is to cover all the customers (rows) with subsetting routes (columns) incurring a minimum cost. However, the formulation is not that efficient because of (1) a large number of potential routes, and (2) difficulty in solving the optimum cost for an exponential number of NP-hard routes (Laporte G. , 2009). To solve this problem, several *column generation* algorithms were proposed (Rao & Zionts, 1968; Foster & Ryan, 1976; Agrawal, Mathur, & Salkin, 1989). Some LP models are too large to consider all the variables explicitly, and since most of the variables are assumed to have zero value in the objective function, the column generation only considers a subset of the problem and generates only those variables which have potential to improve the objective function. Column generation is widely used exact algorithm to solve vehicle routing problem with time window (VRPTW) since (Desrochers, Desrosiers, & Solomon, 1992). When column generation is combined with branch and bound algorithm, it is called *Branch and price algorithm.* Fukasawa, et al. (2006) developed a branch-cut-and-price algorithm which showed that the combination of cut and column generation can be much more effective. Most recently Pecin et al. (2017) proposed an improved branch-cut-and-price algorithm for the capacitated vehicle routing problem by combining several elements of the previous research. Their algorithm was able to optimally solve CVRP for up to 360 customers, which were only considered by approximation method before.

2.2.1.2 Dynamic Programming

Dynamic programming is a *recursive* approach to solve a complex sequential problem by breaking it into simpler sub-problems. Then, each sub-problem is solved and the solution is stored. The solutions are used to make the sequence of interrelated decisions. Unlike linear programming, dynamic programming does not have a standard mathematical formulation. It is a general approach to problem-solving in which the equations are developed to fit the situation under the study. The dynamic programming based exact algorithm to solve VRP can be seen in the work of (Christofides, Mingozzi, & Toth, 1981; Mahmoudi & Zhou, 2016).

Other exact algorithms to solve VRP are *Vehicle flow formulation based algorithm*, *A commodity flow formulation based algorithm*, etc. The Vehicle flow formulation based algorithm can be subdivided in (i) Two- index vehicle flow formulation (Laporte, Nobert, & Desrochers, 1985) and (ii) Three-index vehicle flow formulation. The Commodity flow formulation based algorithm can be subdivided into (i) Single-commodity flow, (ii) Two-commodity flow (Baldacci, Hadjiconstantinou, & Mingozzi, 2004), and (iii) Multi- commodity flow (Letchford & Salazar-González, 2015).

As mentioned earlier, VRP is NP-hard problem. Therefore, it is often not possible to solve large problems by exact methods.

2.2.2 Heuristics

A heuristic is a rule of thumb technique, designed to solve the problems more quickly when the exact methods fail to do so in a "reasonable" amount of time. It is a technique to find a satisfactory approximate solution when exact methods cannot find an optimum solution.

Dantzig & Ramser (1959) seem to introduce the first heuristic for VRP in their paper. Their algorithm iteratively matches vertices and partial routes, to form a final set of the vehicle route. They do so by setting the decision variables to 1 if two vertices, or vertices and partial routes are matched. They illustrated this heuristic for seven-vertex indices. Since then, several heuristics were developed among which some were construction heuristics whereas some contained improvement phase. Some of the popular heuristics are explained below.

The *saving heuristic* was first proposed by Clarke & Wright (1964) which yields a good solution, and is easy to describe and implement. The algorithm starts with an initial route containing the depot (0) and one other vertex (i) i.e. $(0, i, 0)$. At each iteration, it merges two routes which are feasible to merge (one ending with i and the other starting with j and maximize the saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$), where c_{ij} denotes the cost incurred while traveling from vertex i to vertex j. The algorithm is terminated when no merging of routes is further possible. The output of this algorithm is the minimum number of vehicles used in the solution. Some enhancements in the original algorithms were proposed by several researchers afterwards: Nelson et al. (1985) proposed six methods for implementing the saving heuristic. They tested 55 instances to compare these methods and they suggested which method to choose for different VRPs. Paessens (1988) proposed the enhancement making use of efficient data structures to speed up the computation requiring reduced storage. Altinel & O¨ncan (2005) proposed a new enhancement in Clarke-Wright's algorithm which differed from the previous enhancements in saving criteria: they considered customer demands in addition to the distances. Gronalt et al. (2003) proposed four different saving heuristics for the pickup and delivery problem under time window constraint which were able to find a good solution in quick time.

Another popular heuristic is *Cluster-first, Route-second heuristics*. For VRP, this algorithm was first introduced by Fisher & Jaikumar (1981). It first locates a certain number of *seeds* and then clusters the customers to each seed so as to minimize the sum of the distances between seed and customer, while satisfying the capacity constraint. Then, one vehicle is assigned to each cluster and the vehicle route is constructed for each cluster. After that, Baker & Sheasby (1999) proposed better methods to select the seeds. This heuristic follows two steps to achieve the solution, i.e. Clustering and Routing. This can be advantageous as the method can take benefit from the algorithmic improvements for any of the Clustering or Routing.

Moreover, one of the most popular heuristics is *Two-phase heuristics*. As mentioned in (Laporte G. , 2009), in the first phase of this heuristic, initial routes of the vehicles are *constructed* using the construction heuristics, and in the second phase, those routes are *improved* through improvement heuristics. The initial VRP solution generated by construction heuristics can be post-optimized using two broad types of methods: (i) *Intra-route improvement* (ii) *Inter-route improvement*. In Intra-route improvement, each route is improved separately using TSP algorithms, whereas, in Inter-route improvement, several routes are improved simultaneously. The algorithms to improve single route can be found in the TSP literature review conducted by Laporte (2009). In Inter-route, the routes are improved by transfer or swap steps. In transfer step, one or several vertices are removed from one route and relocated to another route, whereas in a swap, the same number of vertices are swapped between two routes in search for the better overall solution. According to Toth & Vigo (2014), inter-route heuristics play a significant role in achieving good results. Wang & Wang (2009) implemented two-phase heuristics for VRP with backhaul.

In general, a heuristic is a problem-dependent approach. They take the advantage of specialty of the problem. However, they often perform in a greedy way and do not allow deterioration in the objective function from one iteration to the next one. Therefore, they often get trapped in the local optimum and cannot achieve the global optimum. To overcome this disadvantage, metaheuristics are used which provide a better chance to escape from local optimum.

2.2.3 Metaheuristics

Metaheuristics are the problem-independent techniques, which allow deteriorating intermediate solutions during the search process, providing a better chance to get out of the local optima, which results in more extensive search for the global optima (Bräysy & Gendreau, 2005). Therefore, metaheuristics have a better chance to achieve global optimum than heuristics. Because of this feature, the metaheuristic approach has captured the interest of the researchers over the past twenty-five years. Most of the metaheuristics work as improvement methods and produce a relatively high-quality solution even if they are initiated from a low-quality solution. Metaheuristics can be broadly classified as local search, population search, and learning mechanism. In the search for the near optimum solution, local search explores the solution by moving from one solution to another in its neighborhood, whereas population search works with the entire population of trial solutions rather than just one. The learning mechanism learns from its experience during the progression of the search process. In recent years, the hybrid methods are developed which use more than one principle of main metaheuristics, for example, memetic algorithms combine the principles of local search and population search. Li, Tian, & Leung (2009) hybridized the principle of learning mechanism with local search to solve open vehicle routing problem. The basic principles of the algorithms and special features of important metaheuristics are discussed below.

2.2.3.1 Local search

In the search for the best solution, local search explores the solution space by moving from the current solution to the best one in its neighborhood at each iteration. The performance of the local search highly depends on two things: (i) the mechanism used to define the neighborhood, and (ii) the method used to explore the solution space (Laporte G. , 2009). The classical metaheuristics using the principle of local search are tabu search, simulated annealing, very large neighborhood search (Ergun, Orlin, & Steele-Feldman, 2006; Kytöjoki, Nuortio, Bräysy, & Gendreau, 2007), an adaptive large neighborhood search (Ropke & Pisinger, 2006; Mattos & Laporte, 2012; Mancini, 2016), and variable neighborhood search (Polacek, Hartl, Doerner, & Reimann, 2004; Liu, Liu, & Zhang, 2008; Wei, Zhang, Zhang, & Lim, 2015). Among them, two of the most popular metaheuristics are tabu search and simulated annealing.

Tabu search is a metaheuristic technique which accepts the non-improving solutions in addition to the improving solutions. Moreover, it records the recently visited solutions in a list called *tabu* list, and prohibits the search returning to those solutions for a certain number of iterations. This attribute helps the search process to get out of the strict local optimum in search for the global optimum. For tabu move, one exception is there that the tabu-move is allowed if and only if the resulting solution is the best solution achieved so far. Tabu search can be found in the work of (Gendreau, Laporte, Musaraganyi, & Taillard, 1999; Derigs & Reuter, 2009; Silvestrin, Paulo, & Ritt, 2017).

On the other hand, simulated annealing mimics the annealing process of metallurgy, a technique in which the material is heated at high temperature and slowly cooled down to reduce the defects in its crystals. Simulated annealing is a probabilistic technique to achieve the approximate global optimum in the larger solution spaces. The unique feature of this metaheuristics is that at each iteration it does not directly reject the non-improving solution, but it accepts it with some probability, allowing an extensive search for the global optimum. The slow cooling in the annealing process is interpreted as the slow decrease in the probability of accepting the non-improving solution in the simulated annealing. (Chiang & Russell, 1996; Lin, Ying, Lee, & Hsi, 2007; Yu, A.A.N., Hidayat, & Wibowo, 2017) implemented simulated annealing to solve different variants of VRP.

2.2.3.2 Population search

Unlike local search where a single solution is tackled at a time, population search works with the entire population of trial solutions. The best-known example of population search is the Genetic algorithm (GA). The core of the genetic algorithm is defined by the analogy of the biological evolution. In each iteration, the fitness of each of the trial solution from the population is calculated based on their objective function value. Based on this fitness index, two fittest parent solutions are stochastically selected to generate the new solution (child) which will contain some properties of both the parents. This is done by crossover. If a mutation occurs in the child solution, it can contain the property that none of its parents possessed, which helps the algorithm to explore the new part of the feasible region. Depending on its fitness value, the child will replace the worst solution in the population and it will be used to construct the next generation. Similarly, the desired number of children are generated in each iteration, which results in a new and better population of the solutions. The algorithm is terminated when the stopping criteria are achieved. The stopping criteria can be the level of fitness or a maximum number of generations (Hillier & Lieberman, 2015). The Genetic Algorithm is well documented in the work of (Potter & Bossomaier, 1995; Ting & Huang, 2005; Mocková & Rybicˇková, 2014; Kurniawan, Sulistiyo, & Wulandari, 2015)

Rochat & Taillard (1995) proposed an excellent method to combine the population search with local search. In this method, the local search is first applied to the initial solution and the best-known solutions are recorded. Then, the genetic search is applied on these solutions to create the new candidate solutions. After that, the local search is restarted from these new candidate solutions, and the memory is updated to the newly recorded solutions. Recently, Seidgar et al. (2016) combined simulated annealing and genetic algorithm to solve multi-server vehicle routing problem with multi-entry. The progress in genetic algorithms for VRP is well documented in a survey conducted by (Karakatič & Podgorelec, 2015).

2.2.3.3 Learning Mechanism

As the name suggests, these algorithms learn from their experience while progressing. One of the most popular forms of this algorithm is Ant Colony Optimization (AOC). The ant colony optimization originally proposed by Dorigo & Gambardella (1997), is a probabilistic technique which mimics the food seeking behavior of ants from their colony to the source of food. When an ant finds the source of food, it returns to its colony laying down a trail. If the other ants find this trail, they start following the trail in search of the food. Since more and more ants are likely to take the shortest path, the trail on the shortest path becomes denser. Similarly, in this algorithm, more weight is put on those parts which appear commonly in good solutions. This leads to the convergence towards a near optimum solution. For instance, to solve the CVRP using AOC, the researchers have provided two stages. In the first stage, each ant starts its journey from depots and return to the depot after visiting the customers. In the second stage, the feasible solution is created from ants' good paths. To improve AOC, several enhancements are proposed by the researchers. Some of them are to update the trail (Mavrovouniotis & Yang , 2015), to introduce multiple types of ants (Rizzoli, Montemanni, Lucibello, & Gambardella, 2007), to reduce the solution space (Baskan, Haldenbilen, Ceylan, & Ceylan, 2009), etc. Wang et al. (2016) simplified the construction of feasible paths by allowing the ants to go in and out from the depots until it visits all the customers. This novel approach performs very well compared to conventional AOC. Another learning algorithm to solve VRP is a Neural Network (NN) algorithm which can be found in the work of (Torki, Somhon, & Enkawa, 1997; Steinhaus, Shirazi, & Sodhi, 2015).

2.3 Classes of the problem closely related to the thesis problem

One of the variants of Capacitated Vehicle Routing Problem (CVRP) is *Cumulative Capacitated Vehicle Routing Problem (CCVRP)* whose objective is to minimize the sum of the "arrival times" (time taken to arrive) of the vehicles at each customer instead of the total routing time or cost or distance. This objective is similar to the "primary" objective of this thesis- to minimize the overall Restoration time. In CCVRP, a set of k vehicles, each with same capacity $Q > 0$ are available at a depot and supposed to visit n customers, where customer i has a demand q_i. If the vehicle is traveling from a customer *i* to customer *j,* then the arrival time at customer *j* is the sum of the travel time from customer i to *j,* and the time taken by the vehicle before reaching to customer *i.* The remaining constraints are same as CVRP, except the objective in CCVRP is to minimize the sum of all n arrival times at the customers. Lysgaard & Wøhlk (2014) seems to propose the first exact algorithm (branch-and-cut-and-piece algorithm) for CCVRP. Before that, some researchers solved CCVRP with heuristic and metaheuristic approach which includes Memetic algorithms (Ngueveu, Prins, & Wolfler Calvo, 2010), Local Search (Chen, Dong, & Niu, 2012), Adaptive Large

Neighborhood Search (Ribeiro & Laporte, 2012), two-phase heuristic (Ke & Feng, 2013), etc. Ke & Feng (2013) compared their two-phase heuristics with Memetic algorithms (Ngueveu, Prins, & Wolfler Calvo, 2010) and Adaptive Large Neighborhood Search (Ribeiro & Laporte, 2012), and concluded that two-phase heuristics and Adaptive Large Neighborhood Search outperformed Memetic algorithms for over fifty percent of the tested instances. Recently, Rivera et al. (2016) worked on one variant of CCVRP called Multi-trip Cumulative Capacitated Vehicle Routing Problem (MT-CCVRP) in which a vehicle can perform multiple trips to serve the customers. They proposed two mixed integer linear programs- flow based model and set partitioning model. Moreover, they also developed an exact method which transforms the MT-CCVRP into a resource constrained shortest path problem.

In the literature, CCVRP is also known as k-Traveling Repairmen Problem (k-TRP) (where k represents the number of available vehicles or repairmen) (Fakcharoenphol, Harrelson, & Rao, 2007), and Multi-Vehicle Minimum Latency Problem (Post & Swamy, 2015). These classes, in turn, are the generalization of Traveling Repairman Problem (TRP) (Dewilde, Cattrysse, Coene, Spieksma, & Vansteenwegen, 2013) and Minimum Latency Problem (MLP) (Blum, et al., 1994) respectively, which are also regarded as The Traveling Deliveryman Problem in literature (Méndez-Díaz, Zabala, & Lucena, 2008).

In networks, latency is a metric used to define the length of the tour from each vertex to other vertices in the network. Latency at customer *i* is basically the distance traveled by the vehicle before visiting customer *i.* The Minimum Latency Problem (MLP) aims to minimize the sum of the latencies of all visited customers. Therefore, MLP can be stated as the “customer-centric” routing problem. Several approximation algorithms are proposed to solve MLP (Goemans & Kleinberg, 1998; Chaudhuri, Godfrey, Rao, & Talwar, 2003; Arora & Karakostas, 2003; Archer, Levin, & Williamson, 2007). Among them, Chaudhuri et al. (2003) have achieved the current best approximation ratio of 3.59α, where α is an approximation factor and 3.59α is the ratio of the latency of the i^{th} vertex in the best tour found by the algorithm to the latency of the i^{th} vertex in the optimum tour. Several exact algorithms are available to solve MLP such as dynamic programming (Wu, 2000) and branch-and-bound algorithm (Wu, Huang, & Zhan, 2004). The branch and bound algorithm proposed by Wu et al. (2004) is much more efficient than the dynamic programming proposed by Wu (2000) earlier. Post & Swamy (2015) proposed linear programming based approximate algorithms for Multi-Depot Multi-Vehicle Minimum Latency Problem (multi-depot *k*-MLP). Using the LP-relaxation technique, they have achieved 7.183α-approximation for the problem where all the vehicles start their journey from the same depot, and 8.497α-approximation for the problem where the vehicles are located at different depots.

In addition to the arrival times considered in CCVRP and MLP, in TRP the time spent at each customer (repair time) may also be considered. The objective of TRP is to minimize the sum of the wait time of all the customers (vertices). The wait time of customer *i* is actually the arrival time of the repairman at customer *i.* This problem is studied by a large number of researchers (Afrati, Cosmadakis, Papadimitrious, Papageorgiou, & Papakostantinou, 1986; García, Jodrá, & Tejel, 2002; Salehipour, Sörensen, Goos, & Bräysy, 2011). Fakcharoenphol et al. (2007) proposed *k*-TRP for the first time and they proposed 8.497α-approximation algorithm for this problem. Jothi & Raghavachari (2007) considered both cases where the repair times for all the customers were same as well as different. Recently, Nucamendi-Guillén et al. (2016), proposed a mixed integer formulation to solve the *k*-TRP which produces optimum results in lesser time outperforming the previously proposed algorithms for *k*-TRP.

Dewilde et al. (2013) studied *Traveling Repairman Problem with Profits (TRPP)* in which a profit is obtained when the customer is visited by a repairman (vehicle) for the first time. The objective is to maximize the sum of the profits minus the sum of the arrival times. In this problem, it is not mandatory to visit all the customers. This introduces us to the new class of the problems called *Maximum Collection Problem with time-dependent rewards (MCPTDR).* Consider a graph $G(V, E)$, where $V = \{0, 1, 2, \dots, n\}$ denotes the locations to be visited with depot located at vertex 0, E represents the set of edges $E = \{(i, j) : i, j \in V\}$. Reward r at each node $j \in V$ (except depot where $r_0 = 0$) is a decreasing function of time and denoted by $r_j(t) = \max(b_j - s_j t, 0)$, where b_j and s_j denote an initial reward and penalty rate at location j, respectively; $t \geq 0$ is an arrival time at the location. The time required to collect the reward at location j can be denoted by v_j. The objective is to select which locations to visit and in what order, to maximize the total reward collected. Time is spent while traveling between the locations as well as while visiting the locations. The reward r_j is collected upon arrival at location j and time v_j is spent before location j is departed. Erkut & Zhang (1996) proposed penalty-based-greedy heuristic and branch-and-bound algorithm for this problem that could solve up to 20 nodes optimally. Ekici & Retharekar (2013) solved the same reward collection problem with *multiple agents (MA-MCPTDR)* where all the agents are initially located at a central depot. The objective is to route multiple agents to maximize the total surplus (total reward minus total travel cost). The agents start and end their journey at the depot. They used *cluster-first, route-second* heuristic to solve this problem in which they first clustered the locations using well known *k*-means clustering algorithm and then solved single agent routing problem for each cluster. They proposed a Modified Penalty-Based Heuristic in which they modified a loss function of not visiting a node proposed by Erkut & Zhang (1996). Ekici & Retharekar (2013) also proposed several other heuristics to solve MCPTDR such as:

Profit-Based Heuristic in which the profit per unit time is calculated for each node and the node providing maximum profit per unit time is visited next, Total Loss-Based Heuristic in which they calculated the loss incurred in reward of other nodes if one node is chosen to visit next and based on total loss the next node to be visited is selected, and Profit-Loss Based Heuristic in which both profit and total loss for the remaining nodes are calculated and based on their (profit and total loss) difference the next node is selected. They have also proposed several improvement ideas to improve the solution obtained from the initial routing of the agents. Butt & Ryan (1998) also, proposed an algorithm using set-partitioning formulation and column generation to optimally solve Multiple Tour Maximum Collection Problem (MTMCP). The difference between MTMCP and MA-MCPTDR is that the decreasing rate of reward in each problem was different which significantly affected the routing.

The objective of MCPTDR can be compared with the secondary objective of this thesis- to minimize the *total weighted wait time.* The "reward" and "penalty" of MCPTDR can be compared with the "wait time" and "weight (number of customers associated with each outage location)", respectively. In contrast, to "maximize" the total reward in MCPTDR, we need to "minimize" the total weighted wait time. Additionally, the reward is a "decreasing" function of time, whereas "wait time" is an increasing function of time. Moreover, in MCPTDR the locations producing zero rewards may not be visited, but in the thesis's problem, it is mandatory to visit all the locations. Additionally, in MCPTDR the reward is collected upon arrival at the location and after that, some time is spent before leaving the location, therefore, the reward does not account for the time spent at the location. Whereas in thesis's problem, the customers have to wait until the outage at their location is restored. Therefore, the time spent before departing the location is also included in the wait time of the customers at that location.

In addition to the above-mentioned classes of problems, extensive research has been done in the field of disaster operations management (DOM). The Operations Research and Management Science (OR/MS) community has evolved significantly in the field of DOM in last few years, providing better ways for humanitarian aid after disasters. The researchers in this field have addressed several criteria in their objective- e.g. cost, response time, travel distance, reliability, security, equity, etc. Among them, response time is closely related to the primary objective of the thesis's problem. Campbell et al. (2008) were the first to show the importance of considering the objective function regarding the urgency of the situation rather than classic cost-minimizing routing problems. They proposed algorithms to solve two different objectives: one was to minimize the maximum arrival time and the second one was to minimize the average arrival time. They solved both objectives separately. Later, Lysgaard & Wøhlk (2014) used this concept to propose their objective function for CCVRP which was to minimize the sum of the arrival times at all locations. Several surveys have addressed the development of OR/MS community in DOM, which provides better ideas and different ranges of the problems and their solutions (Altay & Green, 2006; Galindo & Batta, 2013; Gutjahr & Nolz, 2016).

In the problem of the thesis, both primary (minimize restoration time) and secondary (minimize total weighted wait time) objectives are addressed simultaneously. To our knowledge, in the literature, both objectives are addressed separately in which the primary objective is clearly addressed in several classes, while the secondary objective is addressed in MCPTDR with the differences mentioned above. Moreover, in this thesis, a hybrid approach is developed to solve the problem which uses a combination of heuristics, metaheuristics, and optimum method. The incorporation of the optimum method in metaheuristic method is to our knowledge, is the first attempt to solve this type of problem.

Chapter 3 Methodology

In the problem under investigation, a certain number of Power Line Technicians (PLT) are present across all the depots governed by a power supply organization. After a disaster, the power supply organization needs to allocate the PLTs to the depots depending on the number of outages in the whole region. After that, the power supply organization needs to decide the order in which PLTs will fix the outages. The primary objective of this problem is to minimize the overall restoration time of all the customers which is valuable for the organization and to minimize the total weighted wait time of all the customers which is more valuable for the customers and has some value to the organization. In this chapter, we will discuss three approaches which are proposed to solve this problem: (i) Optimum method- MILP model (ii) Multi-Stage Metaheuristic method (iii) Hybrid method- combining optimum method with metaheuristics and heuristics.

3.1 Optimum Method (Formulation)

In this method, a mathematical model is developed using a Mixed Integer Linear Program (MILP) to obtain the optimal solution. Erkut & Zhang (1996) developed an MILP model for the maximum collection problem with time-dependent rewards. Using the concept of their model, a new objective function and some additional constraints are developed according to the problem under investigation. The objective function and all the constraints of the model are linear. Assume that there are m depots and K total PLT crews available to fix n outages.

3.1.1 Sets and Parameters

D	Set of all the depots, $\{1,2,\dots,m\}$
R	Set of all the outages, $\{m+1,\dots,m+n\}$

s_i	Number of customers (weight) associated with outage i
v_i	Service time to fix an outage i
d_{ij}	Time to travel between depot/outage i and depot/outage j
K	Total number of PLT crews available
α_1	Importance factor of maximum restoration time objective
α_2	Importance factor of total weighted wait time objective

3.1.2 Decision Variables

$$x_{ij} = \begin{cases} 1, & \text{if outage } j \text{ is visited right after outage } i \\ 0, & \text{Otherwise} \end{cases}$$

$t_i \geq 0$ Arrival time at outage i

$r_i \geq 0$ Restoration time of outage i

$Z \geq 0$ Maximum overall restoration time

3.1.3 Objective function

Minimize: $\alpha_1 * Z + \alpha_2 * \sum_{i \in R} s_i(t_i + v_i)$ (1.1)

3.1.4 Constraints

Subject to:

$r_i = t_i + v_i \; ; \; \forall \, i \in R$ (1.2)

$t_j \geq t_i + v_i + d_{ij} - M(1 - x_{ij}) \; ; \; \forall \, i \in (R \cup D), j \in R$ (1.3)

$\sum_{j \in (R \cup D)} x_{ij} = 1 \; ; \; \forall \, i \in R$ (1.4)

$\sum_{i \in (R \cup D)} x_{ij} = \sum_{i \in (R \cup D)} x_{ji} \; ; \; \forall \, j \in (R \cup D)$ (1.5)

$\sum_{j \in R} x_{ij} \leq K \; ; \; \forall \, i \in D$ (1.6)

$Z \geq r_i \; ; \; \forall \, i \in R$ (1.7)

The objective function (1.1) minimizes the sum of the maximum restoration time and the total weighted wait time of the entire problem. α_1 and α_2 are weights which show the relative importance of restoration time and total weighted wait time. Relationship between α_1 and α_2: $(\alpha_1 + \alpha_2 = 1);\ where, 0 \leq \alpha_1, \alpha_2 \leq 1$.

Constraint (1.2) defines the restoration time of an outage i. If a crew member is going from outage i to j, constraint (1.3) ensures that the arrival time at outage j becomes greater than or equal to the arrival time at outage i, plus the time spent to fix outage i, and the time to travel from outage i to j. If a crew member is not going from outage i to j, the big M ensures that the condition becomes always true eliminating that constraint. Constraint (1.4) ensures that all the outages are visited exactly once and crew members return to the depot after visiting few outages. Constraint (1.5) is the flow balance equation, which ensures that a crew member visiting an outage leaves that outage. Constraint (1.6) ensures that the total number of crews assigned to work does not exceed the number of crews available. And finally, constraint (1.7) is a min-max constraint which defines the overall maximum restoration time and is minimized in the objective function.

The objective function described in the proposed MILP model contains two different objectives which sometimes may contradict each other. This kind of problem is regarded as multi-objective optimization and multi-criteria optimization in the literature. As mentioned by (Demir, Bektas, & Laporte, 2014), the kind of objective function proposed in this method needs to be solved with preference based method. In this method, the preferences of the objectives are sought from the decision maker. These preferences vary from one decision maker to another. Hence, it is hard to estimate values of α_1 and α_2. Knowing the preference of the decision maker, the proposed MILP model can achieve the optimum solution, but the methods to estimate the preferences of the decision maker is not considered in the scope of this thesis, and needs further investigation.

3.2 Multi-Stage Metaheuristic method

The proposed MILP model was solved by Gurobi 6.5.2 solver on a 64-bit personal computer with an intel(R) Core(TM) i5 processor having 6.00 GB RAM and 3.20 GHz CPU clock rate, and it was only able to solve maximum of 11 outages in a reasonable amount of time (4612.25 seconds). For 12 outages, the model was run for 24 hours, but it could not produce the final results. Therefore, to solve medium (100 outages) and larger (500 outages) instances in less computational time, the problem is solved using approximation algorithms. For this, an approach using the combination of heuristic and metaheuristics is developed. The method can be divided mainly into three stages according to the algorithm used in each stage- (i) Clustering stage (ii) Initial routing stage (iii) Improvement stage. The idea of dividing this method in above three cases is based on the one described by (Ekici & Retharekar, 2013). As shown in Figure 2, in the first stage, the targets are split into *K* clusters, where *K* is the available number of PLTs. In the second stage, a single agent VRP problem is solved for each cluster. In the third stage, several improvement methods are iteratively implemented to improve the objective functions.

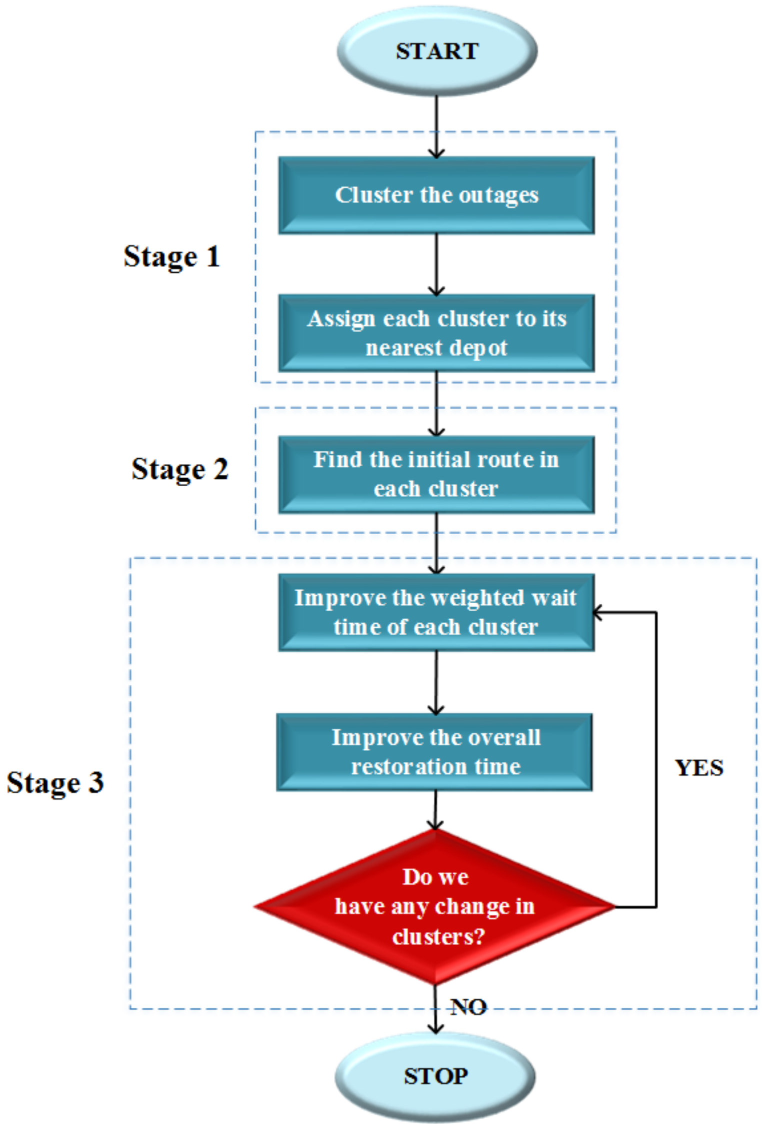

START
Cluster the outages
Stage 1
Assign each cluster to its nearest depot
Stage 2
Find the initial route in each cluster
Improve the weighted wait time of each cluster
Improve the overall restoration time
Stage 3
YES
Do we have any change in clusters?
NO
STOP

Figure 2. Overview of the Approximate method

3.2.1 Clustering stage

Clustering is simply the task of grouping a set of objects such that the objects in the same group (cluster) are somewhat more similar to each other than the objects in other clusters. In this thesis, we cluster the outages based on their proximity to each other. The outages that are nearer to each other are grouped together to form a cluster. The purpose of this stage is to break the problem under investigation into smaller sub-problems. We assume one PLT will fix all the outages of one cluster and then solve a single-agent problem for each cluster, rather than a multi-agent problem for all the outages together.

The outages are clustered using the *k-means clustering algorithm.* k-means clustering is a well-known algorithm which uses iterative refinement to generate a final solution. The algorithm aims to partition N objects into k clusters. The objective of this algorithm is to minimize the sum of distance functions of each point in the cluster to the centre of that cluster (Hartigan & Wong, 1979). The inputs for the algorithm are the data set and the number of clusters k. In our case, the data set is the set of locations of the outages and k is the number of PLTs available.

As shown in Figure 3, the algorithm starts with inserting k number of centroids at random positions within the plane of the outage locations. In the next step, each outage is assigned to its nearest centroid to form k groups, which are known as clusters. After forming k clusters, the mean position

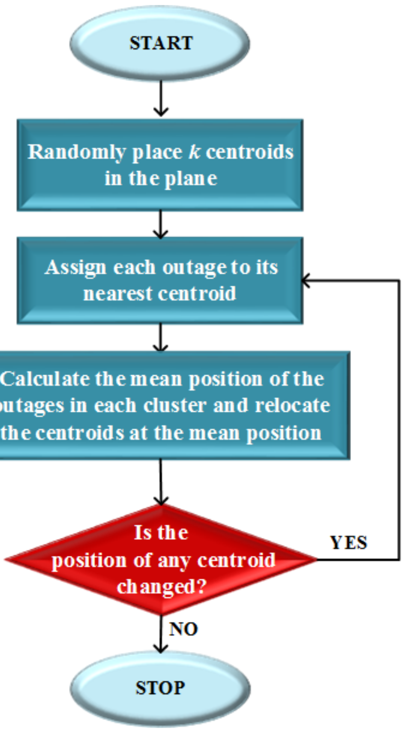
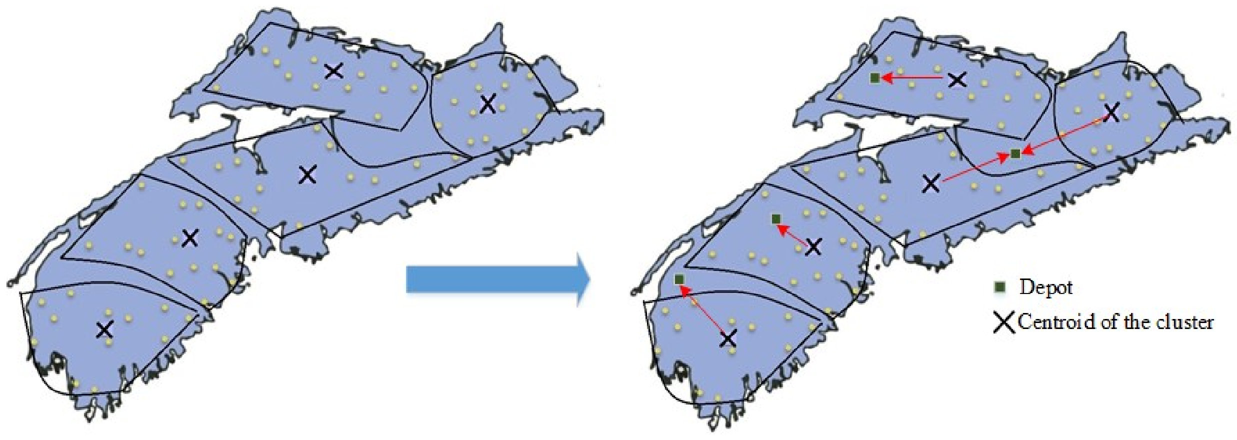

START
Randomly place k centroids in the plane
Assign each outage to its nearest centroid
Calculate the mean position of the outages in each cluster and relocate the centroids at the mean position
Is the position of any centroid changed?
YES
NO
STOP

Figure 3. k-means clustering algorithm

of the outages (mean of the coordinates of outages) in each cluster is calculated, and the centroid of that cluster is relocated at that mean position. As the centroids' positions change, the distance of the outages from the centroids will also change. Therefore, the outages are reassigned to their nearest centroid forming potentially different clusters. The same procedure is repeated until there is no change in any of the clusters.

At the end of the *k*-means clustering, we have *k* clusters (equal to the number of PLTs) as shown in Figure 4(a), each cluster is then assigned to the nearest depot from its centroid as shown in Figure 4(b). This identifies how many PLT are required at each depot. In this process, we may have some depots to which no cluster is assigned.



(a) Final clusters after *k*-means clustering

(b) Assignment of the clusters to the nearest depot from their centroid

Figure 4. Assignment of clusters to depots

3.2.2 Initial routing stage

After identifying the clusters which group the outages and assigning one PLT to each cluster, the initial route by which PLT will visit and restore each outage has to be identified. In this stage, we introduced an *Initial Routing Heuristic* which works greedily to construct the initial route of each PLT.

Erkut & Zhang (1996) developed a penalty based heuristic to solve MCPTDR in which the cost-per-unit time of not visiting a node at any given point in that tour is calculated and the next node is selected based on its cost. Using the same concept, Ekici & Retharekar (2013) proposed a Total Loss-Based Heuristic in which the total loss in rewards of other nodes is calculated if we choose to visit a node, say node *j*, next. Based on this loss, the node providing minimum total loss is selected to be visited next and added to the partial tour *P*. In this thesis, the Total Loss-Based Heuristic is modified to propose an Initial Routing Heuristic.

The initial routing heuristic is a construction heuristic which constructs the initial route of each PLT. This heuristic selects the next location to visit based on the total increase in weighted wait time if we choose to visit that location instead of others at any given point in the tour. To understand it clearly, let's consider one instance where q is the last outage visited in the partial tour (yet to be fixed), and we have two more outages to visit- j and k. The time to fix the outage j and k are v_j and v_k respectively, whereas the number of customers (weight) associated with the outages j and k are s_j and s_k respectively. The time to travel from any outage j to another outage k can be denoted by d_{jk}. After fixing outage q, if we visit outage k, the weighted wait time at outage k becomes $s_k(d_{qk} + v_k)$ and if we visit outage j before k then the weighted wait time at outage k becomes $s_k(d_{qj} + v_j + d_{jk} + v_k)$. Therefore, after visiting outage q, the increase in the weighted wait time to visit outage j instead of k will be $\{s_k(d_{qj} + v_j + d_{jk} + v_k) - s_k(d_{qk} + v_k)\} = s_k(d_{qj} + v_j +$

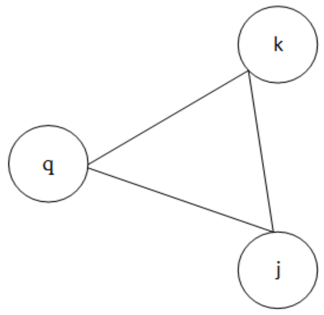


Figure 5. Initial Routing Heuristic

$d_{jk} - d_{qk}\big)$. Similarly, the increase in weighted wait time to visit outage k before j is calculated $\big(s_j\big(d_{qk} + v_k + d_{kj} - d_{qj}\big)\big)$, and the outage providing minimum increase in weighted wait time is selected to be visited after q.

Now, instead of just two outages, suppose we have a set of outages $V = \{j, k, l, m\}$ to visit after q. If the outage j is visited next, then the total increase in weighted wait time is equal to $\sum_{n \in V} s_n\big(d_{qj} + v_j + d_{jn} - d_{qn}\big)$. Similarly, the total increase in weighted wait time for visiting each outage in set V is calculated and the outage producing a minimum increase in weighted wait time is selected to visit after node q. Suppose outage m is selected to be visited next. In the next stage, outage m is removed from set V and added to the partial tour after q, and the next outage is selected using the same method from outage m. The heuristic continues until all the outages from set V are visited. Following is the pseudo code for the same algorithm:

```
**ALGORITHM 1:** The Initial Routing Heuristic
**Input:** Partial tour $T = \{0\}$, currently at outage $q = 0$, set of unvisited outages $V = \{1,2,\dots,m\}$,
time = 0, weighted_wait_time = 0
**while** $V \neq \emptyset$ **do**
    **for** $j \in V$ **do**
        $increase_j = \sum_{n \in V-\{j\}} s_n\big(d_{qj} + v_j + d_{jn} - d_{qn}\big)$
    **end for**
    $l = argmin_{j \in V}\{increase_j\}$
    $V = V - \{l\}$
    arrival_time = time + $v_q$ + $d_{ql}$
    weighted_wait_time = weighted_wait_time + (arrival_time + $v_l$ ) $s_l$
    $T = T \cup \{l\}$
    $q = l$
    time = arrival_time
**end while**
```

Suppose that Figure 6(a) is an outcome of the clustering stage, in which three clusters A, B, C (generated from *k*-means clustering algorithm) are assigned to a depot D. Then, Figure 6(b) shows hypothetical routes generated by an initial routing heuristic in each of the clusters. By following

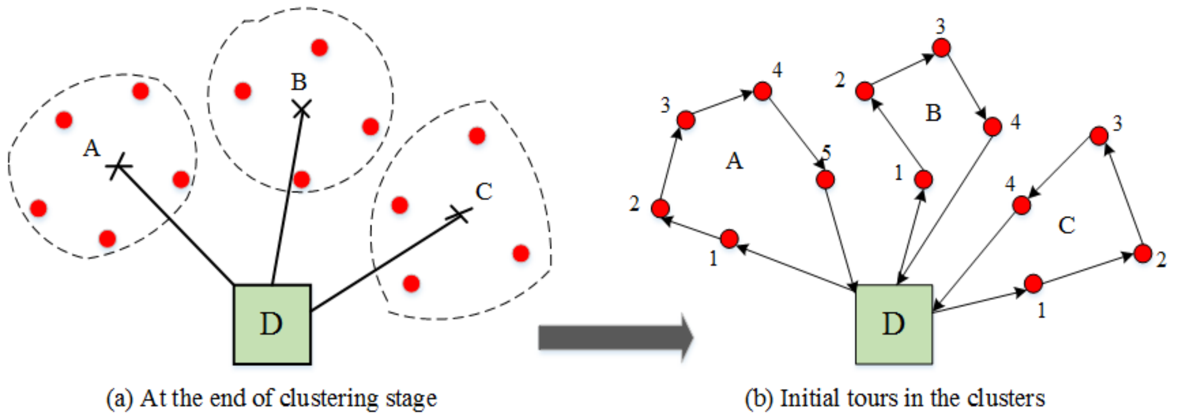

A
B
C
D
(a) At the end of clustering stage
3
4
2
3
B
4
A
5
1
3
2
4
C
1
2
D
1
2
(b) Initial tours in the clusters

Figure 6. Initial tours in clusters

the same method, the routes in all the clusters are constructed using an initial routing heuristic. These routes are actually the order in which PLTs will fix the outages minimizing the total weighted wait time of their cluster.

3.2.3 Improvement stage

As mentioned earlier, the initial routing heuristic is applied to construct the initial route in each cluster. However, the outcome of this initial solution may be far from an optimal solution and needs to be improved. In this stage, the initial solution generated by the initial routing heuristic is improved to minimize the overall restoration time as well as total weighted wait time. This is achieved by two methods- (i) Intra-route improvement (ii) Inter-route improvement

3.2.3.1 Intra-route improvement

The proposed initial routing heuristic works greedily to construct the initial routes in each cluster. Any greedy algorithm makes the locally optimum choice at each stage with the hope of finding a solution close to the global optimum at the end. It produces a good solution from which we can start the improvement. Therefore, before moving any further, the routes generated by the initial routing heuristic needs to be improved in order to minimize the total weighted wait time of each cluster. In this section, a method is described to improve these routes using a *tabu search metaheuristic* with the combination of *sub-tour reversal* technique.

The sub-tour reversal is a well-known technique which generates new solutions by reversing the partial tours of the original solution. Since these new solutions are generated from the original solution, they are known as the neighbors of the original solution, and a set of all the neighbors is

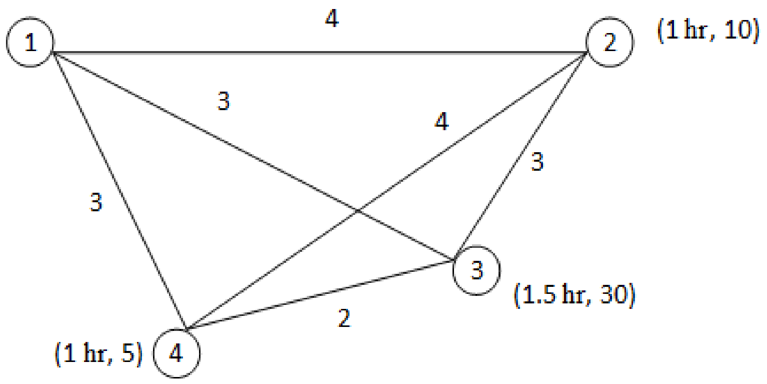


Figure 7. Sub-tour reversal example

called the neighborhood of the original solution. Let us consider an example shown in Figure 7, in which each node denotes an outage and the time to travel between any two nodes is denoted on the arcs, whereas a pair of (time to fix the outage, and the number of customers associated with each outage) is denoted at respective nodes. Node 1 is the depot. As shown in Figure 8(a), let's assume that 1-2-3-4-1 is an initial solution obtained from the proposed initial routing heuristic as mentioned earlier. By reversing nodes 2 and 3 in the initial solution, the sub-tour reversal generates a new solution (1-3-2-4-1) as shown in Figure 8(b). Similarly, by reversing nodes (3, 4) and (2, 3, 4) in the initial solution, the sub-tour reversal generates two other neighbors 1-2-4-3-1 (Figure 8(c)) and 1-4-3-2-1 (Figure 8(d)), respectively. The process is repeated until all the possible partial tours are reversed. After calculating the objective value of each neighbor, which is a weighted wait time in this case, the neighbor with the least objective value is selected. For our example, there are only three possible neighbors and we can see that 1-3-2-4-1 (Figure 8(b)) has the least objective value among all the neighbors including the initial solution. In this way, the sub-tour reversal results in an improved solution from initial solution. This concept of sub-tour reversal is used in

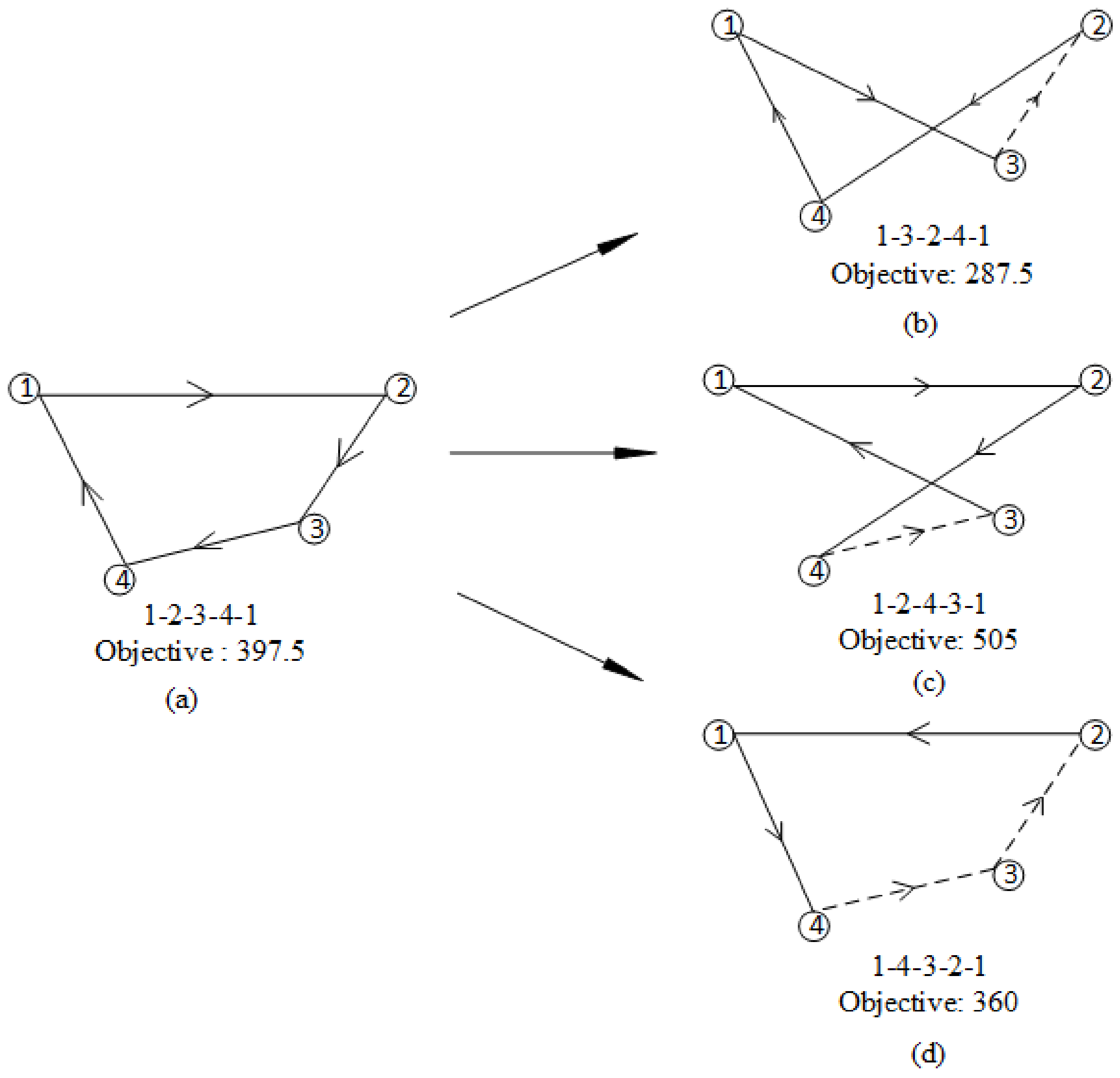


Figure 8. Neighborhood of current solution after sub-tour reversal

the tabu search metaheuristic which eventually improves the weighted wait time of each tour in the problem.

Tabu search is a metaheuristic technique which provides a better chance to escape from a local optimum and reach the global optimum by accepting non-improving solutions in addition to the improving solutions. Moreover, it records the recently visited solution and *prohibits* (hence the term *tabu*) the search returning to previously visited solutions for a certain number of iterations. The recently visited solutions are stored in a list called the *tabu* list. The size of the tabu list is defined by the user based on problem characteristics. When the list gets full, the metaheuristic starts to omit the oldest solution in the list to add a newly visited solution. Tabu search iteratively moves from one potential solution to an approved solution in its neighborhood until some user specified stopping criterion is met. Two examples of the stopping criteria are CPU runtime and the maximum number of consecutive non-improving solutions.

For the problem under investigation, the tabu search metaheuristic initiates from the solution obtained after the initial routing heuristic. As mentioned in the flowchart (Figure 9), all the neighbors of the initial solution are obtained using the sub-tour reversal technique and the best one is selected. If the objective value of that neighbor is the best (minimum) value found so far, it is immediately updated as the next trial solution. In case if that neighbor is not the best solution found so far by the metaheuristic, the tabu list is checked. If that neighbor is present in the tabu list, the next best neighbor not in the tabu list is selected from the neighborhood and it becomes the trial (current) solution for the next iteration and the tabu list is updated. If the selected neighbor's objective value is worse than that of the current solution, the counter of consecutive non-improving solutions is increased by 1, otherwise, the counter is reset to zero. As soon as the tabu list's size

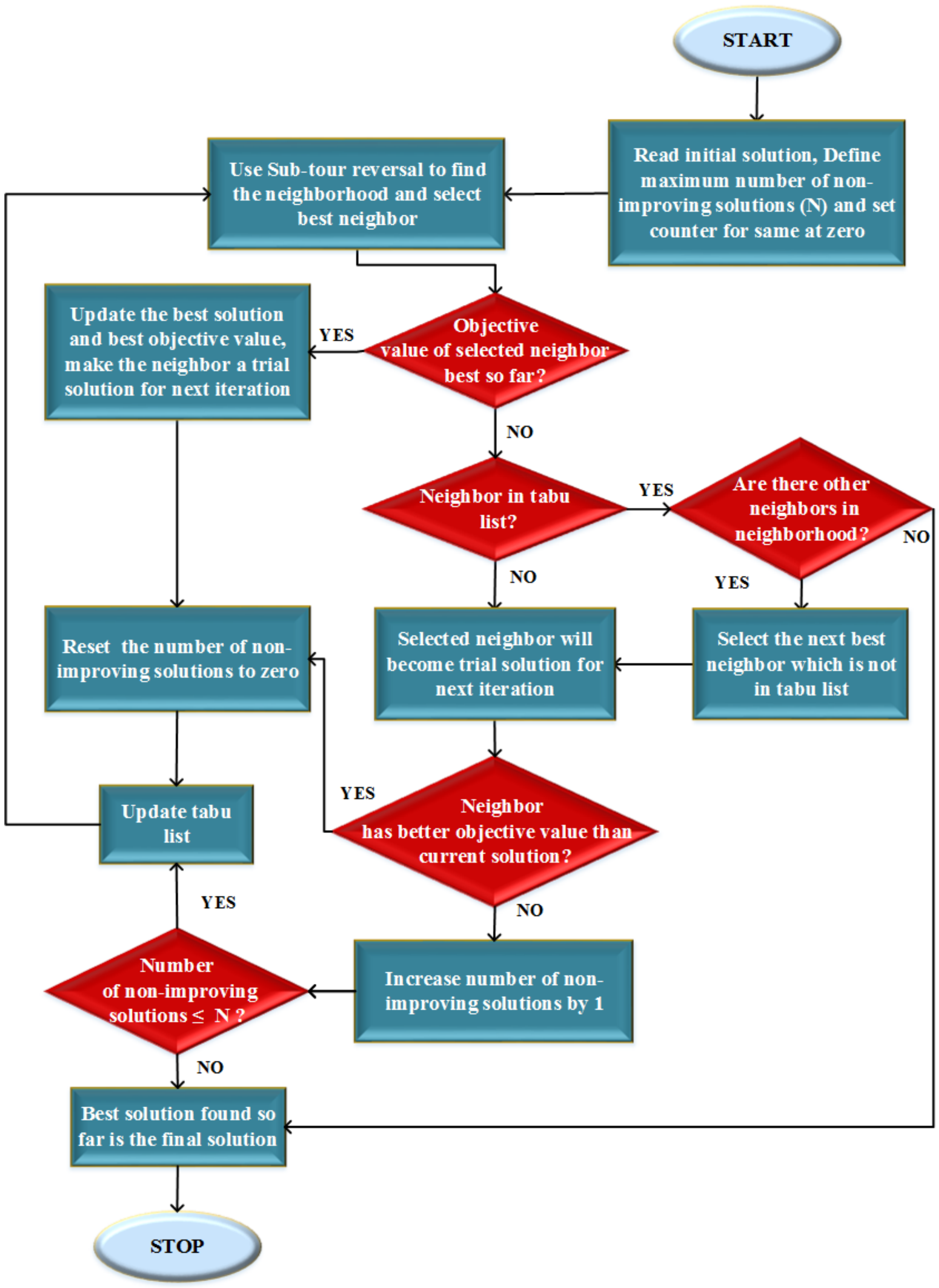

START
Read initial solution, Define maximum number of non-improving solutions (N) and set counter for same at zero
Use Sub-tour reversal to find the neighborhood and select best neighbor
Objective value of selected neighbor best so far?
YES
Update the best solution and best objective value, make the neighbor a trial solution for next iteration
NO
Neighbor in tabu list?
YES
Are there other neighbors in neighborhood?
NO
YES
Select the next best neighbor which is not in tabu list
NO
Selected neighbor will become trial solution for next iteration
Reset the number of non-improving solutions to zero
YES
Neighbor has better objective value than current solution?
NO
Update tabu list
YES
Increase number of non-improving solutions by 1
Number of non-improving solutions ≤ N ?
NO
Best solution found so far is the final solution
STOP

Figure 9. Tabu search metaheuristic for intra-route improvement

reaches to the maximum limit defined by the user, the oldest added solution is omitted from the list with each addition of a new solution. The same process continues until the counter for consecutive non-improving solutions becomes greater than that defined by the user, say N, or the search process runs out of neighbors which are not in the tabu list. At the end of the process, the best solution found is selected as the final solution.

This metaheuristic is used as one step in the inter-route metaheuristic discussed in the next section.

3.2.3.2 Inter-route improvement

The metaheuristic mentioned in section 3.2.3.1 improves the total weighted wait time (secondary objective) by improving the weighted wait time of each cluster. To minimize the overall (maximum) restoration time of all customers (primary objective), the tabu search metaheuristic is again used for inter-route improvement, but with a different tactic. In contrast to the intra-route improvement phase, which works with only one tour (cluster) at a time, the inter-route improvement works with all the tours simultaneously. After clustering the outages and obtaining an initial tour (route) of each of the clusters using the initial routing heuristic, all the tours in the solution are once improved using the tabu search metaheuristic mentioned in section 3.2.3.1. After that, two improvement steps are implemented iteratively to minimize the maximum restoration time of all tours: (i) transfer, and (ii) insertion. These two improvement steps follow the idea proposed by (Ekici & Retharekar, 2013). As the names suggest, the transfer and insertion steps *transfer* an outage from one cluster and *insert* it into another cluster at the most suitable position according to the desired objective. Both steps are discussed in detail in the following paragraph.

The process of tabu search metaheuristic for inter-route improvement applied to the problem under investigation is shown in the flowchart (Figure 13). In the beginning of this metaheuristic, the improved solution obtained after applying tabu-search for intra-route improvement (section

3.2.3.1) is set as the current trial solution for inter-route improvement. Initially, this trial solution is assumed as the best solution and its objective is assumed as the best objective. The counter for a consecutive non-improving solution is set to zero.

As the objective of this metaheuristic is to minimize (improve) the maximum restoration time of the entire solution, the restoration times of all the current routes (tours) are calculated, and the tour having the highest restoration time is selected to transfer (donate) an outage to one of the other tours (including the tours assigned to other depots). Let's assume that Figure 10 shows the initial tours obtained in three clusters (A, B, C) after applying tabu-search for intra-route improvement

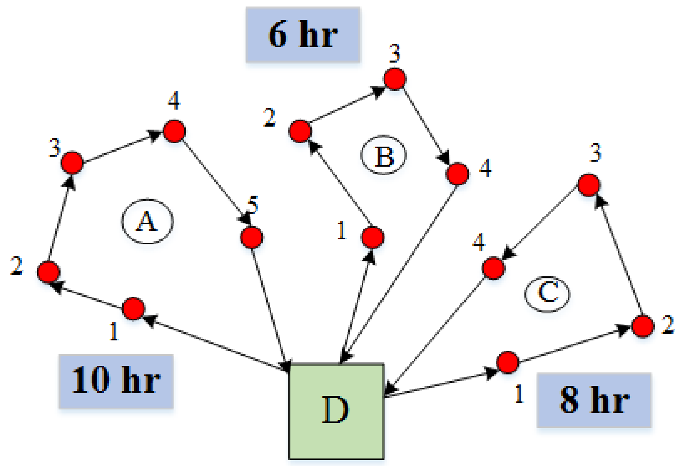


Figure 10. Selection of donor to transfer

(section 3.2.3.1). The resulting restoration times are mentioned near respective tours. Among all three clusters, as cluster-A has maximum restoration time of 10 hours, it will be selected as a donor. Say outage-5 is the first outage to be considered for transfer from cluster-A to cluster-B or cluster-C.

After the selection of the donor, the selected outage is inserted in all the other tours at the most suitable position such that the weighted wait time of that tour is minimized. This process of insertion is demonstrated in Figure 11. Outage-5 is transferred from cluster-A (which has the longest restoration time) and inserted in all the positions in cluster-B. All potential resulting insertion into cluster-B are demonstrated in Figure 11. The weighted wait time of cluster-B after each insertion is indicated above the respective figures. In the example, Figure 11(a) (the insertion of outage-5 between outage-1 and outage-2) shows the least weighted wait time of cluster-B. Therefore, that position is selected as the best insertion for outage-5 in cluster-B. Similarly, the best position for outage-5 in cluster-C can also be obtained.

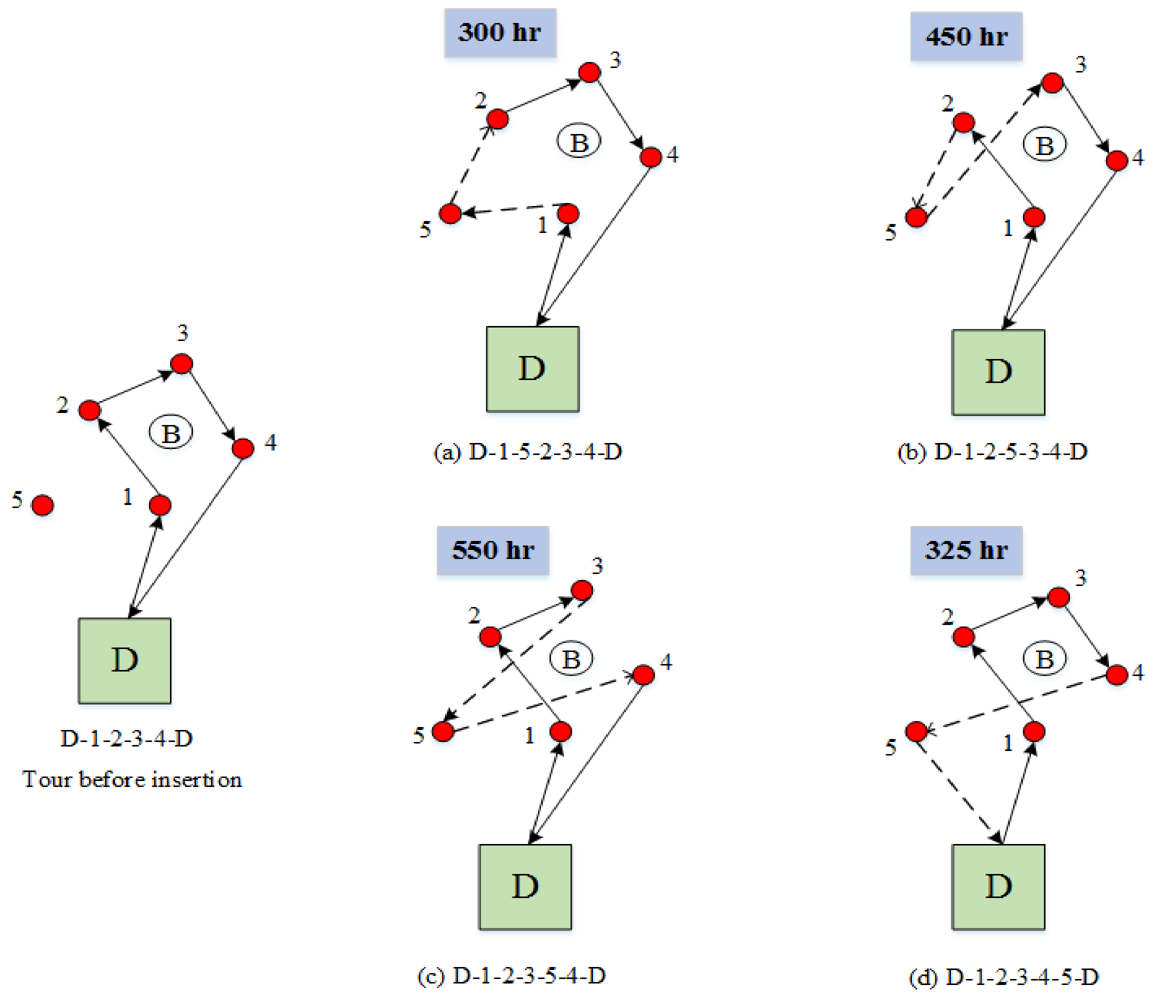
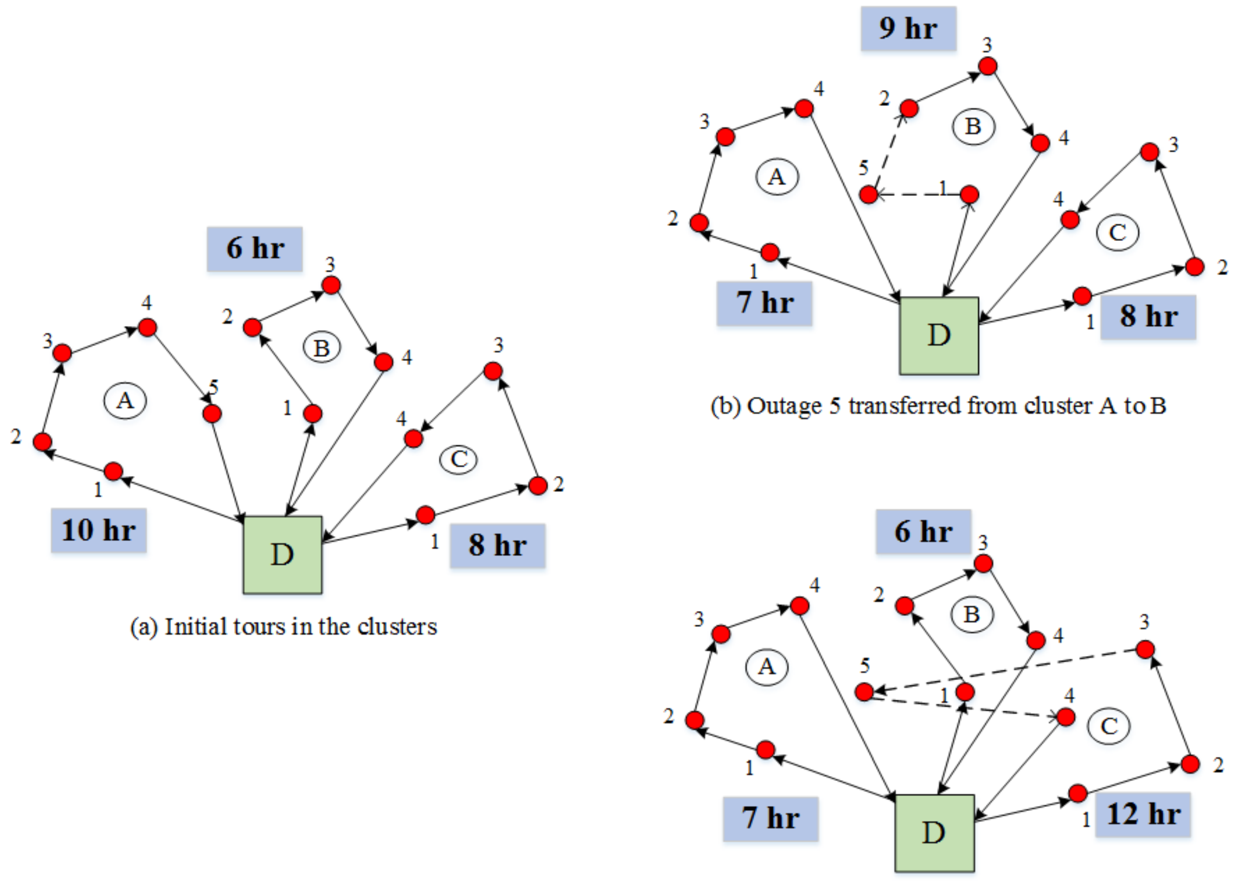


Figure 11. The transferred outage from the donor is inserted at all the places in another cluster

After inserting the outage at the best position in all clusters (tours), the cluster producing the least overall restoration time of the solution is selected as the *receiver*. The resultant solution is regarded as one neighbor of the current solution. As exemplified in Figure 12(b) and Figure 12(c), if outage-5 is transferred to cluster-B and cluster-C, the resultant maximum restoration time becomes 9 hours and 12 hours for each solution, respectively. Hence, cluster-B is selected as the receiver of outage-5. These transfer and insertion steps are tested for all the outages in the donor (cluster-A), and the transfer- insertion resulting in lowest overall restoration time is selected as the best neighbor of the current solution. The respective resultant solutions obtained after transferring each of the outages of the donor define the neighborhood of the current solution. As in the current solution



(a) Initial tours in the clusters

(b) Outage 5 transferred from cluster A to B

(c) Outage 5 transferred from cluster A to B

Figure 12. Transfer of an outage from one cluster to others

(Figure 12(a)), the donor (cluster-A) has five outages, the current solution will have five neighbors in the neighborhood.

The weighted wait time of each tour of the selected best neighbor will then be improved using the *intra-route tabu search metaheuristic* mentioned in section 3.2.3.1.

One important thing to notice here is the intra-route tabu search does not consider the restoration time at all. Therefore, even if it improves the weighted wait time, it does not guarantee that the overall restoration time will also improve. It may rather increase the overall restoration time in some cases. But the primary objective (overall restoration time) will be addressed in the transfer step as well as the selection process of the neighbors, which will alleviate the potential compromise made by the metaheuristic for intra-route improvement.

If the overall restoration time of the selected neighbor is the best found so far, it immediately becomes the trial (current) solution for the next iteration, the counter for consecutive non-improving solutions is reset to zero, and the tabu list is updated.

The purpose of the tabu list is to restrict the transfer of an outage back to the same tour from which it was transferred a few iterations ago (tabu move). The pair of indices of the *donor* and the transferred outage is stored in the tabu list, which is counted as one tabu move. If the candidate neighbor is not the best one found so far, the indices of the *receiver* tour and the received outage are checked with the tabu moves in the tabu list. If that move is restricted by the tabu list, the next best neighbor having minimum overall restoration time whose move is not in the tabu list is selected. The candidate neighbor becomes a trial (current) solution for the next iteration and the tabu list is updated. If the selected neighbor has a better value of overall restoration time than the current solution, the counter for a consecutive non-improving solution is reset to zero. Otherwise, the counter is increased by one. The entire process of transfer, insertion, and selection of neighbor is repeated until the counter exceeds the predefined number of consecutive non-improving solutions (N), or the search process runs out of neighbors whose transfer move is not restricted by the tabu list. The best solution found by the metaheuristic so far is selected as the final solution of the problem. Figure 13 shows the entire method for inter-route improvement mentioned in this section.

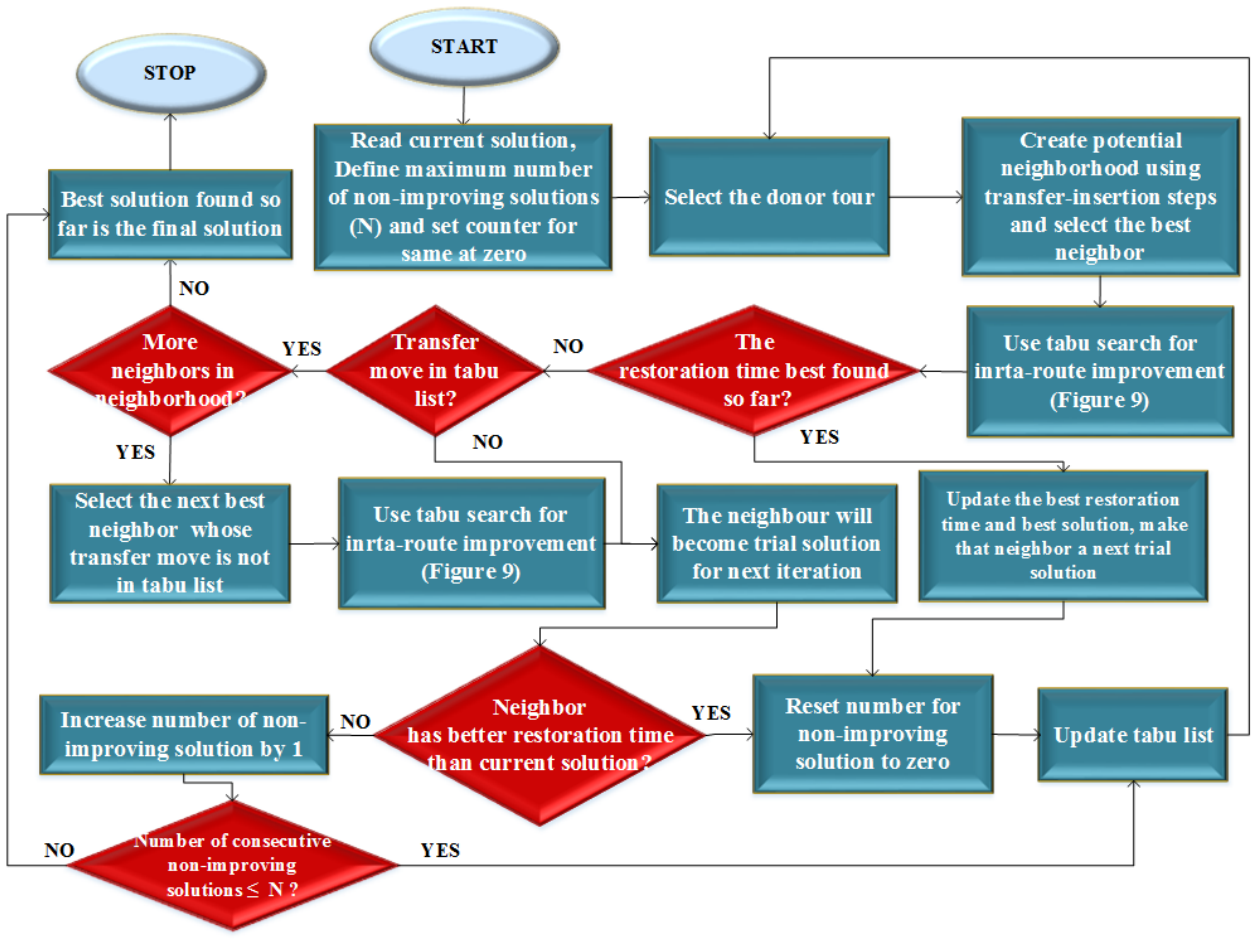

STOP
START
Read current solution, Define maximum number of non-improving solutions (N) and set counter for same at zero
Select the donor tour
Create potential neighborhood using transfer-insertion steps and select the best neighbor
Best solution found so far is the final solution
NO
More neighbors in neighborhood?
YES
Transfer move in tabu list?
NO
The restoration time best found so far?
Use tabu search for inrta-route improvement (Figure 9)
YES
NO
YES
Select the next best neighbor whose transfer move is not in tabu list
Use tabu search for inrta-route improvement (Figure 9)
The neighbour will become trial solution for next iteration
Update the best restoration time and best solution, make that neighbor a next trial solution
Increase number of non-improving solution by 1
NO
Neighbor has better restoration time than current solution?
YES
Reset number for non-improving solution to zero
Update tabu list
NO
Number of consecutive non-improving solutions ≤ N ?
YES

Figure 13. Tabu search metaheuristic for inter-route improvement

3.3 Hybrid method

This method combines the metaheuristic mentioned in section 3.2 with the optimum method (Linear Programming (LP)) for sections of the multi-stage metaheuristic approach to improve the solution. In this method, we use Integer Linear Programming (ILP) to obtain optimum clusters for the problem as well as to find the optimum route in each cluster minimizing the total weighted wait time. The method follows the same three stages as mentioned in section 3.2: (i) clustering stage (ii) Initial routing stage (iii) Improvement stage, but finds an optimum solution at some stages. Because of the optimum solutions achieved in some stages, this method could potentially provide a better chance to reach to a global optimum for the final solution.

3.3.1 Clustering stage alternative

The hybrid method starts with finding the initial clusters using the k-means clustering algorithm as mentioned in section 3.2.1. Since the k-means clustering algorithm does not consider the restoration time or weighted wait time, and only considers the closeness of the outages, it may create an initial solution which is far from optimality, requiring much more computation to improve by heuristic and metaheuristic methods. It may also affect the quality of the final result.

To address this shortcoming, a mathematical model is developed using Integer Linear Programming (ILP) with an objective of allocating a relatively equal number of outages to each cluster. This ILP model initiates from the final centroids obtained after k-means clustering. Let us consider an instance in which we need to cluster n outages into m clusters.

3.3.1.1 Sets and Parameters

centroids	Set of the centroids of each cluster, $\{1,2,\dots,m\}$
outages	Set of the outages, $\{1,2,\dots n\}$

d_{ij}	Distance between centroid i and outage j

3.3.1.2 Decision variable

$$x_{ij} = \begin{cases} 1, if\ outage\ j\ is\ assigned\ to\ cluster\ i \\ \quad 0, otherwise \end{cases}$$

3.3.1.3 Objective function

Minimize: $\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}\, x_{ij}$ (2.1)

3.3.1.4 Constraints

Subjected to:

$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{kj} \leq 1;\ \forall\ i \neq k;\ \forall\, i, k \in$ centroids (2.2)

$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{kj} \geq (-1);\ \forall\ i \neq k;\ \forall\, i, k \in$ centroids (2.3)

$\sum_{i=1}^{m} x_{ij} = 1;\ \forall\, j \in$ outages (2.4)

Objective function (2.1) minimizes the total distance between the outages and the centroid of the cluster to which they are assigned. Constraints (2.2) and (2.3) ensures that the difference between a number of outages in any two clusters is not more than one, which in turn distributes an almost equal number of outages to each cluster. Constraint (2.4) ensures that each outage is assigned to exactly one cluster.

This ILP model is iteratively used to obtain optimum clusters (see Figure 14). After each iteration, the centroids are relocated to the mean position of the current outages in their respective clusters. These positions are selected as initial positions of the centroids for the next iteration. The process is repeated until the optimum combination of clusters is achieved. By doing so, we may add those outages to the cluster, which is relatively far from the rest of the outages in that cluster, but it will be taken care of in upcoming stages.

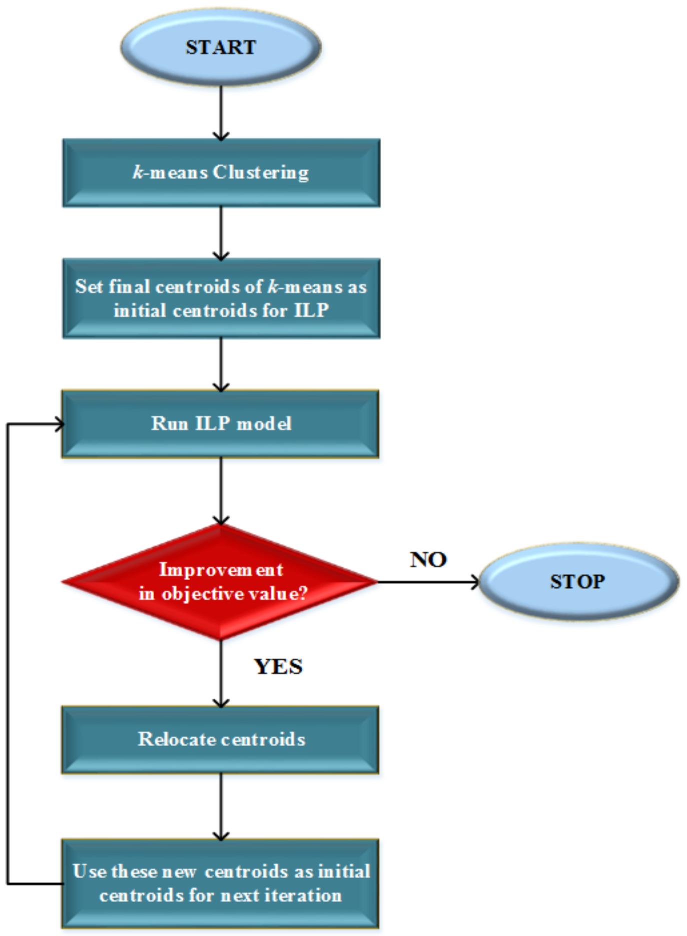

START
k-means Clustering
Set final centroids of *k*-means as initial centroids for ILP
Run ILP model
Improvement in objective value?
NO
STOP
YES
Relocate centroids
Use these new centroids as initial centroids for next iteration

Figure 14. Hybrid clustering

By following the same procedure of assignment of clusters to depots mentioned in section 3.2.1, at the end of this stage, each cluster is assigned to its nearest depot from its centroid and one PLT is assigned to each cluster.

3.3.2 Initial routing alternative

After obtaining optimum clusters from the previous stage, in this stage, an *optimum* route in each cluster is obtained using the MILP model, in contrast to finding *good* initial routes using the initial routing heuristic mentioned section 3.2.2. For this purpose, the mathematical model mentioned in section 3.1 can be modified to minimize only the total weighted wait time. As this model is used to solve only one cluster at a time, it has only one depot and a single crew member. All the constraints, as well as the objective function of this model, are linear.

Assume that we have a cluster having n number of outages and we want to find the optimum route of a PLT who will start his journey from the depot to which the cluster is assigned, visit all the outages in the cluster, and finally end his journey at the same depot.

3.3.2.1 Sets and Parameters

D	Depot
R	Set of all the outages, $\{1, \dots, n\}$
s_i	Number of customers (weight) associated with outage i
v_i	Service time to fix an outage i
d_{ij}	Time to travel between outage i to j

3.3.2.2 Decision Variables

$$x_{ij} = \begin{cases} 1, if\ outage\ j\ is\ visited\ right\ after\ outage\ i \\ 0,\ Otherwise \end{cases}$$

$t_i \geq 0$ Arrival time at outage i

3.3.2.3 Objective function

Minimize: $\sum_{i \in R} s_i(t_i + v_i)$ (3.1)

3.3.2.4 Constraints

Subject to:

$$t_j \geq t_i + v_i + d_{ij} - M(1 - x_{ij}) \; ; \; \forall \, i \in (R \cup D), j \in R \tag{3.2}$$

$$\textstyle\sum_{j \in (R \cup D)} x_{ij} = 1 \; ; \; \forall \, i \in (R \cup D) \tag{3.3}$$

$$\textstyle\sum_{i \in (R \cup D)} x_{ij} = \sum_{i \in (R \cup D)} x_{ji} \; ; \; \forall \, j \in (R \cup D) \tag{3.4}$$

The objective function (3.1) minimizes the total weighted wait time of the cluster. If outage j is visited right after outage i, constraint (3.2) ensures that the arrival time at outage j becomes greater than or equal to the sum of the arrival time at outage i, time spent to fix the outage i, and the time to travel from outage i to j. Constraint (3.3) ensures that all the outages are visited exactly once and a crew member can only return to the depot after visiting all the outages. In contrast to the MILP model proposed in section 3.1, in this case, as the crew member has to visit *all* the outages of his cluster before returning to the depot, the range of i is $(R \cup D)$ instead of (R). And finally, constraint (3.4) ensures that a crew member entering a location leaves that location.

At the end of this stage, we will have tours having optimum weighted wait times. But as mentioned in section 1.1, because of the contradictory behavior of total weighted wait time and restoration time in some cases, it does not guarantee that the clusters will always have the optimum restoration times as well. Therefore, the maximum restoration time will be improved in the improvement stage.

3.3.3 Improvement stage alternative

In this stage, the same tabu search metaheuristic for *inter-route improvement* is used as mentioned in section 3.2.3.2. But instead of using tabu search metaheuristic for *intra-route improvement* (section 3.2.3.1), the MILP model, mentioned in section 3.3.2, is used in each iteration to find an optimal tour of the clusters. The same transfer and insertion steps are used as mentioned in section 3.2.3.2. The transfer and insertion steps will alleviate the compromise made in restoration time by MILP model.

3.4 Summary of the proposed methodologies

Methodology	Multi-Stage Metaheuristic	Hybrid	Optimum
Clustering	*k* -means clustering	*k* -means clustering + ILP	MILP
Initial routing	Initial Routing Heuristic	MILP	
Intra-route improvement	Tabu search + Subtour reversal	MILP	
Inter-route improvement	Tabu search + Transfer-Insertion	Tabu search + Transfer-Insertion	

Table 1. Summary of the proposed methodologies

Chapter 4 Results

In chapter 3, the MILP models and heuristic, as well as metaheuristic algorithms, are proposed to solve the problem. In this chapter, the simulation process used to generate the data and the results obtained from several instances are discussed. The simulation process is written in Python language and the MILP/ILP models are implemented in GLPK 0.2.18 and solved by GUROBI 6.5.2. The heuristic and metaheuristics are written and conducted in Python 2.7.10. All the software are operated on a 64-bit personal computer with an intel(R) Core(TM) i5 processor having 6.00 GB RAM and 3.20 GHz CPU clock rate.

4.1 Data Simulation

The simulation process requires the number of depots, number of outages and the number of available PLTs as input. It is assumed that the depots are responsible for maintaining the power grids situated within the radius of 25 kilometers from their position. Therefore, the maximum length and maximum width of the plane are assumed to be $(25 * number\ of\ depots)$. The square plane is generated in the first quadrant, from the origin (0, 0) having sides equal to the maximum length. In the real-life scenario, when the storm passes from one area, it creates a more disastrous effect in the areas nearer to its centre compared to those far from its centre. Therefore, a centre of the storm is randomly selected within that plane and the number of overall outages defined by the user is distributed such that the depots nearer to the centre of the storm have more number of outages than those situated far from the centre. The distributed number of outages to each depot are randomly generated within the radius of 25 kilometers from their respective depots.

The time to fix each of the outages and the number of customers associated with each outage are generated based on the historical data provided by a power company (name withheld). Based on the data, the time to fix an outage is kept within the range of 1 hour to 3 hours. To generate the number of customers associated with each outage, the probabilistic sampling technique is used. After thoroughly observing the data, the number of customers are divided into the ranges of (5 to 50), (51 to 100), (101 to 250), (251 to 500), (501 to 1000) and (1000 to 2000). The probability of the number of customers falling in each range is estimated from the power company's data. Based on those probabilities, one range is randomly allocated to each outage. Then a random integer number within the allocated range is generated to indicate the number of customers associated with the outage.

For simplicity reasons, the travel distance is considered as straight-line Euclidian distance. Given the coordinates of any two points (x, y) and (a, b), the Euclidean distance between them can be calculated using equation $\sqrt{(x^2 - a^2) + (y^2 - b^2)}$. The average speed of the vehicle is estimated to be 50 kilometers per hour considering the traffic restrictions and the road conditions. Therefore, the travel time between any two points will be the distance between them divided by the speed of the vehicle. Moreover, it is assumed that it is always possible to travel between all the points. Therefore, there will not be any infeasible tours in the problem.

4.2 Selection of parameters for tabu search metaheuristic

The parameters used for any metaheuristic may influence its performance, which reflects in the solution and computational time. Therefore, before comparing all the methods mentioned in chapter 3, it is important to determine a good set of parameters for the metaheuristics for the different sizes of the problem. There are two parameters needed to be determined for the tabu search metaheuristic used in the methodology: (i) the number of consecutive non-improving iterations, and (ii) the length of tabu-list. Usually, the tabu search metaheuristic having a higher number of non-improving solutions has a better chance of finding good solutions but needs more computational time.

In this thesis, the parameters are determined for different sizes of the problem. The problems are arbitrarily classified in three sizes, i.e. small, medium, and large. For each of the sizes, problems of two sub-sizes are tested as following: for small size (2 depots, 7 outages, 2 PLTs) and (3 depots, 10 outages, 3 PLTs), for medium size (5 depots, 50 outages, 10 PLTs) and (10 depots, 100 outages, 40 PLTs), for large size (31 depots, 500 outages, 140 PLTs) and (31 depots, 600 outages, 140 PLTs). The proportion of depots, outages, and PLTs for large instances are determined from the power company's data. For each of the sub-sizes, five sets of data are generated, and the conclusion is made based on the average value of the primary objective function (maximum restoration time) of the five sets.

4.2.1 Selection of maximum number of consecutive non-improving solutions

To select the maximum number of consecutive non-improving solutions, a certain length of tabu-list was primarily fixed for each size of the problem, and the problem was solved for a different number of consecutive non-improving solutions. The tabu-list length for the small, medium and large size of problems is initially fixed at 5, 7 and 10 respectively, which will be tested later.

4.2.1.1 Multi-Stage Metaheuristic method

SMALL SIZE					Final restoration time (hour)				Final Weighted wait time (hour)				CPU Runtime (second)			
Number of consecutive non-improving solutions					5	7	10	15	5	7	10	15	5	7	10	15
2 Depots	7 Outages	2 Clusters	5 Tabu list length	1	7.27	7.27	7.27	7.27	1074.2	1074.2	1074.2	1074.2	0.07	0.09	0.14	0.29
				2	7.47	7.47	7.47	7.47	1146.8	1146.8	1146.8	1146.8	0.06	0.09	0.16	0.42
				3	10.49	10.26	10.26	10.26	1867.3	1850.0	1850.0	1850.0	0.06	0.25	0.33	0.69
				4	8.37	8.37	8.37	8.37	1858.8	1858.8	1858.8	1858.8	0.11	0.16	0.22	0.55
				5	7.67	7.67	7.67	7.67	1873.5	1873.5	1873.5	1873.5	0.06	0.10	0.14	0.39
				Avg.	8.25	8.21	8.21	8.21	1564.14	1560.69	1560.69	1560.69	0.07	0.14	0.20	0.47
				Std. Dev.	1.32	1.22	1.22	1.22	414.94	411.84	411.84	411.84	0.02	0.07	0.08	0.16
Number of concecutive non-improving solutions					5	7	10	15	5	7	10	15	5	7	10	15
3 Depots	10 Outages	3 Clusters	5 Tabu list length	1	8.65	8.65	8.65	8.65	2040.1	2040.1	2040.1	2040.1	0.14	0.21	0.32	0.51
				2	9.88	9.52	9.52	9.52	2616.2	2637.1	2637.1	2637.1	0.07	0.27	0.38	0.67
				3	8.78	8.78	8.70	8.70	2463.1	2463.1	2524.4	2524.4	0.17	0.22	0.47	0.77
				4	8.10	8.10	8.10	8.10	1863.1	1863.1	1863.1	1863.1	0.12	0.18	0.28	0.49
				5	8.14	7.75	7.75	7.75	1925.3	1647.5	1647.5	1647.5	0.20	0.30	0.45	0.69
				Avg.	8.71	8.56	8.55	8.55	2181.6	2130.2	2142.4	2142.4	0.14	0.23	0.38	0.63
				Std. Dev.	0.72	0.68	0.68	0.68	337.37	412.41	425.48	425.48	0.05	0.05	0.08	0.12
MEDIUM SIZE					Final restoration time (hour)				Final Weighted wait time (hour)				CPU Runtime (second)			
Number of consecutive non-improving solutions					10	15	20	30	10	15	20	30	10	15	20	30
5 Depots	50 Outages	11 PLT	7 Tabu list length	1	11.39	11.39	11.39	11.39	15057.9	15057.9	15057.9	15057.9	2.26	3.72	4.67	7.31
				2	11.62	11.47	11.47	11.47	22448.3	23884.9	23884.9	23884.9	2.39	6.36	7.67	11.52
				3	10.58	10.58	10.58	10.58	22309.4	22309.4	22309.4	22309.4	3.71	4.97	6.68	11.11
				4	11.59	11.59	11.38	11.38	45487.1	45487.1	48683.4	48683.4	4.51	6.63	12.52	18.27
				5	11.96	11.96	11.96	11.96	28863.2	28863.2	28863.2	28863.2	4.49	6.42	9.02	15.42
				Avg.	11.43	11.40	11.36	11.36	26833.2	27120.5	27759.8	27759.8	3.47	5.62	8.11	12.73
				Std. Dev.	0.51	0.50	0.49	0.49	11515.60	11396.15	12699.17	12699.17	1.09	1.25	2.93	4.22
Number of concecutive non-improving solutions					10	15	20	25	10	15	20	25	10	15	20	25
10 Depots	100 Outages	40 PLT	7 Tabu list length	1	6.89	6.85	6.85	6.85	46822.4	48836.7	48836.7	48836.7	20.32	30.66	38.11	36.65
				2	7.08	7.08	7.08	7.08	59636.1	59688.8	59688.8	59688.8	17.85	22.55	26.43	25.56
				3	7.09	7.09	7.09	7.09	38054.2	38054.2	38054.2	38054.2	15.02	16.09	19.20	17.83
				4	6.33	6.33	6.33	6.33	44713.4	44713.4	44713.4	44724.4	15.25	17.97	23.37	18.92
				5	6.71	6.71	6.71	6.71	51699.0	51699.0	51699.0	51699.0	15.11	15.80	19.01	17.66
				Avg.	6.82	6.81	6.81	6.81	48185.01	48598.43	48598.43	48600.64	16.71	20.61	25.23	23.32
				Std. Dev.	0.31	0.31	0.31	0.31	8062.53	8045.82	8045.82	8044.49	2.34	6.23	7.84	8.13
LARGE SIZE					Final restoration time (hour)				Final Weighted wait time (hour)				CPU Runtime (second)			
Number of consecutive non-improving solutions					10	20	30	100	10	20	30	100	10	20	30	100
31 Depots	500 Outages	140 PLT	10 Tabu list length	1	10.86	10.76	10.76	10.8	270029.8	271702.0	271702.0	271702.0	817.4	1017.4	1085.9	2811.9
				2	10.18	10.04	10.04	10.0	257046.0	255289.2	255289.2	255289.2	790.0	1147.8	1233.3	3144.7
				3	11.01	10.56	10.56	10.6	280777.0	290805.4	290805.4	290805.4	545.3	1187.9	1295.7	3831.3
				4	10.36	10.30	10.30	10.3	253448.7	253261.5	253261.5	253261.5	529.1	633.3	662.7	1707.0
				5	10.15	10.15	10.15	10.2	231668.5	231668.5	231668.5	231668.5	670.7	711.5	735.8	1841.4
				Avg.	10.51	10.36	10.36	10.36	258594.0	260545.3	260545.3	260545.3	670.5	939.6	1002.7	2667.2
				Std. Dev.	0.40	0.30	0.30	0.30	18552.07	22105.98	22105.98	22105.98	133.73	253.42	288.47	895.53
Number of concecutive non-improving solutions					10	20	30	100	10	20	30	100	10	20	30	100
31 Depots	600 Outages	140 PLT	10 Tabu list length	1	12.51	12.46	12.46	12.46	378410.8	378682.9	378682.9	378682.9	1613.9	1828.3	1956.5	5412.3
				2	12.38	11.90	11.90	11.90	347023.7	347875.8	347875.8	347875.8	1535.8	3033.4	3153.9	8565.9
				3	12.78	12.78	12.65	12.62	333099.1	333099.1	341126.8	340369.1	540.5	657.1	994.4	4713.4
				4	11.99	11.80	11.74	11.74	339281.4	333766.9	332132.9	333836.2	1466.8	2610.1	3415.3	9726.7
				5	12.08	11.83	11.45	11.45	306807.6	312742.3	335916.6	335916.6	1028.9	1543.7	2705.3	7233.6
				Avg.	12.35	12.15	12.04	12.04	340924.5	341233.4	347147.0	347336.1	1237.2	1934.5	2445.1	7130.4
				Std. Dev.	0.32	0.44	0.50	0.49	25837.59	24393.89	18593.23	18331.04	450.71	930.03	981.67	2097.95

Table 2. Selection of the number of consecutive non-improving solutions for Metaheuristic method

As shown in Table 2, for the smaller instances, the length of tabu-list was fixed at 5, and the problems were solved for (5, 7, 10, 15) consecutive non-improving solutions. Similarly, for medium and large sized problems, the length of tabu-list was fixed at 7 and 10 respectively, and the problems were solved for (10, 15, 20, 30) and (10, 20, 30, 100) consecutive non-improving solutions respectively. By observing the average value of the maximum restoration time generated by all five instances of each size, we concluded that the proposed multi-stage metaheuristic method will produce good solutions when a minimum of 10 consecutive non-improving solutions are allowed for small instances, whereas, for medium and large instances it will produce good solutions when minimum 20 and 30 consecutive non-improving solutions are allowed, respectively.

4.2.1.2 Hybrid method

The Hybrid method is the combination of metaheuristic and MILP. Therefore, considering the NP-hardness of the problem and the capacity of the computer, it can only solve small and medium size instances in a reasonable amount of time. By following the same technique mentioned in section 4.2.1.1, we will determine the maximum number of the consecutive non-improving solution according to the size of the problem.

As shown in Table 3, for small instances, the length of the tabu-list was fixed at 5, whereas for the medium sized instances it was fixed at 7. The small and medium sized instances were solved for (3, 5, 7, 10) and (5, 10, 15, 20) number of consecutive non-improving solutions respectively. By observing the average value of the maximum restoration time generated by all five instances of each size, we can conclude that for small sized problems, the proposed hybrid method will produce a good solution when we allow 7 consecutive non-improving solutions, whereas for medium size instances the method will produce a good solution when we allow 15 consecutive non-improving solutions.

SMALL SIZE					Final restoration time				Final Weighted wait time				CPU Runtime (second)			
Number of consecutive non-improving solutions					3	5	7	10	3	5	7	10	3	5	7	10
2 Depots	7 Outages	2 PLT	5 Tabu list length	1	7.48	7.48	7.26	7.26	1085.1	1085.1	1060.6	1060.6	0.47	0.93	2.09	2.73
				2	7.47	7.47	7.47	7.47	1146.8	1146.8	1146.8	1146.8	0.39	0.74	1.00	1.55
				3	10.31	10.31	10.26	10.26	1852.7	1852.7	1850.0	1850.0	0.92	1.12	2.26	3.03
				4	8.99	8.37	8.37	8.37	1858.3	1858.8	1858.8	1858.8	0.37	1.36	1.67	2.08
				5	7.67	7.67	7.67	7.67	1873.5	1873.5	1873.5	1873.5	0.54	0.71	1.05	1.23
				Avg.	8.38	8.26	8.21	8.21	1563.3	1563.4	1558.0	1558.0	0.54	0.97	1.61	2.12
				Std. Dev.	1.25	1.20	1.22	1.22	409.0	409.1	415.9	415.9	0.22	0.27	0.58	0.76
Number of consecutive non-improving solutions					3	5	7	10	3	5	7	10	3	5	7	10
3 Depots	10 Outages	3 PLT	5 Tabu list length	1	8.93	8.65	8.65	8.65	1925.4	2040.1	2040.1	2040.1	0.90	2.03	2.41	2.92
				2	9.97	9.97	9.52	9.52	2585.6	2585.6	2665.9	2665.9	0.89	1.40	3.94	4.07
				3	8.86	8.86	8.40	8.40	2335.8	2335.8	2502.6	2502.6	0.86	1.65	3.01	3.79
				4	8.36	8.10	8.10	8.10	2012.0	1863.1	1863.1	1863.1	0.63	1.57	2.02	2.83
				5	7.57	7.57	7.57	7.57	1541.4	1541.4	1541.4	1541.4	0.84	1.12	1.69	2.14
				Avg.	8.74	8.63	8.45	8.45	2080.0	2073.2	2122.6	2122.6	0.83	1.56	2.62	3.15
				Std. Dev.	0.88	0.90	0.72	0.72	399.9	406.0	461.4	461.4	0.11	0.33	0.89	0.78
MEDIUM SIZE					Final restoration time				Final Weighted wait time				CPU Runtime (second)			
Number of consecutive non-improving solutions					5	10	15	20	5	10	15	20	5	10	15	20
5 Depots	50 Outages	11 PLT	7 Tabu list length	1	11.28	11.17	11.17	11.17	15544.8	15999.9	15999.9	15999.9	8.9	18.3	27.0	26.4
				2	11.45	11.45	11.00	11.00	22162.7	22162.7	22851.6	22851.6	15.5	19.1	91.5	92.3
				3	10.14	10.14	10.14	10.14	19610.7	19610.7	19610.7	19610.7	14.8	18.2	24.2	29.6
				4	11.16	11.16	11.16	11.16	46903.3	46903.3	46903.3	46903.3	19.6	19.9	24.5	27.2
				5	12.01	11.94	11.94	11.94	30151.5	30316.7	30316.7	30316.7	27.7	42.3	47.1	50.4
				Avg.	11.21	11.17	11.08	11.08	26874.6	26998.6	27136.4	27136.4	17.31	23.56	42.88	45.19
				Std. Dev.	0.68	0.66	0.64	0.64	12401.8	12310.0	12246.0	12246.0	6.95	10.52	28.81	28.15
Number of consecutive non-improving solutions					5	10	15	20	5	10	15	20	5	10	15	20
10 Depots	100 Outages	40 PLT	7 Tabu list length	1	6.86	6.64	6.64	6.64	44770.9	43530.3	43530.3	43530.3	108.10	141.23	152.85	169.84
				2	7.02	7.02	7.02	7.02	57445.8	57445.8	57445.8	57445.8	112.62	111.81	115.46	124.07
				3	7.02	7.02	7.02	7.02	35310.4	35310.4	35310.4	35310.4	108.65	111.15	122.51	126.39
				4	6.20	6.20	6.20	6.20	45350.1	45350.1	45350.1	45350.1	137.29	149.37	144.53	161.83
				5	6.68	6.68	6.68	6.68	51282.5	51282.5	51282.5	51282.5	69.76	73.33	73.41	86.42
				Avg.	6.78	6.72	6.72	6.72	45719.3	45409.2	45409.2	45409.2	116.7	128.4	133.8	130.6
				Std. Dev.	0.34	0.34	0.34	0.34	8240.9	8336.6	8336.6	8336.6	24.2	30.0	31.1	33.5

Table 3. Selection of the number of consecutive non-improving solutions for Hybrid method

After determining the number of consecutive non-improving solutions that should be allowed for each size, we can determine the most suitable length of tabu-list for each size.

4.2.2 Selection of the length of the tabu-list

To determine the most suitable length of tabu-list, we will fix the number of consecutive non-improving solutions obtained from the method mentioned in section 4.2.1, and test the problems for different lengths of the tabu-list.

4.2.2.1 Multi-Stage Metaheuristic method

As mentioned in section 4.2.1.1, the suitable number of consecutive non-improving solutions for small, medium and large size problems are 10, 20 and 30 respectively. Therefore, we will fix these numbers of non-improving solutions and solve the problems for different lengths of tabu-list. As shown in Table 4, the small sized problems were solved for the tabu-list's length of (2, 3, 5, 7),

SMALL SIZE					Final restoration time				Final Weighted wait time				CPU Runtime (second)			
Length of tabu list					2	3	5	7	2	3	5	7	2	3	5	7
2 Depots	7 Outages	2 PLT	10 Non-impr. Solutions	1	7.27	7.26	7.27	7.27	1074.2	1060.6	1074.2	1074.2	0.14	0.35	0.15	0.13
				2	7.47	7.47	7.47	7.47	1146.8	1146.8	1146.8	1146.8	0.17	0.17	0.15	0.15
				3	10.49	10.49	10.26	10.49	1867.3	1867.3	1850.0	1867.3	0.13	0.16	0.39	0.11
				4	8.82	8.37	8.37	8.37	1925.8	1858.8	1858.8	1858.8	0.19	0.24	0.25	0.15
				5	7.67	7.67	7.67	7.67	1873.5	1873.5	1873.5	1873.5	0.13	0.14	0.14	0.13
				Avg.	8.35	8.25	8.21	8.25	1577.5	1561.4	1560.7	1564.1	0.15	0.21	0.21	0.13
				Std. Dev.	1.34	1.32	1.22	1.32	427.7	419.0	411.8	414.9	0.03	0.09	0.10	0.02
Length of tabu list					2	3	5	7	2	3	5	7	2	3	5	7
3 Depots	10 Outages	3 PLT	10 Non-impr. Solutions	1	8.93	8.65	8.65	8.65	1925.4	2040.1	2040.1	2040.1	0.19	0.26	0.28	0.28
				2	9.80	9.80	9.52	9.52	2689.8	2590.0	2637.1	2637.1	0.27	0.33	0.38	0.25
				3	8.78	8.78	8.70	8.78	2463.1	2463.1	2524.4	2463.1	0.29	0.28	0.52	0.32
				4	8.10	8.10	8.10	8.10	1863.1	1863.1	1863.1	1863.1	0.24	0.25	0.28	0.25
				5	8.52	7.76	7.75	7.86	1665.2	1796.2	1647.5	1764.0	0.24	0.48	0.42	0.42
				Avg.	8.83	8.62	8.55	8.58	2121.3	2150.5	2142.4	2153.5	0.25	0.32	0.37	0.30
				Std. Dev.	0.63	0.78	0.68	0.65	433.9	357.5	425.5	380.4	0.04	0.10	0.10	0.07
MEDIUM SIZE					Final restoration time				Final Weighted wait time				CPU Runtime (second)			
Length of tabu list					3	5	7	10	3	5	7	10	3	5	7	10
5 Depots	50 Outages	11 PLT	20 Non-impr. Solutions	1	11.39	11.39	11.39	11.39	15057.9	15057.9	15057.9	15088.5	4.2	4.4	4.4	4.6
				2	11.43	11.47	11.47	11.47	21963.1	23844.9	23884.9	23904.3	10.1	6.2	7.1	7.0
				3	10.32	10.32	10.58	10.61	20388.1	20017.6	22309.4	20516.6	6.8	8.7	6.3	5.8
				4	11.51	11.59	11.38	11.34	49387.0	45487.1	48683.4	46667.8	8.1	7.7	11.5	18.4
				5	11.95	11.96	11.96	11.96	30836.3	28863.2	28863.2	28863.2	14.1	8.9	9.1	9.6
				Avg.	11.32	11.34	11.36	11.35	27526.5	26654.1	27759.8	27008.1	8.68	7.19	7.68	9.08
				Std. Dev.	0.60	0.61	0.49	0.48	13474.1	11683.0	12699.2	12080.9	3.74	1.89	2.71	5.55
Length of tabu list					3	5	7	10	3	5	7	10	3	5	7	10
10 Depots	100 Outages	40 PLT	20 Non-impr. Solutions	1	6.86	6.85	6.85	6.85	49013.2	48836.7	48836.7	49352.1	30.37	39.62	39.00	40.41
				2	7.08	7.08	7.08	7.08	59688.8	59688.8	59688.8	61166.9	29.63	27.71	27.04	26.53
				3	7.09	7.09	7.09	7.09	38054.2	38054.2	38054.2	38022.0	19.59	19.88	19.73	21.32
				4	6.33	6.33	6.33	6.37	44715.2	44713.4	44713.4	44724.4	24.90	23.76	24.44	16.54
				5	6.71	6.71	6.71	6.71	52557.0	51699.0	51699.0	51699.0	25.57	19.65	19.67	20.42
				Avg.	6.81	6.81	6.81	6.82	48805.7	48598.4	48598.4	48992.9	26.01	26.13	25.98	25.05
				Std. Dev.	0.31	0.31	0.31	0.30	8137.6	8045.8	8045.8	8576.7	4.32	8.23	7.93	9.30
LARGE SIZE					Final restoration time (hour)				Final Weighted wait time (hour)				CPU Runtime (second)			
Length of tabu list					5	10	15	25	5	10	15	25	5	10	15	25
31 Depots	500 Outages	140 PLT	30 Non-impr. Solutions	1	10.76	10.76	10.92	10.9	270790.5	271702.0	272510.5	274470.5	1153.5	1095.5	897.6	877.3
				2	10.13	10.04	9.92	10.1	257318.4	255289.2	256507.7	255740.4	1125.5	1209.8	1708.8	1321.0
				3	10.56	10.56	11.13	11.1	288486.3	290805.4	283295.9	283295.9	934.1	1326.4	606.4	652.5
				4	10.31	10.30	10.32	10.3	253270.4	253261.5	253376.9	253376.9	643.7	655.1	787.8	761.7
				5	10.15	10.15	10.07	10.1	231668.5	231668.5	233395.9	232181.4	750.5	731.8	1165.3	947.1
				Avg.	10.38	10.36	10.47	10.49	260306.8	260545.3	259817.4	259813.0	921.5	1003.7	1033.2	911.9
				Std. Dev.	0.27	0.30	0.53	0.49	21110.4	22106.0	19123.8	19923.6	224.7	296.0	428.6	254.8
Length of tabu list					5	10	15	25	5	10	15	25	5	10	15	25
31 Depots	600 Outages	140 PLT	30 Non-impr. Solutions	1	12.34	12.46	12.46	12.27	385188.8	378682.9	378682.9	377228.6	2312.5	1920.9	1936.2	2576.4
				2	11.83	11.90	12.00	11.89	340488.6	347875.8	353614.4	341662.0	3475.6	3122.5	2901.2	3530.6
				3	12.59	12.65	12.78	12.78	341349.5	341126.8	330049.8	330049.8	1061.7	982.6	720.4	706.4
				4	11.85	11.74	11.92	11.88	329807.8	332132.9	333163.1	336949.5	2322.4	3359.8	1990.8	2043.1
				5	12.08	11.45	11.56	11.56	310599.1	335916.6	316024.5	313062.0	1334.2	2674.0	2432.3	2674.2
				Avg.	12.14	12.04	12.15	12.08	341486.8	347147.0	342307.0	339790.4	2101.3	2412.0	1996.2	2306.2
				Std. Dev.	0.33	0.50	0.48	0.47	27392.2	18593.2	24370.5	23571.6	955.4	969.0	812.5	1041.2

Table 4. Selection of the length of tabu-list for Metaheuristic method

whereas the medium and large sized problems are solved for tabu-list’s length of (3, 5, 7, 10) and (5, 10, 15, 25) respectively.

By observing the average values of the final restoration time for each of the sizes in Table 4, we can see that the most suitable length of tabu-list for small, medium and large size problems are 5, 3 and 10 respectively. Therefore, to compare the multi-stage metaheuristic method with other proposed methods, we will keep the pair of (number of consecutive non-improving solution, the length of tabu list) at (10, 5) for small instances, (20, 3) for medium instances and (30, 10) for large instances. This method of selecting the tabu search parameter is not optimum and needs more investigation. But since the general recommendation of tabu search parameters is not considered in the scope of the thesis, we have chosen these values for comparison purpose.

4.2.2.2 Hybrid method

The selection of the most suitable length of the tabu list for the Hybrid method is made by using the same technique mentioned in section 4.2.2.1. As mentioned in section 4.2.1.2, for the Hybrid method, the most suitable number of non-improving solutions for small and medium-sized problems are 7 and 10 respectively. Therefore, we will fix this parameter at 7 and 10, and will solve the problems for the tabu-list’s length of (2, 3, 5, 7) and (3, 5, 7, 10) for small and medium size problems, respectively.

By observing the average values of final restoration times in Table 5, we can conclude that the most suitable length of the tabu-list for small sized problems is 5. For medium size problems, we can see that for (10 depots, 100 outages, 40 PLTs), the hybrid method performs better at tabu-list’s length of 3, but the difference between the average final restoration times for the lengths 3 and 7 is only 0.09 hours. On the other hand, for (5 depots, 50 outages, 11 PLTs) the method produces its best result at the tabu-list's length of 7. Therefore, for the medium sized problem, 7 will be a good choice for the length for tabu-list. From the results obtained from section 4.2.1.2 and 4.2.2.2, to compare the hybrid method with other proposed methods we will keep the pair of (number of consecutive non-improving solution, the length of tabu list) at (7, 5) for small instances and (15, 7) for medium instances.

SMALL SIZE					Final restoration time (hour)				Final Weighted wait time (hour)				CPU Runtime (second)			
Length of tabu list					2	3	5	7	2	3	5	7	2	3	5	7
2 Depots	7 Outages	2 PLT	7 Non-impr. Solutions	1	7.48	7.48	7.26	7.31	1085.1	1085.1	1060.6	1060.8	0.79	0.94	1.97	1.18
				2	7.47	7.47	7.47	7.47	1146.8	1146.8	1146.8	1146.8	0.77	0.85	1.03	1.07
				3	10.31	10.31	10.26	10.31	1852.7	1852.7	1850.0	1852.7	0.76	1.07	2.27	1.28
				4	8.82	8.37	8.37	8.37	1925.8	1858.8	1858.8	1858.8	1.10	1.27	1.61	1.26
				5	7.67	7.67	7.67	7.67	1873.5	1873.5	1873.5	1873.5	0.68	0.79	1.02	1.14
				Avg.	8.35	8.26	8.21	8.22	1576.8	1563.4	1558.0	1558.5	0.82	0.98	1.58	1.18
				Std. Dev.	1.23	1.20	1.22	1.23	422.1	409.1	415.9	416.3	0.16	0.19	0.56	0.09
Length of tabu list					2	3	5	7	2	3	5	7	2	3	5	7
3 Depots	10 Outages	3 PLT	7 Non-impr. Solutions	1	8.93	8.65	8.65	8.65	1925.4	2040.1	2040.1	2040.1	1.41	2.34	3.50	2.61
				2	9.97	9.97	9.52	9.52	2585.6	2585.6	2665.9	2665.9	1.13	1.63	3.54	2.04
				3	8.86	8.86	8.40	8.40	2335.8	2335.8	2502.6	2502.6	1.21	1.60	2.97	2.80
				4	8.10	8.10	8.10	8.10	1863.1	1863.1	1863.1	1863.1	1.60	2.08	2.02	2.02
				5	7.57	7.57	7.57	7.57	1541.4	1541.4	1541.4	1541.4	1.29	1.73	1.55	1.55
				Avg.	8.69	8.63	8.45	8.45	2050.3	2073.2	2122.6	2122.6	1.33	1.87	2.72	2.21
				Std. Dev.	0.91	0.90	0.72	0.72	411.6	406.0	461.4	461.4	0.18	0.32	0.90	0.50
MEDIUM SIZE					Final restoration time (hour)				Final Weighted wait time (hour)				CPU Runtime (second)			
Length of tabu list					3	5	7	10	3	5	7	10	3	5	7	10
5 Depots	50 Outages	11 PLT	15 Non-impr. Solutions	1	11.17	11.17	11.17	11.17	15999.9	15999.9	15999.9	15999.9	21.5	21.7	21.6	21.3
				2	11.45	11.29	11.00	11.45	22162.7	24553.7	22851.6	22162.7	22.4	39.5	83.5	19.6
				3	10.14	10.14	10.14	10.31	19610.7	19610.7	19610.7	19527.7	22.5	23.5	23.2	16.2
				4	11.13	11.16	11.16	11.16	49651.9	46903.3	46903.3	46903.3	60.3	20.9	23.1	24.2
				5	11.94	11.94	11.94	11.94	30316.7	30316.7	30316.7	30316.7	47.9	47.0	47.0	47.5
				Avg.	11.17	11.14	11.08	11.21	27548.4	27476.8	27136.4	26982.0	34.93	30.51	39.68	25.77
				Std. Dev.	0.66	0.64	0.64	0.59	13431.4	12120.2	12246.0	12322.5	18.03	11.97	26.70	12.49
Length of tabu list					3	5	7	10	3	5	7	10	3	5	7	10
10 Depots	100 Outages	40 PLT	15 Non-impr. Solutions	1	6.58	6.58	6.64	6.86	45149.7	43147.3	43530.3	44770.9	138.16	210.27	165.25	119.10
				2	6.64	6.97	7.02	7.02	59924.4	57365.7	57445.8	57445.8	157.20	159.31	109.27	107.88
				3	7.02	7.02	7.02	7.02	35310.4	35310.4	35310.4	35310.4	129.68	138.76	113.60	108.24
				4	6.20	6.20	6.20	6.20	45350.1	45350.1	45350.1	45350.1	161.50	191.11	141.76	138.90
				5	6.68	6.68	6.68	6.68	51206.2	51408.2	51282.5	51282.5	84.10	119.66	72.13	69.98
				Avg.	6.62	6.69	6.71	6.76	47388.2	46516.4	46583.8	46831.9	134.13	163.82	120.40	108.82
				Std. Dev.	0.29	0.33	0.34	0.34	9038.4	8365.5	8336.6	8240.9	30.91	37.09	35.24	25.10

Table 5. Selection of the length of tabu-list for hybrid method

As mentioned earlier, this technique of selecting the tabu search parameters needs further investigation, but for the comparison between the proposed methods, this technique yields good results.

4.3 Comparison of proposed methods

In this section, the performances of all three proposed methods i.e. MILP, Hybrid, and Multi-Stage Metaheuristic are compared. Before starting the comparison, it is important to note that the objectives of the problem under investigation (overall restoration time and the total weighted wait time) may contradict in some cases. Therefore, it is not necessary that the solution which causes improvement in overall restoration time will also improve the total weighted wait time. In contrast, it may increase the total weighted wait time. But the performance of any method will be rated based on the primary objective value, i.e. overall restoration time, and by some considerations for weighted wait time explained later.

4.3.1 Performance of Multi-Stage Metaheuristic and Hybrid method with respect to optimum method (MILP)

Because of the NP-hardness of the problem under investigation, the MILP model was only able to solve the small size problems. As mentioned in section 3.1, selection of the importance factor for both the objectives of the problem is very subjective, making the comparison difficult for the solutions obtained from the MILP model with those obtained from the Hybrid method and Multi-Stage Metaheuristic method. Therefore, for the comparison purpose, the importance factors are eliminated from the objective function and two constraints are added (one at a time) in the MILP model to restrict the objective function values of the overall restoration time and total weighted wait time to as discussed in this section.

For instance, a problem is first solved using either a multi-stage metaheuristic method or a hybrid method. As we can see in Table 6- instance #1, the restoration time and weighted wait time obtained from Multi-Stage Metaheuristic method are 7.27 hours and 1074.2 hours respectively.

Therefore, to obtain the optimum restoration time from MILP model, a constraint is added to the model proposed in section 3.1 which ensures that the total weighted wait time of the solution does not exceed 1074.2 hours, obtained from the Meta-Heuristic method. Afterward, the MILP model is solved with Gurobi solver and the resultant restoration time is noted as "the optimum restoration time constrained to the result of Multi-Stage Metaheuristic method". In this case, 7.26 hours is the optimum restoration time, hence the optimality percentage of the Multi-Stage Metaheuristic method for the restoration time objective is $\left(\frac{7.26}{7.27} =\right)$ 99.78%.

Similarly, to obtain the weighted wait time optimality percentage, a constraint is added which ensures that the maximum restoration time does not exceed the restoration time obtained from Multi-Stage Metaheuristic method, i.e. 7.27 hours. The MILP model is again solved to obtain the total weighted wait time and it is noted as "the optimum weighted wait time constrained to the result of Multi-Stage Metaheuristic method". In this case, it is 1054.9 hours, hence the optimality percentage of the Meta-heuristic method for the weighted wait time objective is $\left(\frac{1054.9}{1074.2} =\right)$98.20%.

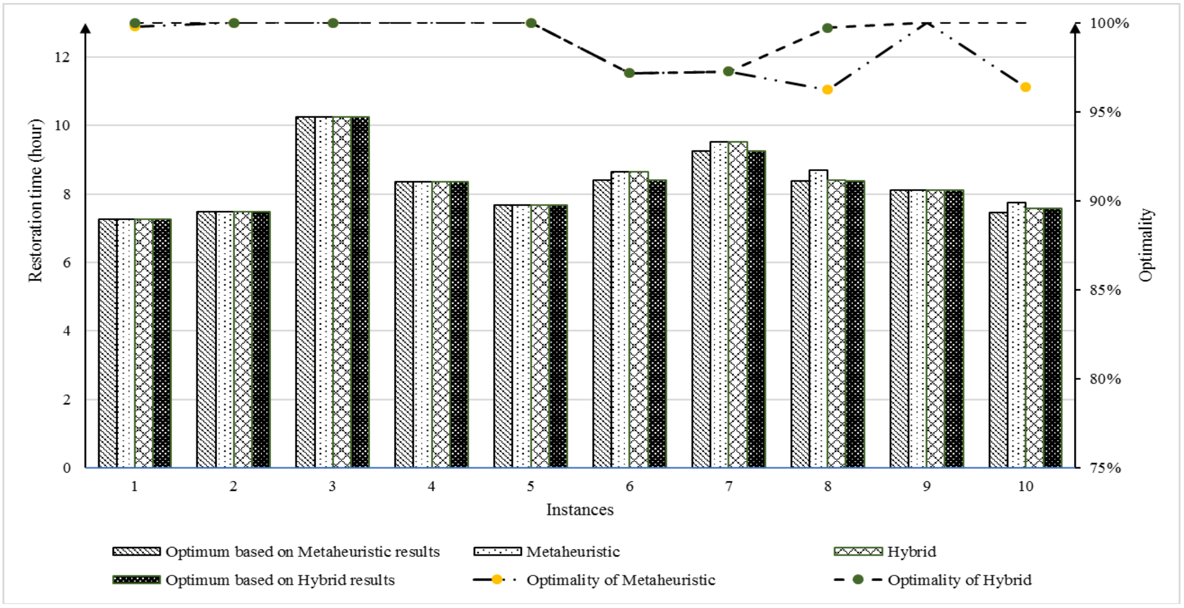
				Restoration time (hour)				Optimality		Weighted wait time (hour)				Optimality		CPU Runtime (second)			
Method				Meta	Hybrid	Optimum constrained to results of		Meta	Hybrid	Meta	Hybrid	Optimum constrained to results of		Meta	Hybrid	Meta	Hybrid	Optimum Restoration	Optimum wait
			Instance			Meta	Hybrid					Meta	Hybrid						
2 Depots	7 Outages	2 PLT	1	7.27	7.26	7.26	7.26	99.78%	100%	1074.2	1060.6	1054.9	1055.3	98.20%	99.50%	0.08	1.97	0.59	0.29
			2	7.47	7.47	7.47	7.47	100%	100%	1146.8	1146.8	1146.8	1146.8	100.00%	100%	0.10	1.03	0.54	0.29
			3	10.26	10.26	10.26	10.26	100%	100%	1850.0	1850.0	1842.0	1842.0	99.56%	100%	0.23	2.27	0.83	0.36
			4	8.37	8.37	8.37	8.37	100%	100%	1858.8	1858.8	1848.4	1848.4	99.44%	99.44%	0.12	1.61	0.75	0.26
			5	7.67	7.67	7.67	7.67	100%	100%	1873.5	1873.5	1873.5	1873.5	100.00%	100%	0.08	0.99	0.95	0.26
			Avg.	8.21	8.21	8.21	8.21	99.96%	100%	1560.7	1558.0	1553.1	1553.2	99.44%	99.70%	0.12	1.57	0.73	0.29
			St. Dev.	1.22	1.22	1.22	1.22	0.001	0.00000	411.8	415.9	414.3	414.2	0.01	0.00	0.06	0.57	0.17	0.04
3 Depots	10 Outages	3 PLT	6	8.65	8.65	8.41	8.41	97.17%	97.17%	2040.1	2040.1	1901.2	1901.2	93.19%	93.19%	0.24	1.99	142.34	18.44
			7	9.52	9.52	9.26	9.26	97.27%	97.27%	2637.1	2665.9	2510.8	2510.8	95.21%	94.18%	0.34	2.84	85.77	2.35
			8	8.70	8.40	8.37	8.37	96.25%	99.73%	2524.4	2502.6	2262.7	2399.1	89.63%	95.87%	0.42	2.39	186.22	17.95
			9	8.10	8.10	8.10	8.10	100%	100%	1863.1	1863.1	1846.8	1846.8	99.12%	99.12%	0.28	2.02	60.13	8.69
			10	7.75	7.57	7.47	7.57	96.39%	100%	1647.5	1541.4	1541.4	1541.4	93.56%	100%	0.42	1.55	16.34	1.92
			Avg.	8.55	8.45	8.32	8.34	97.41%	98.83%	2142.4	2122.6	2012.6	2039.9	94.14%	96.47%	0.34	2.16	98.16	9.87
			St. Dev.	0.68	0.72	0.65	0.61	0.015	0.015	425.5	461.4	378.4	404.9	0.03	0.03	0.08	0.48	67.08	8.06

Table 6. comparison of Metaheuristic and Hybrid method with Optimum MILP method

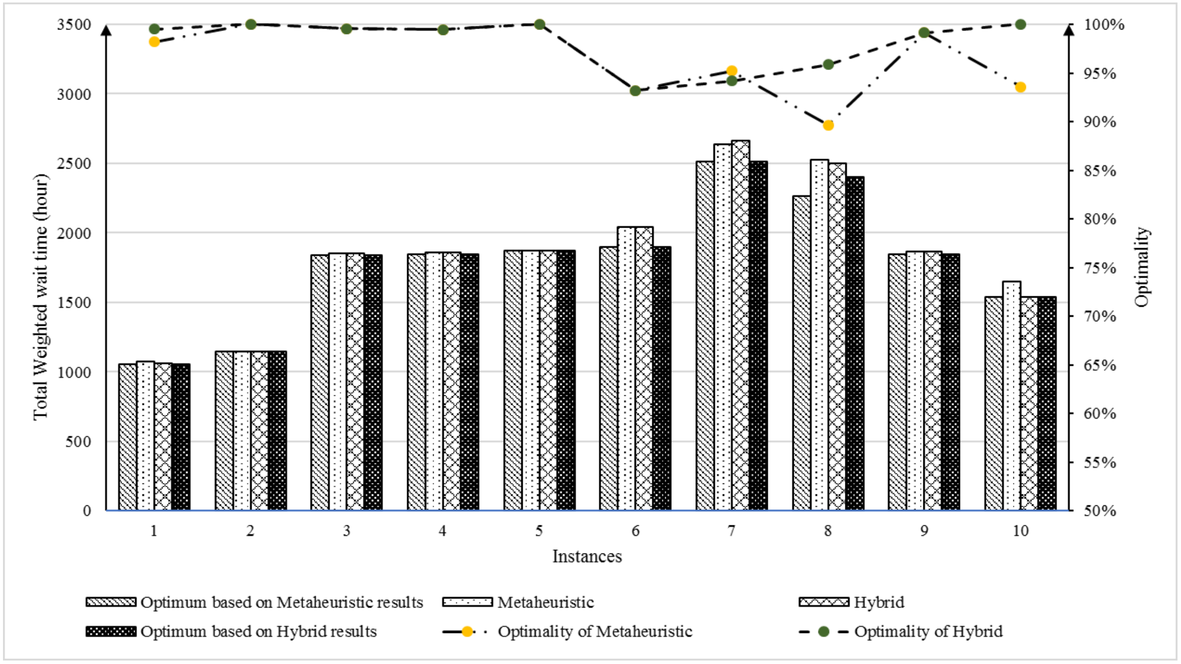
After that, the problem is solved by the Hybrid method as well and the constraints are added in the MILP model as explained for Multi-Stage Metaheuristic method. Following the same procedure, the optimum restoration and total weighted wait time constrained to the results of the hybrid method are obtained. For instance #1, the optimum restoration time and total weighted wait time constrained to the results of the hybrid method are 7.26 hours and 1060.6 hours, respectively, which result in $\left(\frac{7.26}{7.26}=\right)100\%$ and $\left(\frac{1060.6}{1074.2}=\right)99.50\%$ optimality, respectively.

As shown in Table 6 for (3 depots, 10 outages, 3 PLTs), the average optimality of the Multi-Stage Metaheuristic method for overall restoration time and total weighted wait time is 97.41% and 94.14% respectively. And that for the Hybrid method is 98.83% and 96.47% respectively. The computation time for both Metaheuristic and Hybrid method are mentioned in Table 6. The computation time under title "Optimum restoration" is the average of the computation time to obtain optimum restoration time solving the MILP model constrained to the results of Multi-Stage Metaheuristic and Hybrid method. And the computation time under title "Optimum weighted wait time" is the average of the computation time to obtain optimum weighted wait time solving MILP model constrained to the results of Multi-Stage Metaheuristic and Hybrid method.

As shown in Figure 15(a), for all instances, the restoration time of Hybrid method is either same or better than that of the Multi-Stage Metaheuristic method. For (2 depots, 7 outages, 2 PLTs), the Hybrid method was able to achieve the optimum solution for all five instances, whereas the Multi-Stage Metaheuristic method achieved the optimum solution in four instances. For (3 depots, 10 outages, 3 PLTs), the Hybrid method was able to achieve the optimum solution in two instances, whereas Multi-Stage Metaheuristic Method achieved 100% optimality in one instance only. As shown in Figure 15(b), except instance #7, the total weighted wait time of Hybrid method is also either same or better than the Metaheuristic. In instance #7, although Hybrid method obtained better restoration time than Multi-Stage Metaheuristic method, its total weighted wait time was higher than that of Metaheuristic. As mentioned in section 1.1, instance #7 exemplifies an instance where both the primary and secondary objectives may contradict each other.



(a) Based on Restoration time



(b) Based on Total Weighted wait time

Figure 15. Performance of Multi-Stage Metaheuristic and Hybrid methods with respect to Optimum method

Observing the average optimality of both the methods for each sub-size in Table 6, we can conclude that both Metaheuristic and Hybrid method perform very well and are also able to achieve optimum solutions in many cases. Because of the NP-hardness of the problem under investigation, the optimum method was not able to solve more than 11 outages, therefore for medium and large sized problems, we could not test the performance of Metaheuristic and Hybrid method with respect to the optimum MILP method. But looking at their performance on small instances, both the methods are expected to perform well on medium and large sizes as well.

4.3.2 Performance of Hybrid method with respect to Multi-Stage Metaheuristic method

SMALL SIZE				Restoration time (hour)			Weighted wait time (hour)			CPU Runtime (second)	
Method				Metaheuristic	Hybrid	Performance of Hybrid	Metaheuristic	Hybrid	Performance of Hybrid	Metaheuristic	Hybrid
2 Depots	7 Outages	2 PLT	1	7.27	7.26	+0.22%	1074.23	1060.60	+1.27%	0.08	1.97
			2	7.47	7.47	0.00%	1146.78	1146.78	0.00%	0.10	1.03
			3	10.26	10.26	0.00%	1850.04	1850.04	0.00%	0.23	2.27
			4	8.37	8.37	0.00%	1858.84	1858.84	0.00%	0.12	1.61
			5	7.67	7.67	0.00%	1873.53	1873.53	0.00%	0.08	0.99
			Avg.	8.21	8.21	+0.04%	1560.69	1557.96	+0.25%	0.12	1.57
			Std. Dev.	1.22	1.22	0.001	411.84	415.89	0.01	0.06	0.57
3 Depots	10 Outages	3 PLT	1	8.65	8.65	0.00%	2040.08	2040.08	0.00%	0.24	1.99
			2	9.52	9.52	0.00%	2637.14	2665.91	-1.09%	0.34	2.84
			3	8.70	8.40	+3.49%	2524.40	2502.55	+0.87%	0.42	2.39
			4	8.10	8.10	0.00%	1863.11	1863.11	0.00%	0.28	2.02
			5	7.75	7.57	+2.34%	1647.46	1541.43	+6.44%	0.42	1.55
			Avg.	8.55	8.45	+1.17%	2142.44	2122.62	+1.24%	0.34	2.16
			Std. Dev.	0.68	0.72	0.016	425.48	461.37	0.03	0.08	0.48

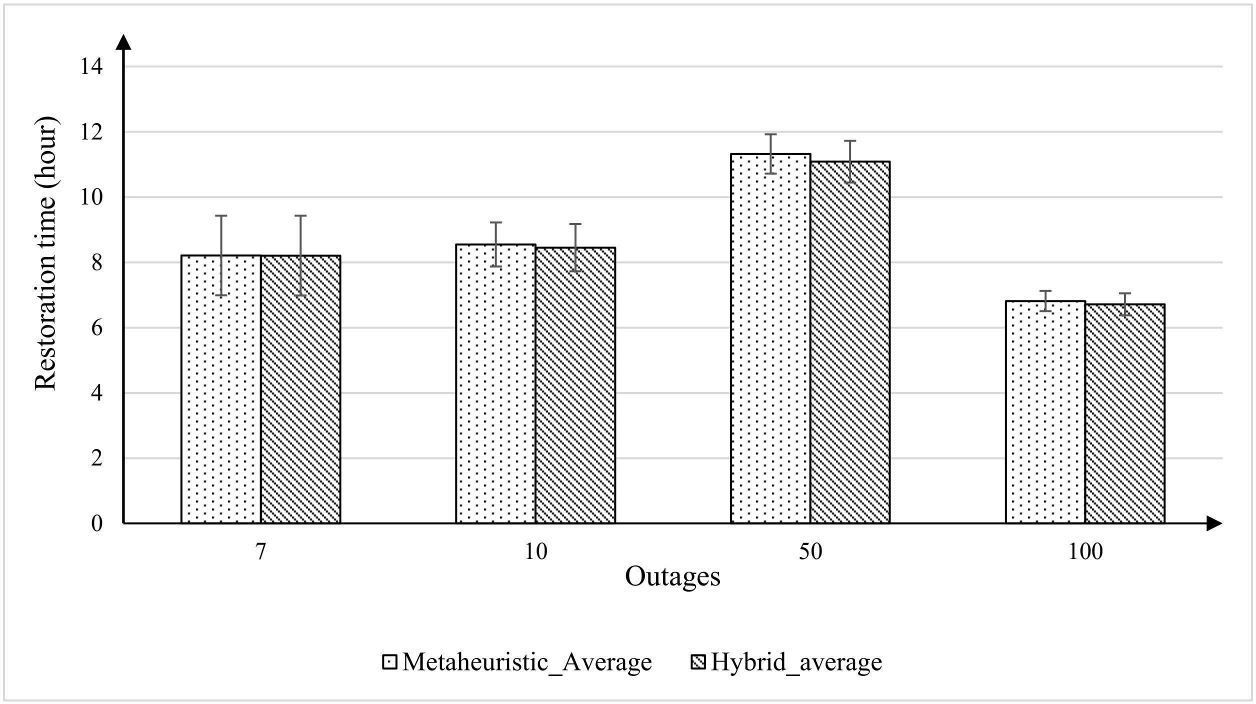
MEDIUM SIZE				Restoration time (hour)			Weighted wait time (hour)			CPU Runtime (second)	
Method				Metaheuristic	Hybrid	Performance of Hybrid	Metaheuristic	Hybrid	Performance of Hybrid	Metaheuristic	Hybrid
5 Depots	50 Outages	11 PLT	1	11.39	11.17	+1.90%	15057.9	15999.87	-6.26%	4.2	21.56
			2	11.43	11.00	+3.78%	21963.1	22851.60	-4.05%	10.1	83.54
			3	10.32	10.14	+1.68%	20388.1	19610.73	+3.81%	6.8	23.19
			4	11.51	11.16	+3.06%	49387.0	46903.27	+5.03%	8.1	23.14
			5	11.95	11.94	+0.07%	30836.3	30316.65	+1.69%	14.1	46.96
			Avg.	11.32	11.08	+2.10%	27526.49	27136.42	+0.05%	8.68	39.68
			Std. Dev.	0.60	0.64	0.014	13474.12	12246.03	0.05	3.74	26.70
10 Depots	100 Outages	40 PLT	1	6.86	6.64	+3.24%	49013.2	43530.33	+11.19%	30.37	165.25
			2	7.08	7.02	+0.77%	59688.8	57445.79	+3.76%	29.63	109.27
			3	7.09	7.02	+0.96%	38054.2	35310.40	+7.21%	19.59	113.60
			4	6.33	6.20	+2.05%	44715.2	45350.13	-1.42%	24.90	141.76
			5	6.71	6.68	+0.40%	52557.0	51282.49	+2.43%	25.57	72.13
			Avg.	6.81	6.71	+1.48%	48805.72	46583.83	+4.63%	26.01	120.40
			Std. Dev.	0.31	0.34	0.012	8137.64	8336.60	0.05	4.32	0.00

Table 7. Performance of Hybrid method with respect to Multi-Stage Metaheuristic method

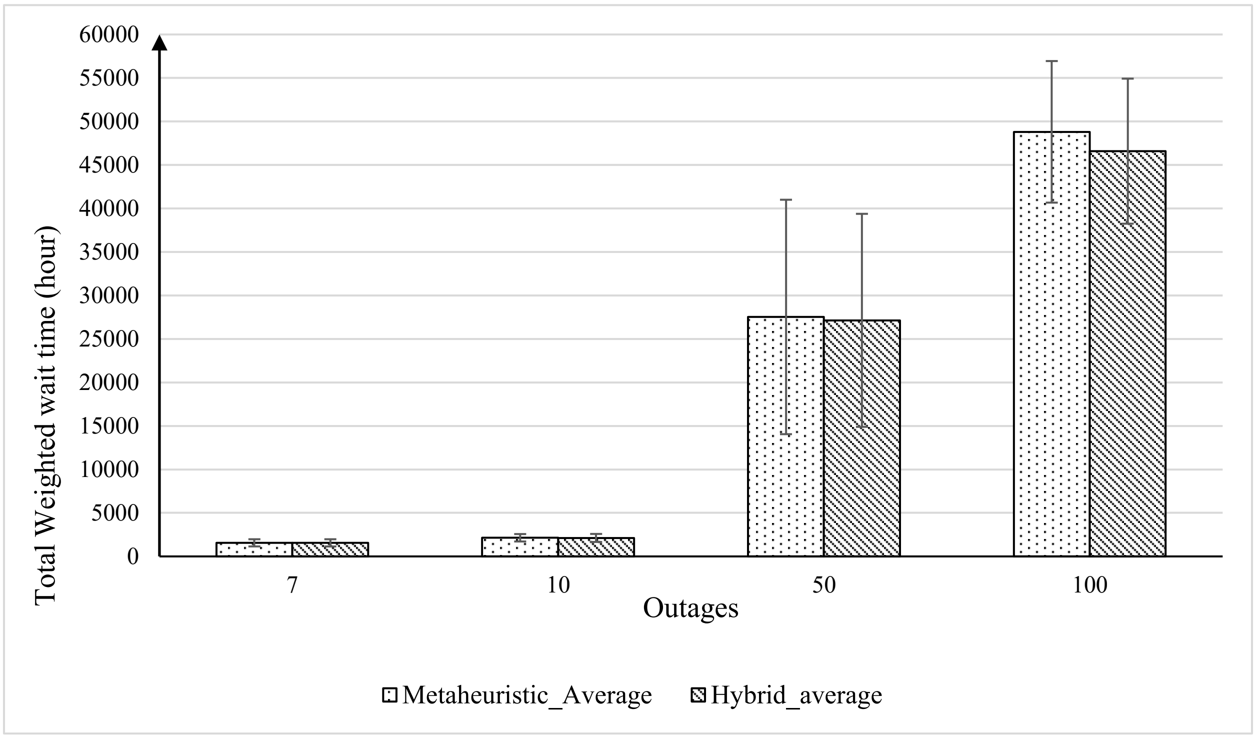
In this section, the performance of the Hybrid method is further tested with respect to the Multi-Stage Metaheuristic method. As the Hybrid method uses the combination of Metaheuristic and *optimum method*, it was only able to solve small and medium sized problems in a reasonable amount of time, and the results are shown in Table 7. The percentage noted under title "Performance of Hybrid" denotes how better (positive) or worse (negative) Hybrid method performed with respect to the Multi-Stage Metaheuristic method. As shown in Table 7, for small-sized instances, the restoration times of Hybrid method are either same or better than the Multi-Stage Metaheuristic method, whereas, for all medium sized instances, the restoration times of Hybrid method are better than Multi-Stage Metaheuristic method. Moreover, in most cases, the

Hybrid method has either same or better weighted wait times than Multi-Stage Metaheuristic method in most cases; the higher weighted wait time in some cases exemplifies the possibility of contradictory behavior of both objective functions as mentioned in section 1.1. Moreover, the difference in computation time of both the methods is not that huge, which implies that it is advantageous to solve these problems with Hybrid method to obtain the better objective values. However, the performance of the hybrid method also depends on which third-party solver is used to solve the problem which might be costly.

Figure 16 shows the performance of Hybrid method based on the average values of restoration and weighted wait times for all five instances in each sub-size. Figure 16(a) shows the performance based on average restoration time, whereas Figure 16(b) shows the performance based on average total weighted wait time. The number of outages on the x-axis indicates the sub-size of the problem. From Figure 16(a) and Figure 16(b), we can see that the *average* restoration time and weighted wait time of Hybrid method are better than that of Multi-Stage Metaheuristic method for all sub-sizes. However, considering the standard deviation of the data, the difference may not be statistically significant which implies the Multi-Stage Metaheuristic method also performs pretty well. The more statistical analysis could be done to investigate the performances of both methods.



(a) Based on Restoration time



(b) Based on total Weighted wait time

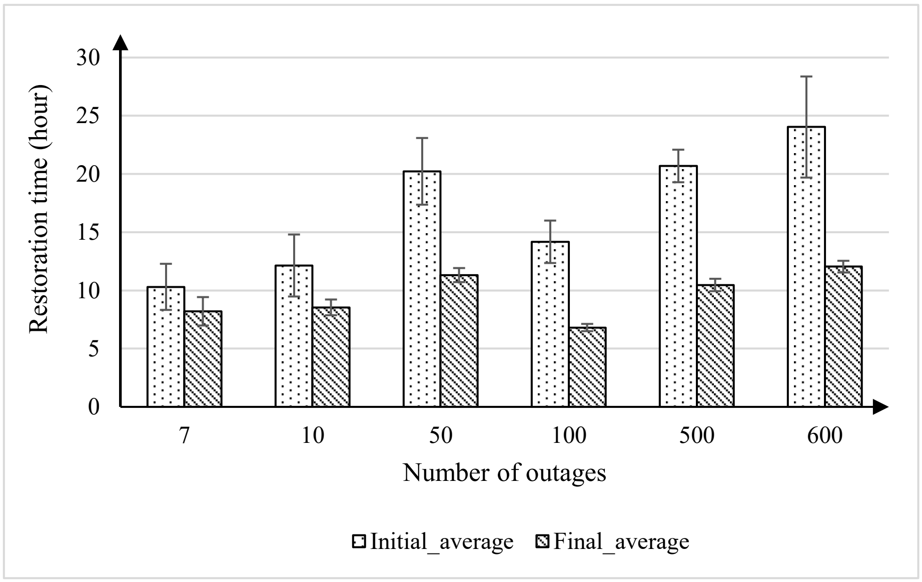
Figure 16. Performance of Hybrid method with respect to Multi-Stage Metaheuristic method

4.3.3 Performance of tabu-search in proposed Multi-Stage Metaheuristic method

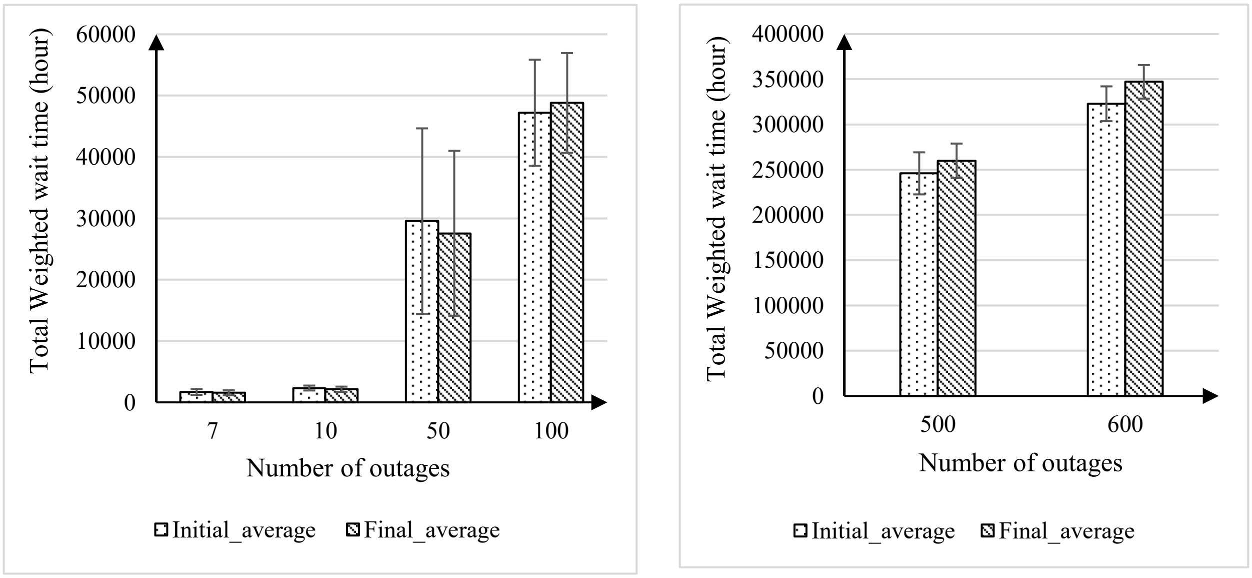
As mentioned in section 3.2.3 two tabu-search metaheuristics are used in the proposed Multi-Stage Metaheuristic method, one for the inter-route improvement, and the other for intra-route improvement which is a part of the inter-route improvement. In this section, we will consider both the tabu-search metaheuristics as one, because the final solution is the result of the performance of both metaheuristics. As mentioned in section 3.2.2, an initial routing heuristic is used to obtain an initial solution after clustering stage, and the tabu search metaheuristic is initiated from this solution. In this section, the performance of the tabu search metaheuristic is discussed based on the results from which it was initiated, and the final results that it achieved.

The Multi-Stage Metaheuristic method was tested on five data sets of each sub-size of the problems, i.e. (2 depots, 7 outages, 2 PLTs), (3 depots, 10 outages, 3 PLTs), (5 depots, 50 outages, 11 PLTs), (10 depots, 100 outages, 40 PLTs), (31 depots, 500 outages, 140 PLTs) and (31 depots, 600 outages, 140 PLTs). The resultant Restoration and total weighted wait time of all five instances of each sub-size can be found in Appendix A. In this discussion, we will denote the sub-sizes based on their respective number of outages. The comparison between the *average* values of the initial solution and that of the final solutions of five instances in each sub-size can be found in Figure 17. As shown in Figure 17(a), for all sub-sizes, there is a great improvement in final restoration time compared to the initial restoration time. For 7 outages and 10 outages, the tabu-search improved the overall restoration time by 19.62% and 26.58% respectively. For medium and large sizes, this improvement was even better. For 50 outages and 100 outages, the improvement was 43.40% and 51.26%, respectively, whereas, for 500 outages and 600 outages, the improvement was 49.13% and 48.45%, respectively.

Above numbers show how well the tabu search metaheuristic performed for all sizes of the problem. But as the objective of the initial routing heuristic is only to minimize the total weighted wait time, a great improvement in restoration time was expected from metaheuristic. Having said



(a) Restoration time



(b) Total Weighted wait time

Figure 17. Performance of tabu-search Metaheuristic

that, because of the contradictory behavior of the primary and secondary objective functions in some cases, the chances of tabu-search resulting in a higher value of the weighted wait time than the greedy heuristic was also expected. As shown in Figure 17(b) and Appendix A, for 7 outages, 10 outages and 50 outages, the tabu-search actually improved the total weighted wait time by 7.28%, 8.65 % and 6.31%, whereas, for 100 outages, 500 outages, and 600 outages, tabu search showed an increment in total weighted wait time by 3.70%, 5.78%, and 7.59%, respectively. On the other hand, as tabu-search improved the restoration time of 100 outages, 500 outages, and 600 outages, by 51.26%, 49.13%, and 48.45%, respectively, we can conclude that tabu-search is performing efficiently for all sizes of the problem under investigation.

Chapter 5 Conclusion and Future Research

5.1 Conclusion

In this research, a crew routing problem for post-disaster service restoration is considered, which addresses the maximum restoration time of all the customers as well as the total weighted wait time incurred by the customers. To the best of our knowledge, this is the first time that both objectives, which in some cases may contradict each other, are addressed together for the problem like Maximum Collection Problem, Minimum Latency Problem, Traveling Repairman Problem. In this thesis, three methods are proposed to solve the problem. An MILP model is proposed to solve the problem optimally, but as it cannot solve large problems in a reasonable amount of time, a Multi-Stage Metaheuristic method is proposed which can solve large instances in less time. Moreover, to take advantage of the optimality of MILP and computation time of Multi-Stage Metaheuristic method, a novel Hybrid method is proposed. To the best of our knowledge, this Hybrid method is proposed for the first time to solve this kind of problem. Additionally, a novel initial routing heuristic is proposed to construct the initial routes. Several improvement steps such as transfer and insertion are also proposed for both Multi-Stage Metaheuristic and Hybrid methods.

The results show that both Metaheuristic and Hybrid method yield near optimum solutions for small instances, in which the Hybrid method was able to achieve optimality more often than Multi-Stage Metaheuristic method. Because of the NP-hardness of the problem, MILP model was not able to solve large instances. By using appropriate settings for tabu-search parameters, the Hybrid method achieves better restoration time than Multi-Stage Metaheuristic method for medium instances. However, because of the contradictory behavior of both the objectives in some cases, Multi-Stage Metaheuristic method was able to yield a better weighted wait time than the Hybrid method in few instances. As the Hybrid method includes an optimum MILP method in some parts, it could not solve large instances in a reasonable amount of time, whereas the Multi-Stage Metaheuristic method solves the problem in less computation time than the Hybrid method. The performance of the tabu search metaheuristic was also tested on small, medium and large instances and it was performing significantly well to give a better combination of both objectives.

The Hybrid method takes more time than Multi-Stage Metaheuristic method, but yields better solutions. Therefore, it can be used to achieve better solutions for big problems which cannot be solved by the optimum method. The proposed Hybrid method can be regarded as a bridge between Metaheuristic and Optimum method.

5.2 Future research

In this section, we will discuss some potential future research which may improve the methodologies proposed in this thesis, as well as the scenarios which may be addressed with the existing problem to make it applicable in more areas.

- In this research, the multi-objective function is to minimize the overall restoration time while minimizing the total weighted wait time of all customers. However, due to widespread of service distortion after disasters, a practical approach will be to only aim at the certain higher percentage of the outage (say 90%) in the optimization process.
- In this research, the efficiencies of all the PLTs are considered to be the same, but in real life the efficiency of each technician is different. Also, different outages may require different skills and equipment. Therefore, a potential extension of this work is to consider heterogeneous fleet in which the skills and efficiency of all PLTs are different.

- Some research is needed on how to select the importance factor for both the objectives. More research could also be done to improve the MILP model for handling bigger sizes of the problem.
- The technique used to select the tabu search parameters in this research is exclusively for testing purpose. As these parameters may affect the performance of tabu search, an extensive research is needed to propose the selection of these parameters according to the characteristics of the problem. Moreover, in this research, the optimum objective values are constrained to the results obtained from the Metaheuristic and Hybrid method. Therefore, a better way is needed to fully investigate the optimality achieved by Metaheuristic and Hybrid methods.
- The performance of Metaheuristic and Hybrid method greatly depends on the improvement steps within them. In this thesis, the transfer and insertion steps are proposed for inter-route improvement. More research can be done to increase the efficiency of the proposed improvement steps or new improvement steps can be used.

References

Afrati, F., Cosmadakis, S., Papadimitrious, C. H., Papageorgiou, G., & Papakostantinou, N. (1986). The complexity of the travelling repairman problem. *Theoretical Informatics and Applications, 20*(1), 79-87.

Agrawal, Y., Mathur, K., & Salkin, H. M. (1989). A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks, 19*(7), 731-749. doi:10.1002/net.3230190702

Akpinar, S. (2016). Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Systems with Applications, 61*, 20-38. doi:10.1016/j.eswa.2016.05.023

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

Altinel, I. K., & O¨ncan, 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.

Altinkemer, K., & Gavish, B. (1991). Parallel savings based heuristics for the delivery problem. *Operations Research, 39*(3), 456-469.

Annouch, A., Bellabdaoui, A., & Minkhar, J. (2016). Split delivery and pickup vehicle routing problem with two-dimensional loading constraints. *11th International Conference on Intelligent Systems: Theories and Applications, SITA 2016.* Mohammedia, Morocco: Institute of Electrical and Electronics Engineers Inc. doi:10.1109/SITA.2016.7772277

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2007). *The Traveling Salesman Problem. A Computational Study.* Princeton, NJ: Princeton University Press.

Archer, A., Levin, A., & Williamson, D. P. (2007). A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing, 37*(5), 1472-1498.

Archetti, C., & Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research, 19*(1/2), 3-22.

Arora, S., & Karakostas, G. (2003). Approximation schemes for minimum latency problems. *SIAM Journal on Computing, 32*(5), 1317-1337.

Azi, N., Gendreau, M., & Potvin, J. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research, 202*(3), 756-763. doi:10.1016/j.ejor.2009.06.034

Azi, N., Gendreau, M., & Potvin, J. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers and Operations Research, 41*(1), 167-173. doi:10.1016/j.cor.2013.08.016

Baker, B. M., & Sheasby, J. (1999). Extensions to the generalised assignment heuristic for vehicle routing. *European Journal of Operational Research, 119*(1), 147-157.

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.

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.

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.

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

Baskan, O., Haldenbilen, S., Ceylan, H., & Ceylan, H. (2009). A new solution algorithm for improving performance of ant colony optimization. *Applied Mathematics and Computation, 211*(1), 75-84. doi:10.1016/j.amc.2009.01.025

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., & Sudan, M. (1994). Minimum latency problem. *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing* (pp. 163-171). Montreal, Canada: Publ by ACM, New York, NY, United States.

Braaten, S., Gjønnes, O., Hvattum, L. M., & Tirado, G. (2017). Heuristics for the robust vehicle routing problem with time windows. *Expert Systems with Applications, 77*, 136-147.

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science, 39*(1), 119-139. doi:10.1287/trsc.1030.0057

Butt, S. E., & Ryan, D. M. (1998). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers and Operations Research, 26*(4), 427-441.

Campbell, A. M., Vandenbussche, D., & Hermann, W. (2008). Routing for relief efforts. *Transportation Science, 42*(2), 127-145.

Chaudhuri, K., Godfrey, B., Rao, & Talwar, K. (2003). Paths, trees, and minimum latency tours. *44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003*, (pp. 36-45). Cambridge, MA, United states. doi:10.1109/SFCS.2003.1238179

Chen, P., Dong, X., & Niu, Y. (2012). An iterated local search algorithm for the cumulative capacitated vehicle routing problem. *Advances in Intelligent and Soft Computing, 136,* 575-581. doi:10.1007/978-3-642-27711-5_76

Chen, P., Golden, B., Wang, X., & Wasil, E. (2017). A novel approach to solve the split delivery vehicle routing problem. *International Transactions in Operational Research, 24*(1-2), 27-41.

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

Christofides, N. (1976). The Vehicle Routing Problem. *RAIRO (Recherche opérationnelle), 10*(2), 55-70.

Christofides, N., & Eilon, S. (1969, September). An Algorithm for the Vehicle-dispatching Problem. *Journal of the Operational Research Society, 20*(3), 309-318. doi:10.1057/jors.1969.75

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

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

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*(4), 568-582.

Contardo, C., & Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization, 12*(1), 129-146. doi:10.1016/j.disopt.2014.03.001

Crevier, B., Cordeau, J.-F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research, 176*(2), 756-773. doi:10.1016/j.ejor.2005.08.015

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science, 6*(1), 80-91. doi:10.1287/mnsc.6.1.80

Davis, R., Kendrick, D., & Weitzman, M. (1971). Branch- and- bound algorithm for zero- one mixed integer programming problems. *Operations Research, 9*(4), 1036-1044.

Dayarian, I., Crainic, T. G., Gendreau, M., & Rei, W. (2015). A branch-and-price approach for a multi-period vehicle routing problem. *Computers and Operations Research, 55*, 167-184. doi:10.1016/j.cor.2014.06.004

Demir, E., Bektas, T., & Laporte, G. (2014). The bi-objective Pollution-Routing Problem. *European journal of operational research, 232*(3), 464-478.

Derigs, U., & Reuter, K. (2009). A simple and efficient tabu search heuristic for solving the open vehicle routing problem. *Journal of the Operational Research Society, 60*(12), 1658-1669. doi:10.1057/jors.2008.107

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.

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. doi:10.1016/j.cor.2013.01.003

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

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53-66.

Ekici, A., & Retharekar, A. (2013). Multiple agents maximum collection problem with time dependent rewards. *Computers and Industrial Engineering, 64*(4), 1009-1018.

Ergun, O., Orlin, J. B., & Steele-Feldman, F. (2006). Creating very large scale neighborhoods out of smaller ones by compounding moves. *Journal of Heuristics, 12*(1), 115-140.

Erkut, E., & Zhang, J. (1996). Maximum collection problem with time-dependent rewards. *Naval Research Logistics, 43*(5), 749-763.

Fakcharoenphol, J., Harrelson, C., & Rao, S. (2007). The k-traveling repairmen problem. *ACM Transactions on Algorithms, 3*(4). doi:10.1145/1290672.1290677

Feng, Y., Zhang, R., & Jia, G. (2017). Vehicle Routing Problems with Fuel Consumption and Stochastic Travel Speeds. *Mathematical Problems in Engineering*. doi:10.1155/2017/6329203

Finkel'shteyn, Y. (1971). Method of Cuts and Branches for solving Integer Linear Programming Problems. *Engineering Cybernetics, 9*(4), 619-623.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum. *Operations Research, 42*(4), 626-642.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks, 11*(2), 109-124. doi:10.1007/s11590-015-0878-3

Flood, M. M. (1956). Traveling-salesman problem. *Operations Research Society of America, 4*(1), 61-75.

Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. *Operations Research, 27*(2), 367-384.

Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., 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.

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

Gao, Y., Wang, C., & Liu, C. (2016). An archived multi-objective simulated annealing algorithm for vehicle routing problem with time windows. *International Journal of u- and e-Service, Science and Technology, 9*(12), 187-198. doi:10.14257/ijunesst.2016.9.12.17

García, A., Jodrá, P., & Tejel, J. (2002). A note on the traveling repairman problem. *Networks, 40*(1), 27-31.

Garey, M. R., & Johnson, D. S. (1979). *A guide to the theory of NP-completeness.* New York, USA: W.H. Freeman & Co Ltd.

Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, E. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers and Operations Research, 26*(12), 1153-1173. doi:10.1016/S0305-0548(98)00100-2

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533-549.

Goemans, M., & Kleinberg, J. (1998). An improved approximation ratio for the minimum latency problem. *Mathematical Programming, 82*(1-2), 111-124.

Goetschalckx, M., & Jacobs-Blecha, C. (1989). Vehicle routing problem with backhauls. *European Journal of Operational Research, 42*(1), 39-51. doi:10.1016/0377-2217(89)90057-X

Golden, B. L., Magnanti, T. L., & Nguyen, H. Q. (1977). Implementing vehicle routing algorithms. *Networks, 7*(2), 113-148. doi:10.1002/net.3230070203

Gronalt, M., Harti, R., & 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. doi:10.1016/S0377-2217(02)00650-1

Gutjahr, W. J., & Nolz, P. C. (2016). Multicriteria optimization in humanitarian aid. *European Journal of Operational Research, 252*(2), 351-366.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 28*(1), 100-108.

Hiller, F. S., & Lieberman, G. J. (2015). Integer Programming. In *Introduction to Operations research* (pp. 474-532). New-York: McGraw-Hill Education.

Hillier, F. S., & Lieberman, G. J. (2015). Genetic Algorithm. In *Introduction to Operations Research* (Tenth ed., pp. 645-655). New York: McGraw-Hill.

Jordan, W. C., & Burns, L. D. (1984). Truck backhauling on two terminal networks. *Transportation Research, 18B*, 487-503.

Jothi, R., & Raghavachari, B. (2007). Approximating the k-traveling repairman problem with repairtimes. *Journal of Discrete Algorithms, 5*(2), 293-303.

Kallehauge, B. (2008, July). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers and Operations Research, 35*(7), 2307-2330. doi:10.1016/j.cor.2006.11.006

Karakatič, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing, 27*, 519-532.

Ke, L., & Feng, Z. (2013). A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 40*(2), 633–638.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*(4598), 671-680.

Koç, C., & Karaogˇlan, G. (2012). A mathematical model for the vehicle routing problem with time windows and multiple use of vehicles. *Journal of the Faculty of Engineering and Architecture of Gazi University, 27*(3), 569-576.

Kulkarni, R., & Bhave, P. (1985). Integer programming formulations of vehicle routing problems. *European Journal of Operational Research, 20*(1), 58-67.

Kumar, Y., & Jain, S. (2015). School bus routing based on branch and bound approach. *IEEE International Conference on Computer, Communication and Control, IC4 2015.* Indore, India: Institute of Electrical and Electronics Engineers Inc. doi:10.1109/IC4.2015.7375684

Kurniawan, R., Sulistiyo, M., & Wulandari, G. (2015). Genetic Algorithm for Capacitated Vehicle Routing Problem with considering traffic density. *2nd International Conference on Information Technology Systems and Innovation, ICITSI 2015.* Bandung, Bali, Indonesia: Institute of Electrical and Electronics Engineers Inc. doi:10.1109/ICITSI.2015.7437695

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers and Operations Research, 34*(9), 2743-2757. doi:10.1016/j.cor.2005.10.010

Laporte, G. (1992). Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*(3), 345-358.

Laporte, G. (2009). A Concise Guide to the Traveling Salesman Problem. *The Journal of the Operational Research Society, 61*(1), 35-40.

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

Laporte, G., & Nobert, Y. (1987). Exact Algorithms for the Vehicle Routing Problem. *North-Holland Mathematics Studies, 132*, 147-184.

Laporte, G., Mercure, H., & Nobert, Y. (1986). Exact Algorithm for th Asymmetrical Capacitated Vehicle Routing Problem. *Networks, 16*(1), 33-46.

Laporte, G., Nobert, Y., & Desrochers, M. (1985). Optimal Routing Under Capacity and Distance Restrictions. *Operations research, 33*(5), 1050-1073. doi:10.1287/opre.33.5.1050

Laporte, G., Nobert, Y., & Taillefer, S. (1988). Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science, 22*(3), 161-172.

Lenstra, J. K., & Rinnooy Kan, A. (1987). Complexity of vehicle routing and scheduling problem. *Networks, 11*(2), 221-227.

Letchford, A. N., & Salazar-González, J.-J. (2015). Stronger multi-commodity flow formulations of the Capacitated Vehicle Routing Problem. *European Journal of Operational Research, 244*(3), 730-738.

Li, H., Wang, L., Hei, X., Li, W., & Jiang, Q. (2017). A decomposition-based chemical reaction optimization for multi-objective vehicle routing problem for simultaneous delivery and pickup with time windows. *Memetic Computing*, 1-18.

Li, X., Tian, P., & Leung, S. (2009). An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. *Journal of the Operational Research Society, 60*(7), 1012-1025. doi:10.1057/palgrave.jors.2602644

Lin, S.-W., Ying, K., Lee, Z., & Hsi, F. (2007). Applying simulated annealing approach for capacitated vehicle routing problems. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics.* Taipei, Taiwan: Institute of Electrical and Electronics Engineers Inc., 3 Park Avenue, 17th Floor, New York, NY 10016-5997, United States.

Little, J., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research, 11*(6), 972-989. doi:10.1287/opre.11.6.972

Liu, S., Liu, L., & Zhang, T. (2008). Variable neighborhood search for solving vehicle routing problems with backhauls and time windows. *Dongbei Daxue Xuebao/Journal of Northeastern University, 29*(3), 316-319.

Lysgaard, J., & Wøhlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research, 236*(3), 800-810.

Mahmoudi, M., & Zhou, X. (2016). Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state-space-time network representations. *Transportation Research Part B: Methodological, 89*, 19-42. doi:10.1016/j.trb.2016.03.009

Mancini, S. (2016). A real-life Multi Depot Multi Period Vehicle Routing Problem with a Heterogeneous Fleet: Formulation and Adaptive Large Neighborhood Search based Matheuristic. *Transportation Research Part C: Emerging Technologies, 70*, 100-112. doi:10.1016/j.trc.2015.06.016

Marsten, R. (1974). Algorithm for large set partitioning problems. *Management Science, 20*(5), 774-787.

Mattos, R., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers and Operations Research, 39*(3), 728-735. doi:10.1016/j.cor.2011.05.005

Mavrovouniotis, M., & Yang , S. (2015). Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. *Information Sciences, 294*, 456-477.

Méndez-Díaz, I., Zabala, P., & Lucena, A. (2008). A new formulation for the Traveling Deliveryman Problem. *Discrete Applied Mathematics, 156*(17), 3223-3237. doi:10.1016/j.dam.2008.05.009

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

Mocková, D., & Rybicˇková, A. (2014). Application of genetic algorithms to vehicle routing problem. *Neural Network World, 24*(1), 57-78.

Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers and Industrial Engineering, 79*, 115-129.

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.

Ngueveu, S. U., Prins, C., & Wolfler Calvo, R. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 37*(11), 1877-1885.

Noble, C. E. (1957). Simplex method for solving linear programming problems. *Industrial Quality Control, 13*(9), 5-9.

Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., & Moreno-Vega, J. (2016). Mixed integer formulation and an efficient metaheuristic procedure for the k-Travelling Repairmen Problem. *Journal of the Operational Research Society, 67*(8), 1121-1134.

Oesterle, J., & Bauernhansl, T. (2016). Exact Method for the Vehicle Routing Problem with Mixed Linehaul and Backhaul Customers, Heterogeneous Fleet, time Window and Manufacturing Capacity. *48th CIRP International Conference on Manufacturing Systems, CIRP CMS 2015. 41*, pp. 573-578. Ischia, Italy: Elsevier.

Og˘uz, O. (2002). Generalized column generation for linear programming. *Management Science, 48*(3), 444-452. doi: 10.1287/mnsc.48.3.444.7729

Ouaddi, K., Benadada, Y., & Mhada, F. (2016). Multi period dynamic vehicles routing problem: Literature review, modelization and resolution. *3rd IEEE International Conference on Logistics Operations Management, GOL 2016.* Fes, Morocco: Institute of Electrical and Electronics Engineers Inc. doi:10.1109/GOL.2016.7731706

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

Pecin, D., Pessoa, A., Poggi, M., & Uchoa, E. (2017, March 1). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation, 9*(1), 61-100.

Polacek, M., Hartl, R., Doerner, K., & Reimann, M. (2004). AVariable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of Heuristics, 10*(6), 613-627. doi:10.1007/s10732-005-5432-5

Post, I., & Swamy, C. (2015). Linear programming-based approximation algorithms for multi-vehicle minimum latency problems. *26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, (pp. 512-531). San Diego, CA, United states.

Potter, T., & Bossomaier, T. (1995). Solving vehicle routing problems with genetic algorithms. *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation. Part 1 (of 2)* (pp. 788-793). Perth, Australia: IEEE, Piscataway, NJ, United States.

Rao, M. R., & Zionts, S. (1968). Allocation of transportation units to alternative trips—A column generation scheme with out-of-kilter subproblems. *Operations Research, 16*(1), 52-63.

Ribeiro, G. M., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 39*(3), 728-735.

Rivera, J. C., Murat Afsar, H., & Prins, C. (2016). Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem. *European Journal of Operational Research, 249*(1), 93-104. doi:10.1016/j.ejor.2015.08.067

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

Rochat, Y., & Taillard, E. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics, 1*(1), 147-167.

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research, 171*(3), 750-775.

Salehipour, A., Sörensen, K., Goos, P., & Bräysy, O. (2011). Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem. *4OR: A Quarterly Journal of Operations Research, 9*(2), 189-209.

Seidgar, H., Abedi, M., Rad, S., & Rezaeian, J. (2016). An efficient hybrid of genetic and simulated annealing algorithms for multi server vehicle routing problem with multi entry. *International Journal of Industrial and Systems Engineering, 24*(3), 333-360.

Silva, M. M., Subramanian, A., Thibaut, V., & Ochi, L. S. (2012). A simple and effective metaheuristic for the Minimum Latency Problem. *European Journal of Operational Research, 221*(3), 513-520.

Silvestrin, P., Paulo, V., & Ritt, M. (2017). An iterated tabu search for the multi-compartment vehicle routing problem. *Computers and Operations Research, 81*, 1925-202.

Solomon, M. M., & Desrosiers, J. (1988). Time window constrained routing and scheduling problems. *Transportation Science, 22*(1), 1-13.

Steinhaus, M., Shirazi, A. N., & Sodhi, M. (2015). Modified self organizing neural network algorithm for solving the Vehicle Routing Problem. *18th IEEE International Conference on Computational Science and Engineering, CSE 2015* (pp. 246-252). Porto, Portugal: Institute of Electrical and Electronics Engineers Inc.

Ting, C.-J., & Huang, C.-H. (2005). An improved genetic algorithm for vehicle routing problem with time windows. *International Journal of Industrial Engineering : Theory Applications and Practice, 12*(3), 218-228.

Torki, A., Somhon, S., & Enkawa, T. (1997). Competitive Neural Network algorithm for solving Vehicle Routing Problem. *Computers and Industrial Engineering, 33*(3-4), 473-476. doi:10.1016/S0360-8352(97)00171-X

Toth, P., & Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics, 123*(1-3), 487-512.

Toth, P., & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications.* Society for Industrial and Applied Mathematics (SIAM).

Uchoa, E., Pecin, D., Pessoa, A., Vidal, T., & Subramanian, A. (2017). New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research, 257*(3), 845-858. doi:10.1016/j.ejor.2016.08.012

Wang, X., Choi, T., Liu, H., & Yue, X. (2016). Novel Ant Colony Optimization Methods for Simplifying Solution Construction in Vehicle Routing Problems. *IEEE Transactions on Intelligent Transportation Systems, 17*(11), 3132-3141. doi:10.1109/TITS.2016.2542264

Wang, Z., & Wang, Z. (2009). A novel two-phase heuristic method for vehicle routing problem with backhauls. *Computers and Mathematics with Applications, 57*(11-12), 1923-1928.

Wassan, N., Wassan, N., Nagy, G., & Salhi, S. (2017). The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and a Two-Level Variable Neighbourhood Search. *Computers and Operations Research, 78*, 454-467. doi:10.1016/j.cor.2015.12.017

Wei, L., Zhang, Z., Zhang, D., & Lim, A. (2015). A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research, 243*(3), 798-814. doi:10.1016/j.ejor.2014.12.048

Wu, B. Y. (2000). Polynomial time algorithms for some minimum latency problems. *Information Processing Letters, 75*(5), 225-229.

Wu, B. Y., Huang, Z., & Zhan, F. (2004). Exact algorithms for the minimum latency problem. *Information Processing Letters, 92*(6), 303-309.

Yu, V., A.A.N., P., Hidayat, Y., & Wibowo, O. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing Journal, 53*, 119-132. doi:10.1016/j.asoc.2016.12.027

Zhang, D., Cai, S., Ye, F., Si, Y., & Nguyen, T. (2017). A hybrid algorithm for a vehicle routing problem with realistic constraints. *Information Sciences, 394-395*, 167-182. doi:10.1016/j.ins.2017.02.028

Appendices

Appendix A. Performance of tabu search Metaheuristic

Small Size				Restoration time (hour)			Weighted wait time (hour)			CPU Runtime (sec)
				Initial	Final	Improvement	Initial	Final	Improvement	Total
2 Depots	7 Outages	2 PLT	1	9.78	7.27	+25.66%	1275.52	1074.23	+15.78%	0.15
			2	8.83	7.47	+15.40%	1110.59	1146.78	-3.26%	0.15
			3	13.69	10.26	+25.08%	2067.55	1850.04	+10.52%	0.39
			4	8.99	8.37	+6.91%	1858.27	1858.84	-0.03%	0.25
			5	10.23	7.67	+25.04%	2163.66	1873.53	+13.41%	0.14
			Avg.	10.31	8.21	+19.62%	1695.12	1560.69	+7.28%	0.21
			Std. Dev.	1.98	1.22	0.08	475.02	411.84	0.08	0.10
				Initial	Final	Improvement	Initial	Final	Improvement	Total
3 Depots	10 Outages	3 PLT	1	13.91	8.65	+37.80%	2311.65	2040.08	+11.75%	0.28
			2	12.66	9.52	+24.81%	2783.85	2637.14	+5.27%	0.25
			3	10.69	8.70	+18.61%	2692.28	2524.40	+6.24%	0.32
			4	8.36	8.10	+3.12%	2012.01	1863.11	+7.40%	0.25
			5	15.07	7.75	+48.57%	1884.77	1647.46	+12.59%	0.42
			Avg.	12.14	8.55	+26.58%	2336.91	2142.44	+8.65%	0.30
			Std. Dev.	2.66	0.68	0.18	398.96	425.48	0.03	0.07
Medium Size				Restoration time (hour)			Weighted wait time (hour)			CPU Runtime (sec)
				Initial	Final	Improvement	Initial	Final	Improvement	Total
10 Depots	100 Outages	40 PLT	1	20.06	11.39	+43.24%	17515.01	15057.93	+14.03%	897.64
			2	19.00	11.43	+39.85%	25994.79	21963.10	+15.51%	1708.85
			3	17.42	10.32	+40.78%	20134.32	20388.13	-1.26%	606.38
			4	25.02	11.51	+53.99%	55394.87	49386.97	+10.85%	787.82
			5	19.63	11.95	+39.12%	28661.79	30836.33	-7.59%	14.15
			Avg.	20.23	11.32	+43.40%	29540.16	27526.49	+6.31%	802.97
			Std. Dev.	2.86	0.60	0.06	15123.50	13474.12	0.10	610.47
				Initial	Final	Improvement	Initial	Final	Improvement	Total
10 Depots	100 Outages	40 PLT	1	14.41	6.86	+52.42%	45342.57	49013.24	-8.10%	30.37
			2	15.48	7.08	+54.28%	59042.98	59688.85	-1.09%	29.63
			3	15.30	7.09	+53.68%	35786.04	38054.24	-6.34%	19.59
			4	14.68	6.33	+56.85%	44418.75	44715.23	-0.67%	24.90
			5	11.01	6.71	+39.06%	51380.39	52557.04	-2.29%	25.57
			Avg.	14.18	6.81	+51.26%	47194.15	48805.72	-3.70%	26.01
			Std. Dev.	1.82	0.31	0.07	8648.64	8137.64	0.03	4.32
Large Size				Restoration time (hour)			Weighted wait time (hour)			CPU Runtime (sec)
				Initial	Final	Improvement	Initial	Final	Improvement	Total
31 Depots	500 Outages	140 PLT	1	20.63	10.92	+47.05%	254595.65	272510.51	-7.04%	897.64
			2	19.84	9.92	+50.01%	244543.49	256507.72	-4.89%	1708.85
			3	18.90	11.13	+41.14%	280238.95	283295.92	-1.09%	606.38
			4	21.65	10.32	+52.35%	230950.91	253376.90	-9.71%	787.82
			5	22.42	10.07	+55.10%	219805.08	233395.91	-6.18%	1165.34
			Avg.	20.69	10.47	+49.13%	246026.81	259817.39	-5.78%	1033.20
			Std. Dev.	1.40	0.53	0.05	23243.05	19123.84	0.03	428.58
				Initial	Final	Improvement	Initial	Final	Improvement	Total
31 Depots	600 Outages	140 PLT	1	29.13	12.46	+57.22%	348720.48	378682.93	-8.59%	1920.94
			2	22.83	11.90	+47.88%	320817.44	347875.77	-8.43%	3122.47
			3	18.47	12.65	+31.49%	332047.93	341126.78	-2.73%	982.64
			4	27.66	11.74	+57.55%	316247.71	332132.93	-5.02%	3359.78
			5	22.07	11.45	+48.12%	296791.54	335916.60	-13.18%	2673.97
			Avg.	24.03	12.04	+48.45%	322925.02	347147.00	-7.59%	2411.96
			Std. Dev.	4.34	0.50	0.11	19239.41	18593.23	0.04	969.01

Table 1. Results- before and after applying the tabu-search metaheuristic