2019

Solving arc routing problems for winter road maintenance operations

Luning Zhang
Iowa State University

Solving Arc Routing Problems for Winter Road Maintenance Operations

by

Luning Zhang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Civil Engineering (Transportation)

Program of Study Committee:
Jing Dong, Major Professor
Christopher Day
Shauna Hallmark
Sigurdur Olafsson
Jennifer Shane

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

NOMENCLATURE

ARP	arc routing problem
AVL	automatic vehicle location
CARP	capacitated arc routing problem
CARPIF	capacitated arc routing problem and intermediate facilities
CARPTIF	capacitated arc routing problem with time constraint and intermediate facilities
CPP	Chinese postman problem
DOT	Department of Transportation
LOS	level of service
MA	memetic algorithm
MDCARP	multiple depots capacitated arc routing problem
MDCARPIF	multi-depot capacitated arc routing problem with intermediate facilities
MDWRMRPIF	multi-depot winter road maintenance routing problem with intermediate facilities
RAMS	roadway asset management system
RPP	rural postman problem
SDWRMRP	single depot winter road maintenance routing problem
VRP	vehicle routing problem

ACKNOWLEDGEMENT

First, my deepest gratitude goes to my advisor Dr. Jing Dong for her expertise, assistance, guidance, and patience throughout my doctoral period. Dr. Dong provided me extensive professional advice and inspired me by her hardworking and passionate attitude. This work would not have been possible without her support.

I am grateful to all my dissertation committees, Dr. Shauna Hallmark, Dr. Christopher Day, Dr. Sigurdur Olafsson, and Dr. Jennifer Shane, for their great support and invaluable advice. Their insightful comments and tremendous experience help me shaped my final dissertation.

I am thankful for those with whom I have had the pleasure to work during this and other related projects. I would like to thank Shannon Morrissey and Bryce Hallmark for their help of preparing data for this research.

Finally, I would like to express my gratitude to all my close friends and family for your warm love, deep trust, and endless support.

ABSTRACT

For winter road maintenance, a fleet of snowplow trucks is operated by government agencies to remove snow and ice on roadways and spread materials for anti-icing, de-icing, or increasing friction. Winter road maintenance is essential for providing safe and efficient service for road users (Usman et al., 2010). It is also costly due to the high cost of equipment, crew, and materials. Optimizing winter road maintenance operations could result in significant cost savings, improved safety and mobility, and reduced environmental and social impacts (Salazar-Aguilar et al., 2012).

The first topic in this study focuses on designing routes for winter maintenance trucks from a single depot. Real-world winter road maintenance constraints, including road segment service cycle time, heterogeneous vehicle capacity, fleet size, and road-vehicle dependency, are taken into consideration. The problem is formulated as a variation of the capacitated arc routing problem (CARP) to minimize total travel distance. A metaheuristic algorithm, memetic algorithm (MA), is developed to find nearly optimal solutions. This is the first study that developed the model that includes all the constraints listed. This is the first study that used the MA to solve the routing problem with all those constraints, and the first study that developed the route split procedure that satisfies all those constraints. In addition, a paralleled metaheuristic algorithm is proposed to enhance the solution quality and computation efficiency.

The second topic of this study focuses on designing routes from multiple depots with intermediate facilities. The service boundaries of depots are redesigned. Each truck must start and end at its home depot, but they can reload at other depots or reload stations (i.e., intermediate facilities). This problem is a variation of the multi-depot CARP with

intermediate facilities (MDCARPIF). The second topic includes all constraints employed in the first topic. Since the trucks can be reloaded at any stations, a constraint that restricts the length of work time for truck drivers is also included in this topic. This is the first study that developed the model that includes all the constraints listed. This is the first study that uses the MA to solve the problem and the first study that developed the route split procedure that satisfies all those constraints.

The proposed algorithms are implemented to solve real-world problems. Deadhead (travelling without servicing) speed, service speed, and the spreading rate are estimated by the sample from historical winter road maintenance data.

Eighteen traffic networks are used as instances for the first topic. The optimized route in the first topic reduced 13.2% of the deadhead distance comparing to the current practice. Comparing to the single core result, the parallel computation improved the solution fitness on 2 of the 18 instances tested, with slightly less time consumed.

Based on the optimized result in the first topic, the reduction of the deadhead distance of the second topic is insignificant. This could be due to the network structure and depot location of the current operation. A test instance is created to verify the effectiveness of the proposed algorithm. The result shows 10.4% of deadhead distance can be saved by using the reload and multiple depot scenario instead of the single depot scenario on the test instance.

CHAPTER 1. INTRODUCTION

1.1 Background

In the context of winter road maintenance, a fleet of snowplow trucks is operated by government agencies to remove snow and ice on roadways and spread materials for anti-icing, de-icing, or increasing friction. Winter road maintenance is essential for providing safe and efficient service for road users (Usman et al., 2010). It is also costly due to the high cost of equipment, crew, and materials. According to a recent survey by the American Association of State Highway and Transportation Officials (AASHTO), 23 reporting states spent approximately $1.131 billion from October 2014 to mid-April 2015 to pretreat, plow, and spread chemicals and other materials on roadways (ASSHTO, 2018). Optimizing winter road maintenance operations could result in significant cost savings, improved safety and mobility, and reduced environmental and social impacts (Salazar-Aguilar et al., 2012).

Winter road maintenance operations involve a variety of decision-making problems at the strategic, tactical, operational, and real-time level. In the strategic level, decisions involve long-lasting resource allocation, such as depot and reload station constructions. On a tactical level, medium-term decisions are usually updated every few months. For example, the district design involves partitioning a sizable geographical region into districts such that each district contains one depot. On an operational level, short-term decisions are required to be updated daily. Routing and scheduling of vehicles and the staffing of crews belong to the operational level. Last, the real-time level decision-making handles situations that must respond within a tight period (e.g., minutes) due to a sudden change of the situation. The modification of routes based on incoming new weather conditions while the vehicle is in operation is an example of a real-time decision.

1.2 Winter Maintenance Practices

Different snowstorm fighting practices might be adopted depending on the temperature, precipitation, and road surface conditions. The snowstorm fighting practices include plowing alone, spreading materials alone, plowing and spreading consecutively, and plowing and spreading simultaneously. The Salt Institute proposed service practices, as shown in Table 1.1 (Salt Institute, 2013). These practices are widely employed among winter maintenance agencies, such as the Village of Algonquin (2017). Other public maintenance agencies may have different criteria, but the idea of storm fighting practices coincides in treating different weather with different operations.

The spreading rates in Table 1.1 are different under five different conditions. For the same capacity, the maximum distances a truck can spread material on would be different if the spread rate were different. Thus, the optimal routes would be different if a different spread rate was assumed.

First, agencies could design routes based on various spreading rates. Designing routes based on various predefined spreading rates could be seen as the "wait-and-see" concept in the context of stochastic programming. Optimal solution routes can be found in all situations, but the issue is a truck may have a different route under different spreading rates.

Second, agencies could design routes under a predefined high spreading rate. The routes are short and unique for each truck. Such routes can complete demanded service under most weather conditions. However, if the maximum spreading rate is used, it leads to underutilization of the truck capacity and could significantly increase the deadhead distance.

Third, agencies could design the routes to minimize the expectation of total deadhead distance (or any other objective) under various weather conditions. In the context of stochastic programming, this is called the "recourse problem." However, compared to the

second approach, the third approach might generate routes where trucks will have insufficient capacity to complete the service in one run. When it occurs, the recourse procedure must be performed, and more truck runs are involved (Gendreau et al. 1996).

Other route design preferences exist. Nevertheless, routes still need to be designed for each scenario under the appropriate spreading rate. The second approach is employed in this study. This approach is more practical, as truck drivers only need to memorize and become familiar with one route. The spreading rate is determined from historical maintenance operation data to cover most winter storm cases. The solution approach of this study can also serve the need for the first approach by changing the spreading rate to different values.

1.3 Problem Statement

The route for a winter road maintenance truck determines which road segment the truck should service, and what path should the truck follow. Each truck has different service capacity and capability, each road segment has different service expectations, and each depot has a specific number of trucks. It is desirable to develop a systematic way to determine the route with the least cost and meet all the constraints involved.

The first topic in this study focuses on designing routes for winter maintenance trucks for single depots. Roadways with higher traffic should be serviced more frequently during the snowstorm condition. Road segment service cycle time indicates this service frequency. Roadways with a median wide enough can have the snow and ice plowed to the median, but roadways with no median can only have the snow plowed to its shoulder. Therefore, a road-vehicle dependency exists and should be considered. In addition, depots have a different number of maintenance trucks, and each type of truck has a different capacity. All of these constraints are taken into consideration for the routing problem in the first topic.

Table 1.1 *Snowstorm fighting practices proposed by the Salt Institute (Salt Institute, 2013)*

Storm fighting Practices	
Condition 1 **Temperature** Near 30 **Precipitation** Snow, sleet, or freezing rain **Road Surface** Wet	If snow or sleet, apply salt at 500 lb. per two-lane mile. If snow or sleet continues and accumulates, plow and salt simultaneously. If freezing rain, apply salt at 200 lb. per two-lane mile. If the rain continues to freeze, reapply salt at 200 lb. per two-lane mile. Consider anti-icing procedures.
Condition 2 **Temperature** Below 30 or falling **Precipitation** Snow, sleet, or freezing rain **Road Surface** Wet or Sticky	Apply salt at 300-800 lb. per two-lane mile, depending on accumulation rate. As snowfall continues and accumulates, plow and repeat salt application. If freezing rain, apply salt at 200-400 lb. per two-lane mile. Consider anti-icing and de-icing procedures as warranted.
Condition 3 **Temperature** Below 20 and falling **Precipitation** Dry Snow **Road Surface** Dry	Plow as soon as possible. Do not apply salt. Continue to plow and patrol to check for wet, packed, or icy spots; treat them with heavy salt applications.
Condition 4 **Temperature** Below 20 **Precipitation** Snow, sleet, or freezing rain **Road Surface** Wet	Apply salt at 600-800 lb. per two-lane mile, as required. If snow or sleet continues and accumulates, plow and salt simultaneously. If the temperature starts to rise, apply salt at 500-600 lb. per two-lane mile, wait for the salt to react before plowing. Continue until safe pavement is obtained.
Condition 5 **Temperature** Below 10 **Precipitation** Snow or freezing rain **Road Surface** Accumulation of packed snow or ice	Apply salt at the rate of 800 lb. per two-lane mile or salt-treated abrasives at a rate of 1500 to 2000 lb. per two-lane mile. When snow or ice becomes mealy or slushy, plow. Repeat application and plowing as necessary.

The second topic of this study focuses on designing routes of multiple depots with intermediate facilities. The depot service boundaries among the multiple depots can be redesigned. Each route must start and end at its home depot, but they can reload at other depots. Given more flexibility to reload the truck at multiple intermediate facilities and reassign the fleet to depots, more improvements to service efficiency are expected. The second topic includes all constraints employed in the first topic. Since the trucks can be reloaded at any stations, a constraint that restricts the length of work time for truck drivers is also included in the second topic.

Route time duration and service distance are restrictions when designing routes. Hence, caution needs to be paid on estimating relevant parameters, including service speed, deadhead speed, and spreading rate.

The outline of the dissertation is as follows. In Chapter 2, the relevant literature associated with the routing problem for winter road maintenance is reviewed. Parameters; including service speed, deadhead speed, and the spreading rate; are estimated in Chapter 3. The test instances are also discussed in Chapter 3. Chapter 4 presents a description of the constraints regarding winter road maintenance. In Chapter 5, the model, solution algorithm, and result of the first topic are provided, succeeded by the model, solution algorithm, and result of the second topic in Chapter 6. The dissertation concludes with remarks and gives suggestions for further research on the topic in Chapter 7.

CHAPTER 2. LITERATURE REVIEW

This chapter summarizes the related literature on the arc routing problem (ARP). In particular, Section 2.1 introduces the ARP. Since the CARP is the most relevant model to describe the routing problem for winter road maintenance, section 2.2 discusses the CARP literature through exact and heuristic approaches. Sections 2.3 and 2.4 investigate literature regarding the topic of the capacitated arc routing problem with intermediate facilities (CARPIF) and multi-depot capacitated arc routing problem (MDCARP), respectively. Section 2.5 reviews the application of ARP for winter road maintenance, where most of them incorporate problem-specific constraints.

2.1 Introduction of Arc Routing Problem

In the ARP, the objective is to determine a least-cost route(s) that services a subset of arcs in a graph, with or without constraints. Generally, there are three central ARPs:

- the Chinese postman problem (CPP), where all edges of a graph demand service to minimize the traversing cost;
- the rural postman problem (RPP), where only a subset of edges demand service in CPP; and
- the CARP, where multiple vehicles can be used, and all vehicles have the same capacity.

Edmonds and Johnson (1973) showed that undirected CPP is solvable in polynomial time, but mixed graph CPP is NP-hard, with some tractable special cases. Lenstra and Rinnooy (1976) proved that RPP is also NP-hard. The CARP was first defined by Golden and Wong (1981) and was shown to be NP-hard. In the same paper, evidence was shown that even computing a factor 3/2 approximation of CARP is NP-hard.

2.2 Capacitated Arc Routing Problem

CARP is similar to the vehicle-routing problem (VRP), whereas CARP needs to traverse service arcs instead of visiting service nodes. In the classical CARP model, the objective is to minimize the total travel distance of all vehicles. The problem considers an undirected graph $G = (V,\ E)$, where V represents the set of vertices and E represents the set of undirected edges that can be traversed in both directions. A subset of required edges $E_r \subseteq E$ must be serviced by a fleet of K vehicles, all with a limited capacity of Q. All routes must start and return to the depot v_0. Those services required that road segments are serviced exactly once and each road segment can only be serviced by at most one vehicle (no split of service by multiple vehicles). The following two sections review the algorithms for CARP.

2.2.1 Exact Algorithms

Various exact algorithms have been developed to solve the classical CARP. Three categories of integer programming methods; namely branch-and-bound, cutting plane and branch-and-cut, and column-generation and branch-and-price; have all been used to solve CARP instances. Several benchmark instance datasets, with the number of arcs ranging from 11 to 375, have been provided to test the performance of algorithms. Information about the datasets and literature results can be found on the website (CARP benchmarks).

The exact algorithms have been implemented to solve CARP optimally on small networks. For example, all integer solutions had been found on the GDB instances (11 – 55 edges). One of the most recent papers can solve every GDB instance within 2 seconds (Bode and Irnich, 2012). Regarding medium-sized networks, such as VAL (39 – 97 edges) and BMCV (35 – 142 edges), recent research can find the optimal integer solutions for most of the instances (Bode and Irnich, 2012; Bartolini et al. 2011). If the integer solution is not found, that means the computation time is beyond a time limit (the common practice is 4

hours). Bode and Irnich (2012) found 31 of 34 optimal integer solutions for the VAL instances, and Bode and Irnich (2012b) found 84 of 100 integer solutions for the BMCV instances. On networks of EGL (98 – 190 edges), optimal integer solutions have been found for only 11 of 24 instances, although the upper bound is close enough to the lower bound.

However, on large-sized networks, i.e., an EGL-Large dataset with 375 edges, exact algorithms are only used to find the lower bounds. No upper bound is provided by exact algorithms. The most recent lower bounds were updated by Martinelli et al. (2011) through cutting planes and separation algorithms. All integer solutions were found by metaheuristic algorithms (Brandão and Eglese, 2008; Mei et al. 2009; Martinelli et al. 2011). An iterative local search algorithm provided by Martinelli et al. (2011) can find solutions with a maximum gap of less than 5%.

It is not a surprise that the exact algorithms face difficulties in solving large-sized networks since CARP is NP-hard. Thus, heuristic algorithms have been proposed to solve the CARP on large networks.

2.2.2 Heuristic Algorithms

Some of the classic constructive heuristics are still broadly used in metaheuristics, for example, augment-merge (Golden and Wong, 1981), path-scanning (Golden et al., 1983), and route-first, cluster-second (Ulusoy, 1985).

Modern metaheuristics have been applied to solve CARP in the last two decades. Many metaheuristics have been proposed, such as simulated annealing (Eglese, 1994), Tabu search (Eglese and Li, 1996; Brandão and Eglese, 2008), variable neighborhood search (Hertz and Mittaz, 2001; Polacek et al., 2008), guided local search (Beullens et al., 2003), greedy randomized adaptive search procedure (GRASP) (Prins and Calvo, 2005; Usberti et

al., 2012), MAs (Lacomme et al., 2001; Liu et al., 2013) and ant colony optimization (ACO) (Santos et al., 2010).

The ACO (Santos et al., 2010), GRASP with evolutionary path relinking (Usberti et al., 2012), and MA with iterative local search (MAILS; Liu et al., 2013) are the most effective algorithms. A detailed comparison can be found in Liu et al. (2013). Testing of benchmark dataset shows that in 10 runs, the average deviation of the best solution of MAILS from the best-known solution among all metaheuristics is 0.00%, 0.09%, and 0.06% on the GDB, VAL, and EGL instances, separately. Also, the average deviation of the average solution of MAILS from the best-known solution among all metaheuristic research is 0.05%, 0.19%, and 0.56% on the GDB, VAL, and EGL instances, separately. Average computation time is 3.5, 32.9, and 504.7 seconds for those three benchmark datasets. The other two algorithms have similar solution qualities. No tests were made on the EGL-Large instances by any of these three studies, but it is believed that a nearly optimal solution can be found within reasonable computation time.

The first MA was proposed by Lacomme et al. (2001). The CARP solution is represented by a giant tour and route delimiters. The giant tour corresponds to a sequence of service demanding arcs that ignore the capacity restriction, that is, an RPP solution. Ulusoy's split procedure (1985) is used to split any giant tour into routes optimally. Since this encoding can represent any potential sequence, one optimal sequence and the corresponding delimiters always exist. The fitness of a chromosome is the cost of the CARP solution. OX crossover and exchange, relocation, and 2-opt moves operator are used. The resulting chromosome replaces an existing one in the worse half of the population.

Later, Lacomme et al. (2004) presented a more efficient version of this MA. A compact implementation of Ulusoy's split is developed by not explicitly generating the auxiliary graph. A partial replacement restart phase is added to the algorithm.

Tang et al. (2009) modified the local search in the MA. The local search now can violate the capacity constraint at the expense of a penalty. A solution's chromosome within a local search is updated if the penalized cost of the new chromosome is better than the old chromosome, even if the new chromosome is infeasible. However, the returned solution from the local search must be feasible. Removing the capacity constraint in the local search allows it to search larger, probably isolated, solution spaces. A new move called merge-split (MS) is introduced. MS selects a subset of routes at random, merges the edges and applies the path-scanning to generate new routes. The MS can be seen as an extension of the local search.

Liu et al. (2013) used the same encoding, split procedure, local search moves, and restart phase moves as Lacomme et al. (2004). They also used a similar penalty for the infeasible solution during the local search, as in Tang et al. (2009). Liu et al. (2013) developed a new crossover operator based on the longest common subsequence. Iterative local search is performed if the cost of the new solution is close enough to the old solutions. In summary, the MA is one of the successive metaheuristics for CARP and still highly active in this research field.

2.3 Capacitated Arc Routing Problem with Intermediate Facilities

In the CARPIF, the intermediate facilities can be a reload station of salt or grit in the context of winter road maintenance, or a landfill site in the context of waste collection. Vehicles can replenish or empty loads along their route at the home depot or the intermediate facilities. Hence, the route service distance can be extended.

Ghiani et al. (2001) proposed two lower bounds for the CARPIF. The first is based on the RPP, and the second is based on the linear relaxation of an integer programming formulation of the CARPIF. They also used an example to show that CARP optimal is not the proper lower bound for the CARPIF. Ghiani et al. (2001) solved the problem by a Tabu search procedure for the single vehicle CARPIF.

Later, Ghiani et al. (2004) investigated the CARPIF and tour-length restrictions (CLARPIF), where multiple vehicles could be used, and the length of any route cannot exceed a predetermined value. They developed a constructive heuristic based on splitting the RPP tour and two Tabu search algorithms.

Polacek et al. (2008) developed a variable neighborhood search (VNS) approach for the CARPIF and CLARPIF. The k-neighborhood exchange is defined as exchanging k consecutive edges between two solutions by a cross-exchange operator. Local search is employed after the neighborhood exchange. The k grows if no better solution is found after the local search, and the VNS stops at a predetermined maximum value of k.

Ghiani et al. (2010) developed a sophisticated ACO algorithm for the CLARPIF. A test conducted by the authors showed that ACO outperforms the algorithms developed in Ghiani et al. (2004) on the GDB instances. However, in the VAL instances, VNS (Polacek et al., 2008) still obtains better results.

Willemse et al. (2016a) introduced the mixed graph capacity ARP under time restrictions with intermediate facilities (MCARPTIF), which considers the mixed road network, and each route cannot exceed a time duration. To solve the problem in near real-time, they investigated four constructive heuristics for MCARPTIF and compared their performance.

Willemse et al. (2016b) also developed optimal and quick, nearly optimal splitting procedures for the MCARPTIF. The split procedures are extensions of the split procedure developed by Lacomme et al. (2004). Willemse et al. (2016b) showed that the optimal split of a giant tour for the MCARPTIF needs a time complexity of $\mathcal{O}(n^3)$, whereas the nearly optimal split can generate solutions at similar solution quality, with a time complexity of $\mathcal{O}(n^2)$.

In Willemse (2016c), the author implemented the previous algorithms and developed a neutral accelerated Tabu search approach to test the performance of their heuristics under different execution time limits. The result shows that, under long execution time limits, their algorithm performs well on large instances, but struggles in small instances.

2.4 Multi-Depot Capacitated Arc Routing Problem

In the MDCARP vehicles can start their route at different depots, and it is usually assumed that vehicles must return to their home depot.

One approach to deal with MDCARP is solving the problem in sequence, which involves splitting the multi-depot covered region into districts in the first phase, with one depot per district, and routing vehicles in the second phase. In addition to sequential solving, iterations between the two phases could be implemented to satisfy restrictions such as route length, load balance, and fleet size. Another approach focuses on routing directly, and the depot responsibility of arcs is subject to change while the computation proceeds.

Mourão et al. (2009) discussed the sectoring ARP. They developed three heuristic approaches. The first two approaches incorporate iterations between the sectoring phase and the routing phase. The variance of the first two approaches is the sectoring partition strategy, using either small circuits or roadway arcs. Then, routes are designed by extension of the

constructive heuristic, including augment-merge, path-scanning, and route-first, cluster-second. If the cost of any route exceeds a predefined value, the number of sectors is added, and the next iteration begins. The third approach builds the sectors and routes simultaneously by an arc insertion heuristic.

Kansou and Yassine (2009) developed the integer programming model for the MDCARP. A hybrid ant colony system with an insertion heuristic is applied to solve the problem. The insertion heuristic inserts arcs to the nearest giant tour of different depots, where ACO optimizes the order of the insertion of arcs. The giant tour split procedure and local searches are explored to improve the solution. Kansou and Yassine (2010) further implemented the MA for the MDCARP.

Xing et al. (2010) proposed an evolutionary algorithm for the MDCARP. They developed a procedure to guide the exploitation of attributes found in nearly optimal solutions. Performance information is applied to select promising operators for selection, crossover, and mutation. Also, priority information is evaluated to guide the insertion positions for crossover and mutation. Hu et al. (2013) solved MDCARP by the MAILS that is similar to Liu et al. (2013).

Krushinsky and Woensel (2015) presented a mixed integer programming model for the mixed graph MDCARP. They developed a symmetry breaking technique and applied the branch-and-cut procedure to solve the problem.

2.5 Application of Routing Optimization for Winter Road Maintenance

This section focuses on the application of arc routing models to problems for winter road maintenance. Winter road maintenance is often formulated as the CARP with problem-specific objectives and constraints. Winter road maintenance involves decision-making problems at all four levels. Perrier et al. (2006a, 2006b, 2007a, 2007b) conducted a four-part

survey of models and algorithms including vehicle routing, depot location, fleet sizing, and system design for spreading, plowing, and disposal operations. Review in this section mainly focuses on the literature regarding winter road maintenance optimization in the recent decade.

2.5.1 Facilities Allocation and Sector Designs Problems

Muyldermans et al. (2002) introduced the problem of districting for salt spreading operations. Road segments first consist of small cycles, and then these cycles are assigned to depots heuristically. Muyldermans et al. (2003) expanded their previous work. The district is designed by greedily assigning edges or cycles, or through an integer programming model that minimizes a lower bound of fleet size.

Perrier et al. (2008b) discussed the sector design and assignment problem for snow disposal operations. Snow disposal is required when large volumes of snow are accumulated on roadways, and there is not enough space along roads for snow storage. In this situation, snow is loaded on trucks and transported to the disposal site. The problem involves deciding the road segments of each sector and the disposal site of each sector. Perrier et al. (2008b) formulated the problem in an integer programming model and solved by two approaches. The first approach assigns road segments to disposal sites in the first phase, then partitions the streets to form sectors. The second approach partitions the streets first, then assigns sectors to disposal sites. All problems are solved by an exact solution solver.

Cai et al. (2009) proposed a location routing problem-based approach to constructing a combined depot location and spreader routing plans. A two-stage Tabu search algorithm is used to solve the problem. Jang et al. (2010) proposed a formulation and a heuristic algorithm for combined depot location, sector design, spreading and plowing route design, fleet configuration, and vehicle scheduling problem. The heuristic approach solves these sub-

problems iteratively until no better solution can be found. A mixed fleet is assumed. Also, road class service frequency and spreader capacity constraints are considered. Service routes are established exclusively for each class, but a vehicle can service multiple routes in different classes with different frequencies.

2.5.2 Routing Problems

Formulated as a classical CARP model, Omer (2007) solved the CARP for winter maintenance using GRASP. Each iteration consists of two parts: construction and local search. Liu et al. (2014) used a mathematical optimization model based on the CARP to minimize the total travel distance for snowplow operations. The problem is solved by the MA with extended neighborhood search. Sensitivity analyses are conducted to compare different depot locations and route costs for each depot, with the predetermined candidate depot locations and the number of routes.

Variations and special cases of CARP have been proposed by literature. Haghani and Qiao (2001) developed a linear, mixed integer program for routing salt spreader trucks for Calvert County, Maryland. They considered time windows, capacity, and route duration constraints for the network hierarchy. A combination of several heuristic procedures is proposed to solve the CARP for snow removal.

Later, Haghani and Qiao (2002) addressed the service-continuity issue as a constraint, which means a route can only have connected service links with possible deadhead from the depot to the service beginning node and deadhead from the service end node to the depot. They first transformed the traffic network into a link to link connection adjacent matrix; then, used the capacitated minimum spanning tree (CMST) formulation to solve the problem. The CMST is a nonlinear problem. A linear function is used for approximation.

Tagmouti et al. (2007) considered the cost of treating a road depending on the time of service. The application of salt on the road should be neither too early nor too late, which is formulated as a time-dependent piecewise linear service cost function. The objective function minimizes the total deadhead and time-dependent service costs. The problem is solved by a column-generation approach. Later, the same authors applied the variable neighborhood descent (VND) algorithm to solve the problem (Tagmouti et al., 2010).

Hierarchy constraints need to be considered when there is an ordering relation between roadway classes. In this case, a higher hierarchy road must be serviced first, followed by lower hierarchy roads, and so on. Perrier et al. (2008a) proposed a model to incorporate hierarchical objectives, different service and deadhead speed, road-vehicle dependency, load balance, and turn restrictions. A parallel constructing heuristic and a cluster-first, route-second algorithm are developed to solve the snowplow routing problem.

Synchronized arc routing was introduced by Salazar-Aguilar et al. (2012). In this problem, a multi-lane street in the same direction must be plowed simultaneously by different synchronized vehicles. The authors developed the model and applied adaptive large neighborhood search to solve the problem.

When plowing and spreading simultaneously, the road segment must be served by the first truck traversal. That is, deadhead is not allowed before the road is plowed. This idea was first formulated by Lystlund and Wohlk (2012) on the undirected network. In their paper, a constraint called service-time restricted is introduced, restricting the service finishing time of edges to be earlier than the deadheading time of that edge. A constructive heuristic is employed to solve the problem.

Dussault et al. (2013) addressed the problem that plowing uphill takes a much longer time than plowing downhill. The problem is formulated as a variant of the windy postman problem. The authors developed a local search algorithm and tested on theoretical instances.

Hajibabai et al. (2014) developed the model for plowing priorities and resource replenishment to minimize total operation time. Turning delay and U-turn allowance is taken into consideration during network building. The formulation is in the form of a VRP. The solution is generated by a construction heuristic and then improved by a local search algorithm. In further work (Hajibabai et al., 2016), the stochastic version of this problem considered uncertain service demand and service disruptions.

Kinable and van Hoeve (2016) proposed a model for snow plowing operations that considers heterogeneous capacity, fuel and salt limits, and intermediate facilities. An integer programming model, a constraint programming model, and a two-phase heuristic procedure are compared for solving the problem. The result shows that the heuristic procedure has the best performance regarding both computation time and quality.

Quirion-Blais et al. (2017) focused on the problem that streets need to be plowed and spread in a sequence depending on road hierarchy. Turning restriction, various speed by truck type, road class hierarchy and operations, and road-vehicle dependency are also taken into consideration in this paper. Adaptive large neighborhood search is devised to solve the hierarchy ARP.

Gundersen et al. (2017) developed a mixed integer model for snowplow operations with road hierarchy constraints and road-vehicle dependency. The difference between Perrier et al. (2008a) and Gundersen et al. (2017) is that the priority in the former problem is between road classes, whereas precedence relation in the later problem is between pairs of

arcs. U-turn penalty was also studied in Gundersen et al. (2017). The model is implemented and solved by an exact solution solver.

2.5.3 Real-time Routing Problems

Real-time information such as pavement temperature and weather forecast are taken into consideration for dynamic routing. Handa et al. (2005) incorporated a next-generation road weather system into their ARP. The weather system provides road temperature forecast and then yields service demanding roads based on temperature as inputs for CARP. The problem is solved by the MA.

Tagmouti et al. (2011) considered the dynamic weather input based on their previous model (Tagmouti et al., 2010). After the weather updates are received, the relationship between the cost and time of treatment of any road may alter.

Xu et al. (2017) proposed a model for the benefit of snow plowing depending on real-time snow accumulation and the impact to travel time. A cluster-first, route-second heuristic is applied to solve the problem.

Table 2.1 summarizes the literature in section 2.5. The problem type is classified in the “Base Problem” column, indicating the main characteristics of the problems. Almost all papers regarding winter road maintenance routing are problem-specific, not to mention the additional constraints that are taken into consideration. Also, all papers have tailored existing algorithms more or less to solve their problem.

About one-half of these papers employed problem-specific constructive heuristics. Two reasons might lead the authors to choose constructive heuristics. First, the computation time for constructive heuristics is much less compared to metaheuristics. Second, creating a nearly optimal feasible solution is already difficult enough for the problem. However, it is

difficult to evaluate the algorithms, since the instance used is problem-specific, and the heuristic algorithm, by its nature, does not guarantee to provide the optimal global solution.

2.6 Research Contributions

The main contributions of this dissertation are as follows. First, this study developed a mathematical model for the CARP with constraints of road segment service cycle time, heterogeneous vehicle capacity, fleet size, and road-vehicle dependency. This problem is referred to as a single depot winter road maintenance routing problem (SDWRMRP). This is the first study that developed the model for SDWRMRP.

Second, this is the first study that uses the MA approach to solve the SDWRMRP and the first study that developed the route split procedure for SDWRMRP. The previous study only uses a constructive algorithm to solve SDWRMRP without the road-vehicle dependency constraint. Specifically, Jang et al. (2010) built a model that considered constraints of road segment service cycle time, heterogeneous vehicle capacity and fleet size, and solved the problem by the iterative constructive heuristic. A previous study that uses MA to solve the winter road maintenance routing problem only considered homogenous vehicle capacity and unique route cycle time (Liu et al. 2014).

Third, this study developed a mathematical model for the CARP with constraints of road segment service cycle time, heterogeneous vehicle capacity, fleet size, road-vehicle dependency, and work duration. This problem is referred to as a multi-depot winter road maintenance routing problem with intermediate facilities (MDWRMRPIF). This is the first study that developed the model for MDWRMRPIF.

Table 2.1 *Literature of ARP regarding winter road maintenance*

Level	Paper	Base Problem	Additional constraints	Algorithms
Strategic and tactical	Muyldermans et al. (2002, 2003)	sector design		constructive heuristics
	Perrier et al. (2008b)	sector design, snow disposal site assignment	sector contiguity and balance, hourly and annual disposal site capacities, single assignment requirements	two-stage integer programming
	Cai et al. (2009)	depot location, sector design, route design		two-stage Tabu search
	Jang et al. (2010)	depot location, sector design, route design, vehicle scheduling	heterogeneous capacity, fleet size, road service frequency	iterative multi-phase constructive heuristics
Operational	Haghani and Qiao (2001)	CARP	service start time, route duration, time windows, service one or two lanes in a single pass	constructive heuristic with local search
	Haghani and Qiao (2002)	Service-Continuity CARP	both-side service	CMST
	Omer (2007)	CARP		GRASP
	Tagmouti et al. (2007)	CARP with time-dependent cost		column generation
	Perrier et al. (2008a)	Hierarchy ARP	different service and deadhead speed, class upgrading, road-vehicle dependency, load balance, and turn restrictions	1) parallel constructive heuristic and 2) cluster-first, route-second heuristic
	Tagmouti et al. (2010)	CARP with time-dependent cost		VND
	Salazar-Aguilar et al. (2012)	Synchronized ARP		Adaptive Large Neighborhood Search
	Lystlund and Wohlk (2012)	Service-rime restricted CARP		constructive heuristic

Table 2.1. (continued)

Level	Paper	Base Problem	Additional constraints	Algorithms
Operational	Dussault et al. (2013)	Windy CARP		1)linear programming relaxation, and 2) constructive heuristic with local search
	Hajibabai et al. (2014)	CARPTIF with route duration cost	turn delay	constructive heuristic with local search
	Liu et al. (2014)	CARP	route duration	MA with Extended Neighborhood Search
	Hajibabai et al. (2016)	Stochastic CARPTIF with route duration cost		Concave Adaptive Value Estimation
	Kinable and van Hoeve (2016)	CARPTIF	heterogeneous capacity, fuel, and salt limits	1) Integer Programming, 2) constraint programming model, and 3) two-phase heuristic procedure
	Quirion-Blais et al. (2017)	Hierarchy CARP	turning restriction, various speed by truck type, road class operations, and road-vehicle dependency	Adaptive Large Neighborhood Search
	Gundersen et al. (2017)	Hierarchy ARP	hierarchy is between road segment	Integer Programming
Real-time	Handa et al. (2005)	CARP		MA
	Tagmouti et al. (2011)	CARP with time-dependent cost		VND
	Xu et al. (2017)	CARP		cluster-first, route-second heuristic

Fourth, this is the first study that uses the MA approach to solve MDWRMRPIF and the first study that developed the route split procedure for MDWRMRPIF. Previous researchers only solved partial problems, that is, the CARPIF and the MDCARP using the MA approach. Willemse et al. (2016b) developed optimal and quick nearly optimal splitting procedures for the mix graph CARPTIF, but the algorithm only considers single depot. Several studies, Kansou, and Yassine (2010), Xing et al. (2010), and Hu et al. (2013) uses the MA approach to solve the MDCARP, but intermediate facilities are not considered in those algorithms.

In this study, model parameters are derived by the sample from historical snowplow operational data. The proposed algorithms are implemented to solve real-world problems.

CHAPTER 3. DATA DESCRIPTION

3.1 Service Maps

The Iowa Department of Transportation (DOT) is responsible for servicing over 24,000 lane miles for the whole state. The roadways mainly consist of interstates, state highways, and Iowa roads. The road network is split into six districts. In this study, the analysis is conducted for District 3, which is located in Northwest Iowa, containing 20 depots and about 4,000 lane miles. Figure 3.1 shows the service region, with each depots' responsible region color-coded and labeled. The black triangles represent the depot location.

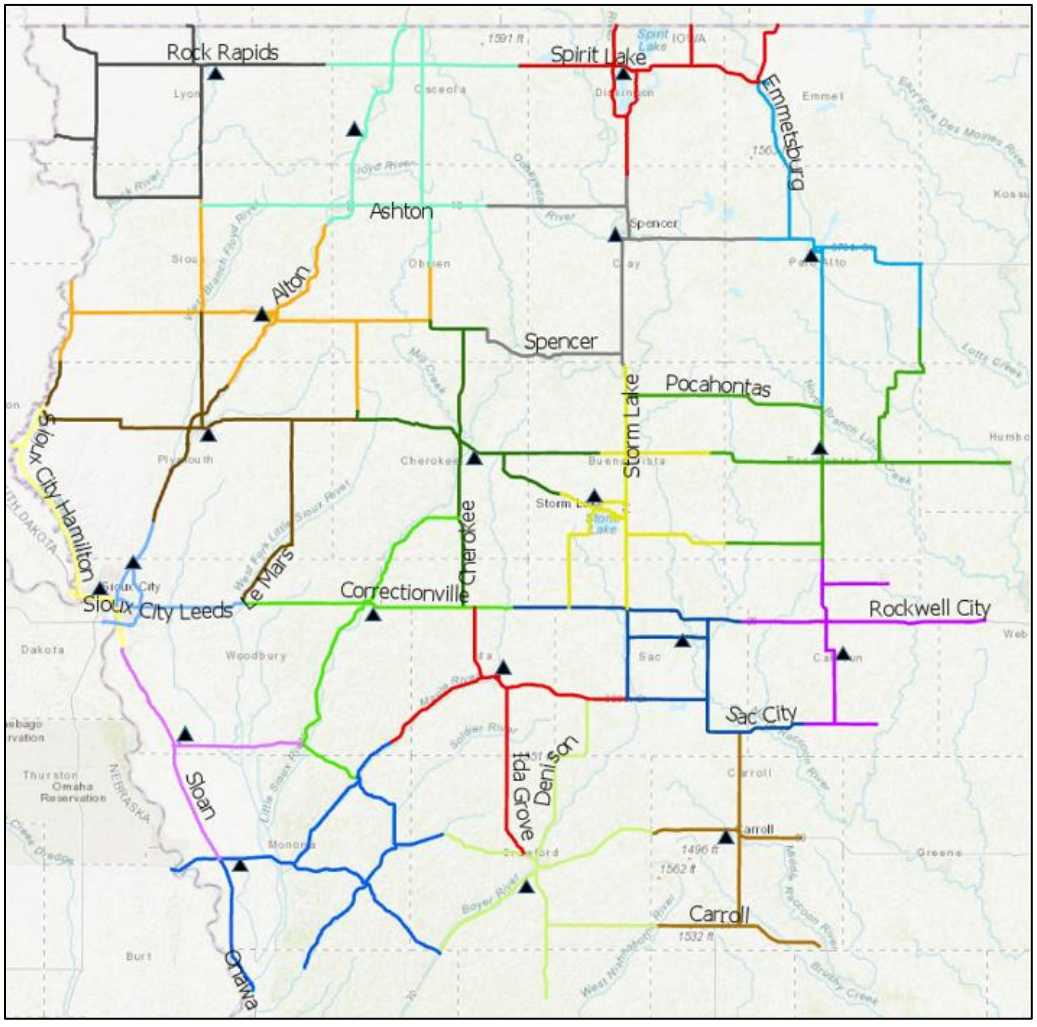

Figure 3.1 *The service region of District 3, Iowa. 20 depots are color-coded and labeled*

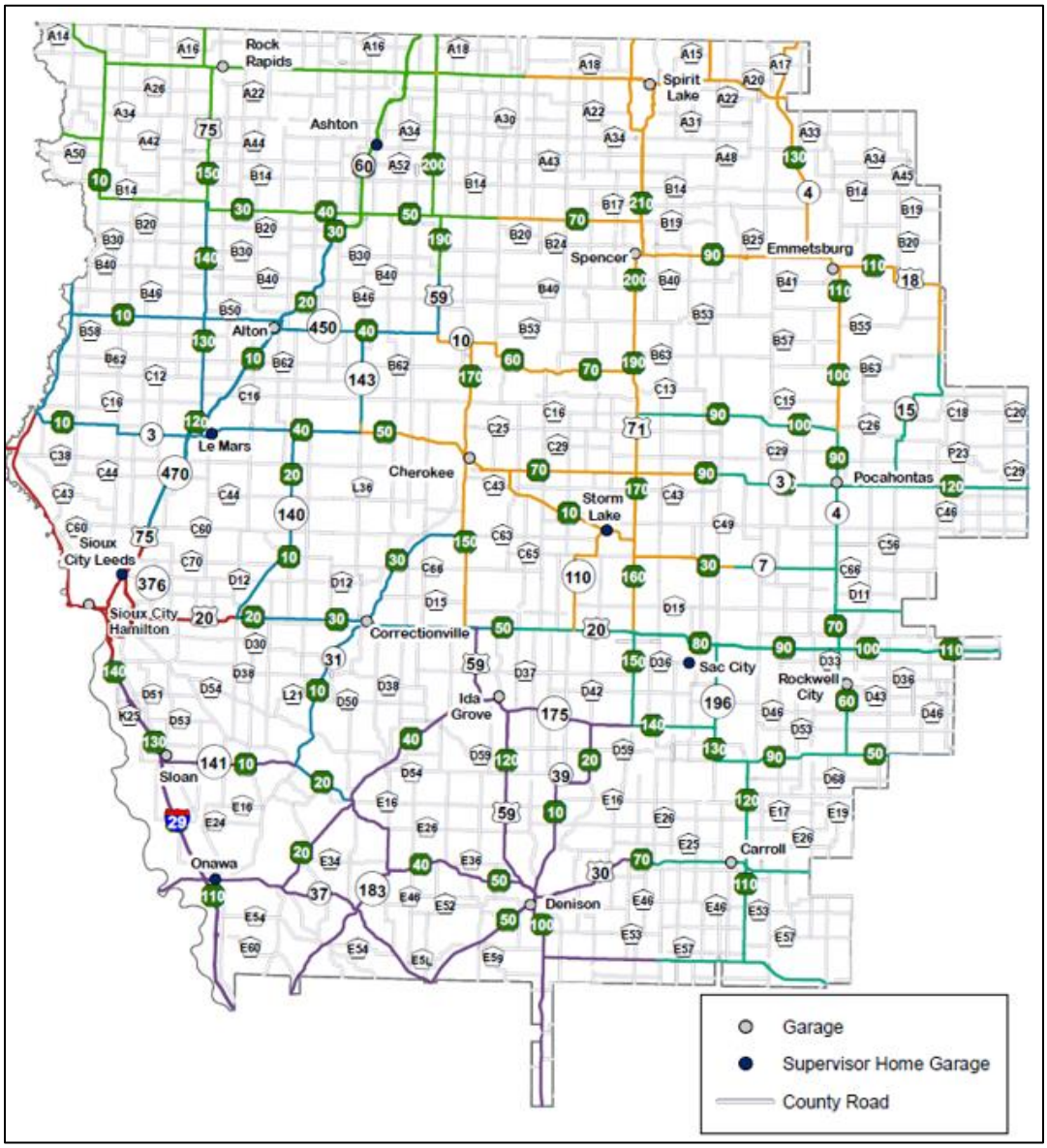

Figure 3.2 *Sector of District 3*

There are six sectors in District 3. In Figure 3.2, the six sectors are represented by different colors. Each depot has an area of service responsibility map. Figure 3.3 shows the service responsibility map of the Storm Lake depot as an example. All service required road segments are drawn by markers. The depot is illustrated by a gray triangle.

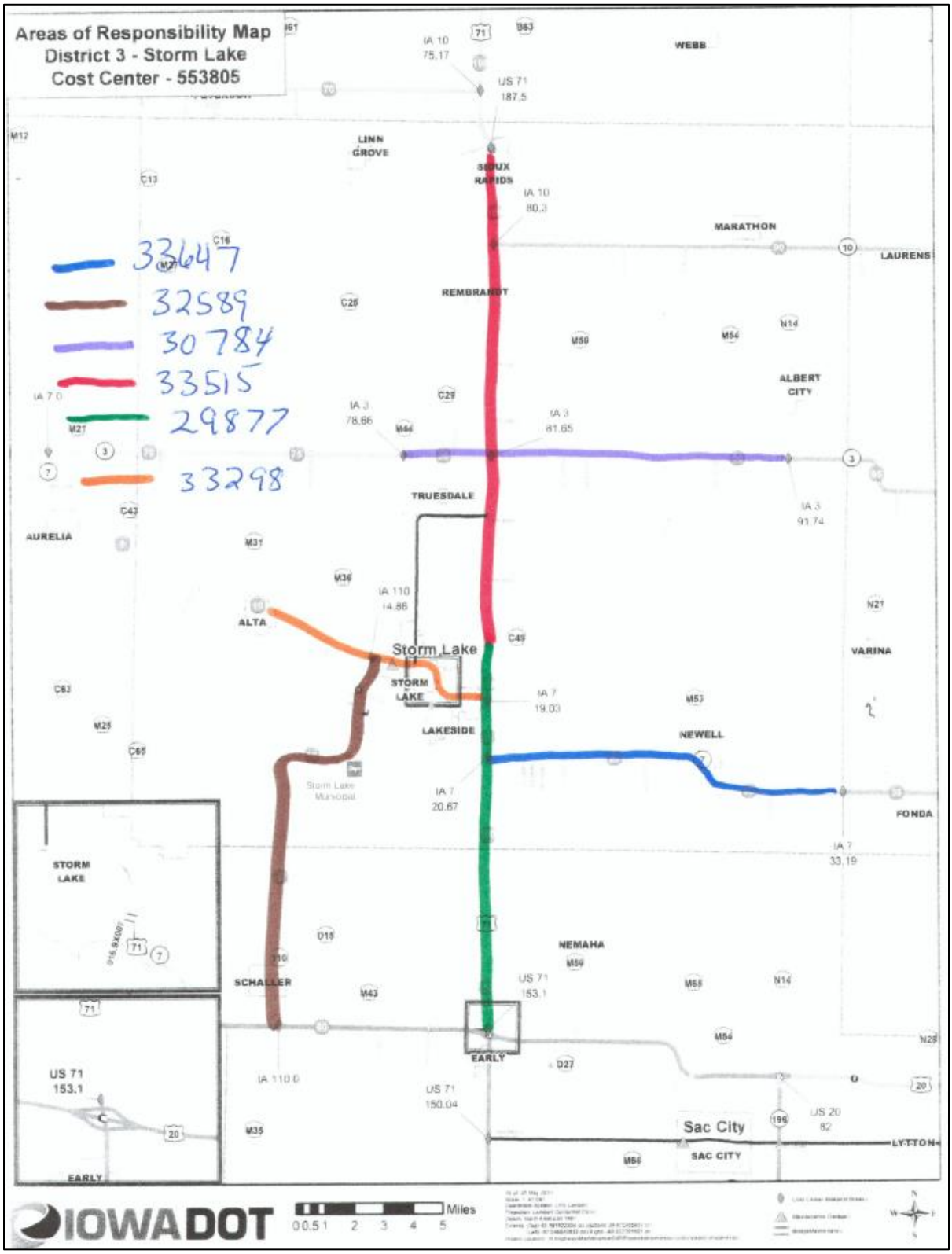

Figure 3.3 *Area of responsibility map of Storm Lake, color-coded for current routes*

From the responsibility map, boundaries for the depot can be determined by mile markers. For example, the “IA 3, 78.66” in the middle of the map indicates that the west boundary of service Iowa road 3 ends at a mileage of 78.66.

The areas of responsibility map also illustrate the designed route of the current operation. The current route has been designed by experience. In Figure 3.3, there are six routes to service the Storm Lake network with each of them assigned a unique color. The numbers next to the color legend are the truck IDs.

3.2 Snow Plow Operation Data

The automatic vehicle location (AVL) system is designed to determine and transmit the geographic location of a vehicle. Each maintenance truck is equipped with an AVL system, and the following information is collected: the GPS location, plow position, spreading rate and type, truck speed and directions, truck ID, and timestamp. Iowa DOT collects the AVL data at a high resolution (< 30-second intervals). Figure 3.4 shows an example of an urban route in Sioux City.

3.2.1 Test Run

On December 19, 2017, all depots in District 3 carried out a test run in the morning. All trucks traveled along their current service routes and spread sand on the road. From this test run, the actual truck routes can be found. The traveling distance can be calculated. Also, the actual turnaround locations can be observed by carefully scanning the AVL data.

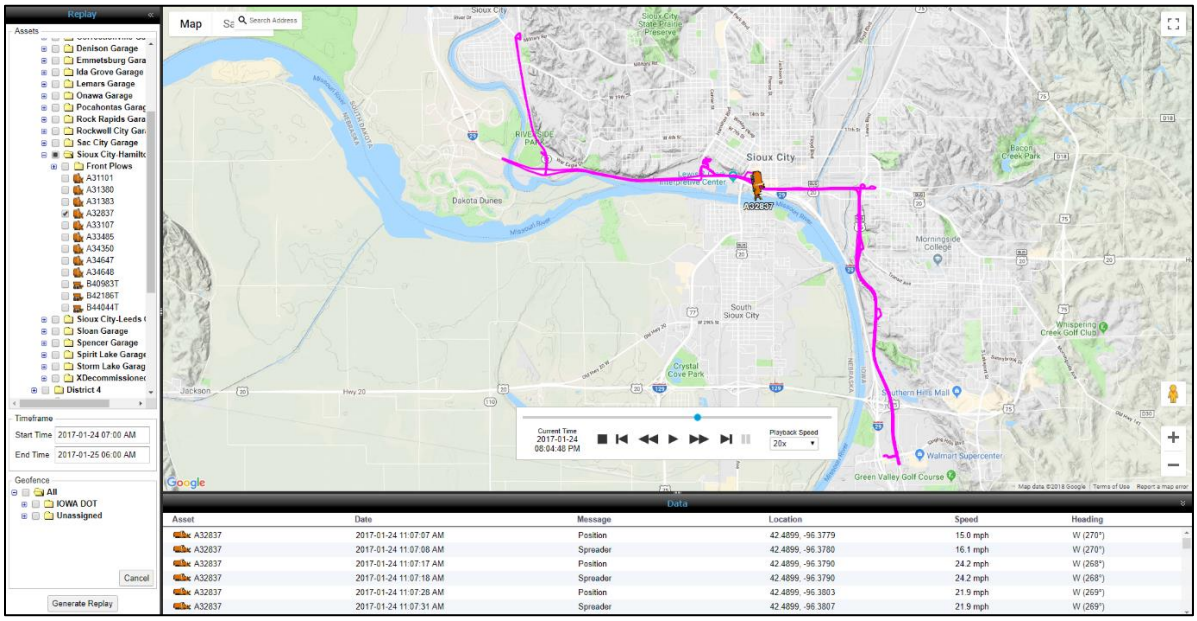

Figure 3.4 *An example of truck tracking showing one route in Sioux City*

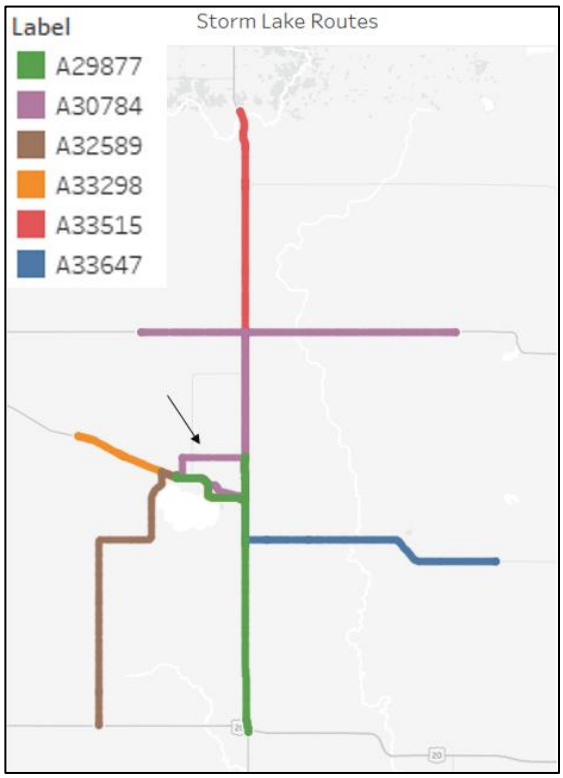

Figure 3.5 *AVL route map of Storm Lake depot*

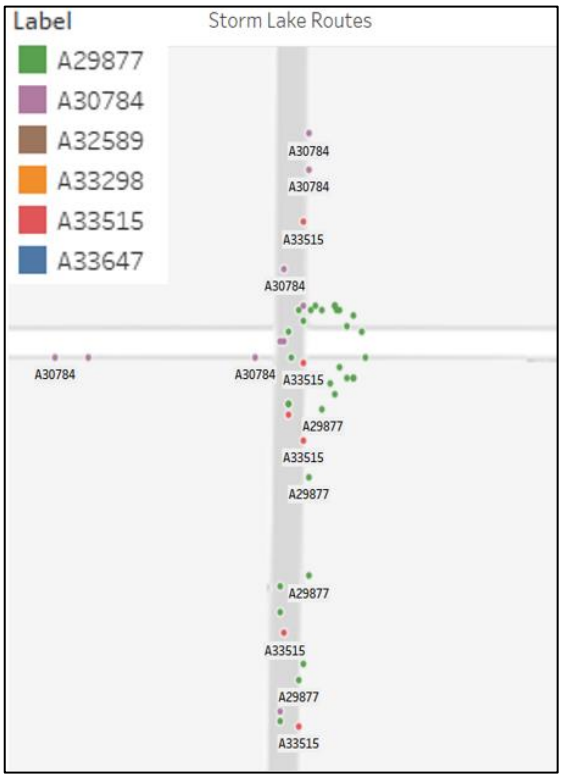

Figure 3.6 *Example of U-turn location shown by AVL route map*

The test run AVL data of Storm Lake depot is shown in Figure 3.5 as an example. All routes are drawn using the same color as in the Storm Lake responsibility map. Each record of the AVL data corresponds to a geographic location (point) on the map, but the map is shown at a scale so that the points aggregate and look like routes. Partial routes are overlapped, and only the route on top can be observed. It can be recognized that the actual AVL routes are almost the same as in the responsibility map. However, AVL routes could use road segments that do not have service demand. For example, compared to Figure 3.3, in the middle of the AVL route map, the purple route "A30784" traverses on some non-service road segments (pointed to by the black arrow).

Maintenance trucks need a larger space to make U-turns than passenger cars. Undivided two-lane roadways do not have enough space for U-turns. The U-turn location generally has additional space such as turning lanes or truck parking lots along the route. Thus, the test run AVL data were explored for U-turn locations that are used by the truck operators.

Figure 3.6 shows an example of the turnaround location, where the truck "A29877" turned around at an intersection. From the AVL data, it is observed that U-turn locations have common characteristics. For example, when making U-turns, multiple points gather in a small area. The truck speed (not shown in Figure 3.6) drops to nearly 0 mph. More illustrative animations that track the AVL routes are employed to facilitate the observations.

3.2.2 Storm Fighting Under Different Weather Condition

3.2.2.1 Weather Data

Weather data from the National Weather Service Cooperative Observer Program (NWS COOP), Automated Surface Observing System (ASOS) and the Roadway Weather Information System (RWIS) are obtained from October 1, 2016, to April 1, 2017, and

October 1, 2017, to April 1, 2018. So, two winter seasons were inspected for storms. The NWS COOP contains daily snowfall and daily snow depth. The ASOS contains precipitation types and rates. The RWIS contains information of roadway surface conditions, such as "dry," "trace moisture," "wet," "chemically wet," "frost," "ice watch," and "ice warning." ASOS is recorded at 5-minute intervals, and RWIS is recorded at 10-minute intervals. Figure 3.7, Figure 3.8, and Figure 3.9 indicate the NWS COOP, ASOS, and RWIS stations in Iowa State, separately. Red dots are the stations, and those in the black circle are within District 3.

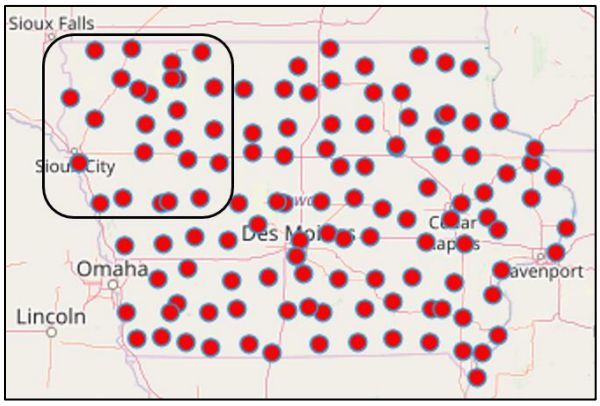

Figure 3.7 *NWS COOP stations*

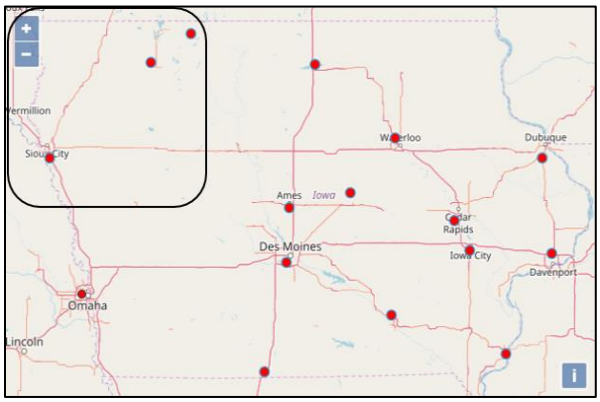

Figure 3.8 *ASOS stations*

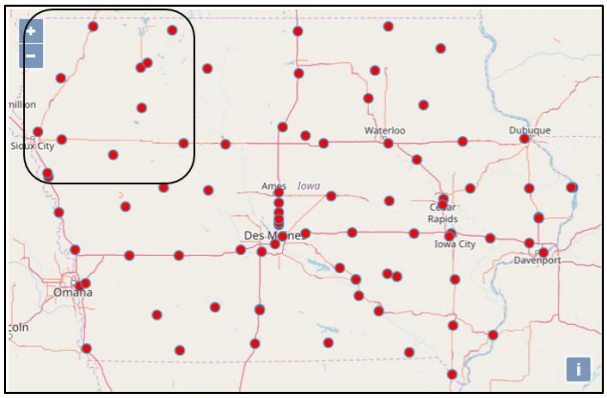

Figure 3.9 *RWIS stations*

3.2.2.2 Speed and Spreading Rate Estimation

The real maintenance data during a severe storm is sampled in this study. Typically, service speed is among 20-35 mph but could be slower if the snow is heavy. Service speed and deadhead speed are examined separately for urban and rural areas. This is because high traffic volume and closely spaced traffic lights and stop signs in urban areas typically reduce truck operation speed.

After careful examination of the weather data and AVL data, January 24 and 25, 2017, January 22 and 23, 2018, and March 6, 2018, are used to retrieve truck operation data. During this period, different road conditions are covered. The storm levels include light, moderate, and severe, and the road level includes interstates, highways, and Iowa roads. Four truck routes were retrieved for each combination of the storm level and the road level. Therefore, 36 routes were retrieved during the study periods. These routes are used to evaluate the speed and spreading rate. All records in these routes were first labeled by urban or rural area for speed evaluation.

Figure 3.10 shows the speed histogram of the urban or rural routes; a filter of speed higher than 0 is applied. The histogram includes service speed and deadhead speed. It can be observed that the histogram only shows one peak for speed over 10 mph. It is difficult to

differentiate the service speed and the deadhead speed only from the histogram. Therefore, the service speed and deadhead speed are estimated by filtering spreading rates.

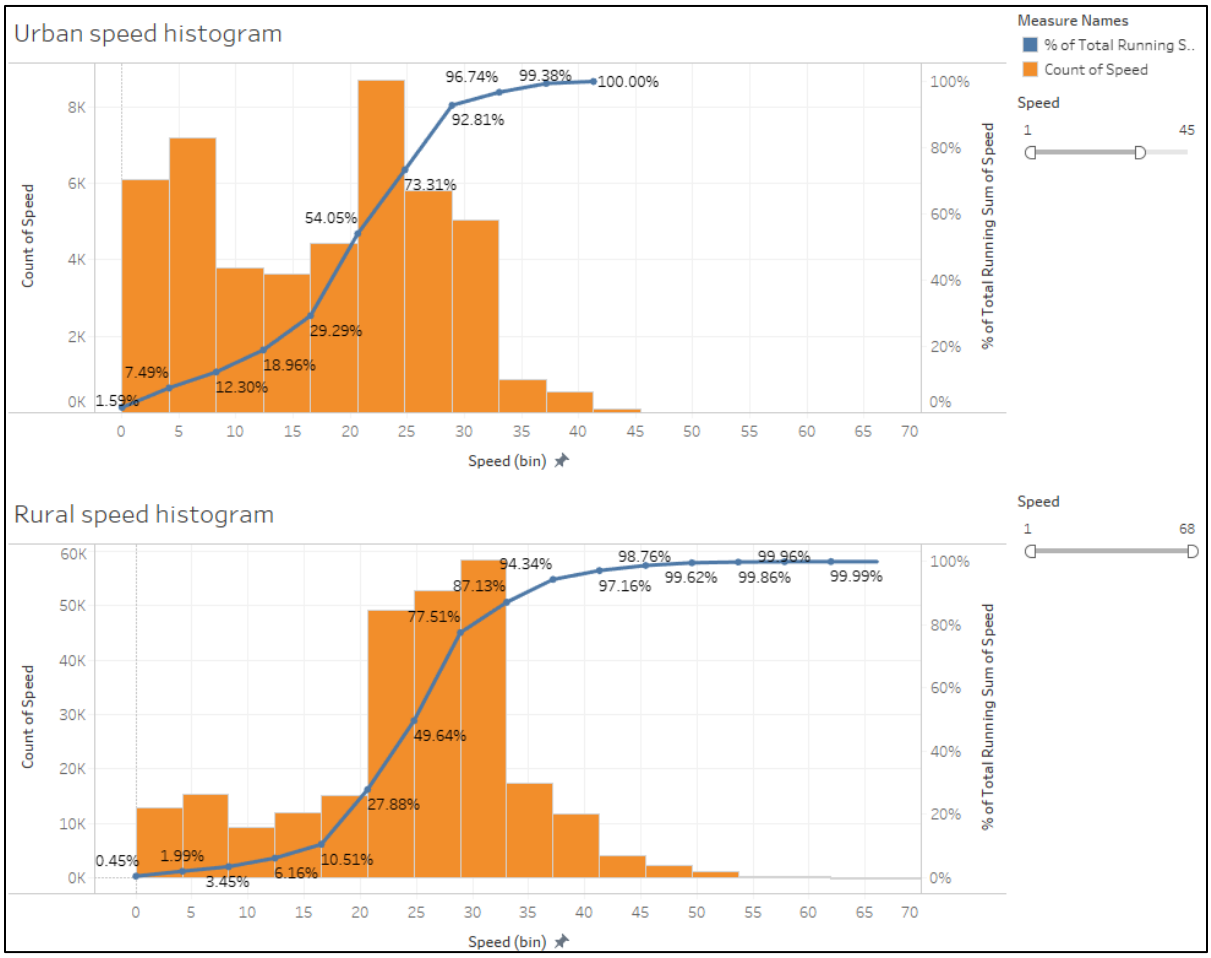

Figure 3.10 *Speed histogram of the city or rural routes*

Figure 3.11 3.11 shows the boxplots of speed versus different weather of the urban or rural routes, separately. It can be seen that weather does not have much influence on speed. Figure 3.12 shows the boxplots of speed versus different road level of the urban or rural routes, separately. It can be seen that road levels only have a large affect on urban interstate roads. The reason could be that drivers would like to use a roadway that serviced more frequently than others, hence more traffic caused the speed decrease.

The 50th percentile is used as the service speed. Hence, urban area service speed is 22 mph, and rural area service speed is 26 mph. The 95th percentile is used as the deadhead speed. The reason is that if the snowplow trucks plow without spreading, it is still considered servicing, so including a filter of set spread rate = 0 lbs./mile will not remove the plowing-only service. The estimated urban area deadhead speed is 32 mph, and rural area deadhead speed is 40 mph.

A filter of set spread rate > 0 lbs./mile, urban speed > 0 mph, and rural speed > 5 mph is applied for all boxplots in Figure 3.11 and Figure 3.12. The reasons that rural speed estimation applied a filter of speed > 5 mph are stated below. First, rural areas have fewer stop signs and traffic lights. Speed in the city can be estimated as a whole, but the speed in rural areas has more variations. Second, if the road segment has additional turning lanes, trucks often make U-turns in a short distance, traveling back and forth to service this road segment. Therefore, a large amount of data is collected for the U-turn area. Figure 3.13 illustrates U-turn records. Hence, it is proper to set a low speed around the intersections and the U-turn locations and evaluate the operation speed of other places separately. By filtering speed > 5 mph, the boxplots are more accurate.

The spreading rate is also estimated from the 36 routes. Figure 3.14 illustrates the boxplots for set spread rate of the road level on the left and the storm level on the right. It can be seen in both boxplots, the maximum set rate is 300 lbs./lane mile, and the median set rate is 150 lbs./lane mile. So, the median set rate is only half of the maximum. The higher the road level, the more the set spread rate. This follows common sense. Since interstates have more traffic volume, they should be serviced more thoroughly. The higher the storm level, the more variate the set spread rate. The reason is below. Within a day, the snow fall

magnitude could changes from time to time. However, the snow fall depth data is only measured about every 24 hours. If the whole day only have light snow during the 24 hours, the spreading rate should be set to deal with light snow. On the other hand, a severe snow depth data could be consist of light, moderate, or severe snow falls during separate hours (or separate snowfalls), as long as the cumulative snowfalls is high enough for that day to be classed as severe snow. Therefore, a severe snow depth's day can still have light snowfalls in some hours, and the treatment spreading rate data for the light snow are also included for that day.

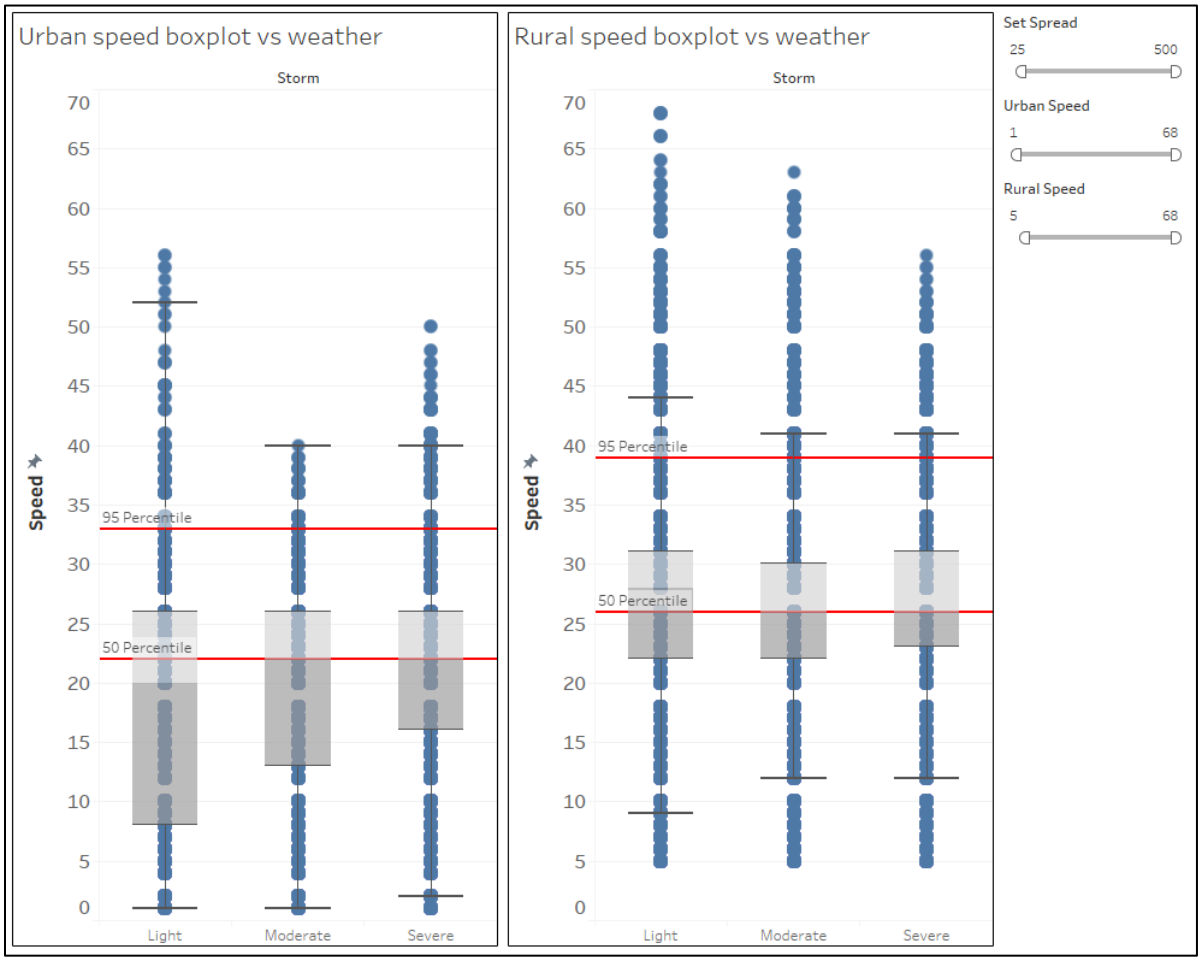

Figure 3.11 *Service speed boxplot of the city or rural routes*

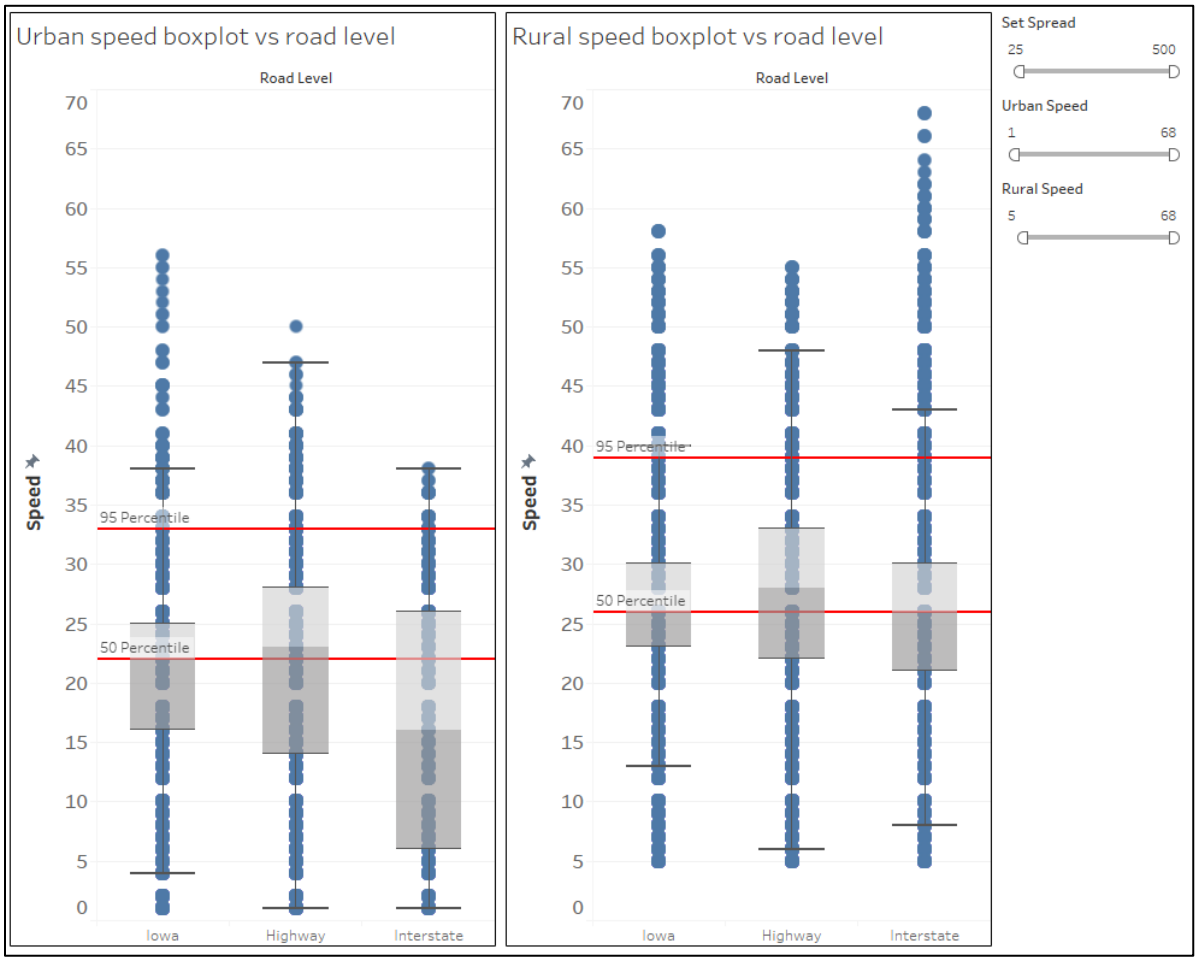

Figure 3.12 *Deadhead speed boxplot of the city or rural routes*

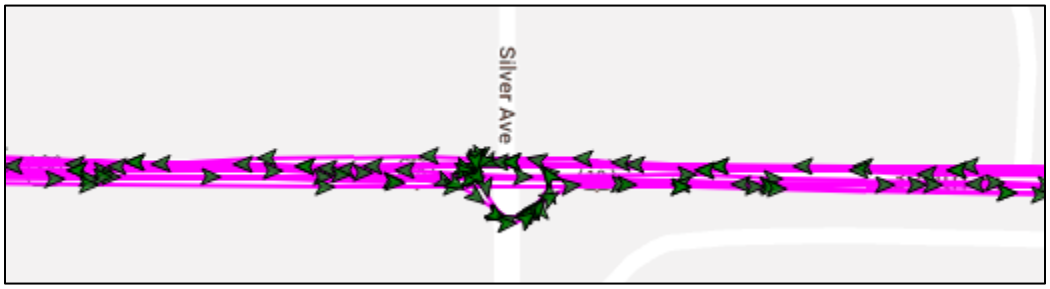

Figure 3.13 *Example of U-turn, each arrow indicates one record*

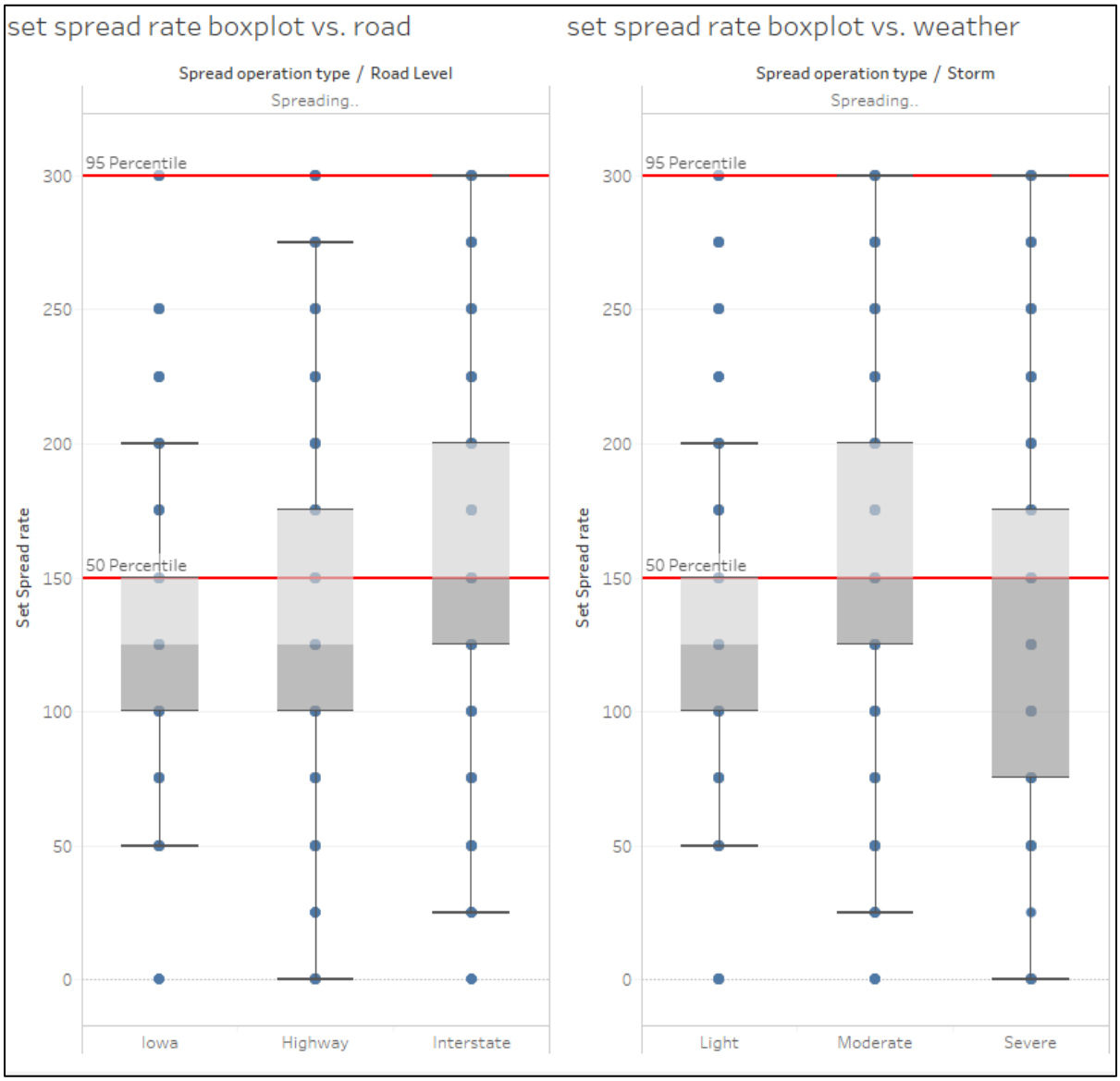

Figure 3.14 *Solid spreading rate of the 36 routes*

3.3 Traffic Network

The Roadway Asset Management System (RAMS) is a database that stores the roadway data for all public roads in Iowa. RAMS manages the roadway data through a Geographic Information System interface. RAMS includes roadway geometrics, maintenance responsibility, traffic information, pavements, and bridges structures. Different types of

roadway attributes are saved in different layers; for example, the number of lanes and the speed limit are saved in two layers separately.

In this study, the network is built as a directed graph since the maintenance truck can service only one lane per run. Non-service road segments and U-turn arcs had been added manually in a way that the traffic network represents accurately where such operations can be made. After merging all relevant data into layers, the network has been inspected thoroughly. Each road segment in the graph has the following attributes: “road segment ID,” “depot,” “facility type,” “number of lanes,” “length,” “maintenance service level,” “service flag,” “deadhead speed” and “service speed.” They are introduced below.

- road segment ID

  A unique index of each road segment.

- depot

  The depot that responds to service this road segment.

- facility type

  This attribute indicates whether the arc represents a divided roadway or an undivided roadway. Together with the “number of lanes,” the road-vehicle dependency (section 4.2.1) can be determined.

- number of lanes

  The directed road segment could have one or multiple lanes. A “wing plow direction” attribute is generated based on the “facility type” and “number of lanes” information. For undivided roadways, the wing plow direction of all lanes is “right.” For divided roadways, the wing plow direction is “left” for only the left-most lane, and the remaining lanes are “right.”

- length

  The road segment length.

- roadway levels

  The roadway levels include “Interstates,” “U.S. highway,” “Iowa road,” and “local road.”

- service flag

  This indicates whether the road segment required service.

- deadhead speed

  This is the deadhead speed determined from section 3.2.2.

- service speed

  This is the service speed determined from section 3.2.2.

Table 3.1 *Network size*

Depot	Node	Arc Req.	Arc Not Req.	Depot	Node	Arc Req.	Arc Not Req.
Alton	253	489	138	Pocahontas	81	96	14
Ashton	166	320	66	Rock Rapids	64	89	11
Carroll	128	211	53	Rockwell City	87	142	35
Cherokee	122	174	43	Sac City	85	106	37
Correctionville	62	93	13	Sioux City-Hamilton	208	418	141
Denison	112	151	34	Sioux City-Leeds	253	525	119
Emmetsburg	82	101	30	Sloan	49	78	12
Ida Grove	53	57	17	Spencer	65	91	18
Le Mars	134	219	44	Spirit Lake	83	134	23
Onawa	117	138	33	Storm Lake	97	116	41

Finally, the network shapefile is exported as an edge list for each depot. If a road segment has k ($k > 1$) number of lanes, $k - 1$ arcs will be duplicated. Each arc is assigned a unique “arc ID.” Thus, the roadway network is represented as a multi-lane directed graph.

Table 3.1 demonstrates the network sizes, including the number of nodes, the number of arcs that require service, and the number of arcs that do not need service (non-service).

3.4 Estimate Current Operations Efficiency

The test run AVL data is used to estimate the travel distance under current operations for each depot. First, the GPS points were snapped to the nearest road segment. Then, all AVL routes were checked for missing data. The missing data were interpolated by adding records along the roadway that mimic the operation path. After inspecting the AVL test run data, three routes were found duplicated, and six routes were found unnecessary (no spreading rate), and these routes were excluded. Finally, the travel distance was calculated for each route using the GPS information and then summed up by depot. The AVL travel distance is displayed in the last column in Table 3.2.

In Table 3.2, the service lane miles represent the demand distance. The second column shows the RAMS service lane miles. The third column represents the travel distance of the test run routes. Some routes traversed on some of their road segments several times, more than what is needed to service and deadhead on the road segments. Travel distance with this problem is labeled with an asterisk. For those garages without duplicated runs, the difference between the travel distance and the service lane mile is the deadhead distance, which accounts for 17.7% to 58.4% of the total travel distance.

Table 3.2 *Service lane miles and travel distance for each depot*

Depot	Service Lane Miles	Travel Distance Miles
Alton	293.9	419.3
Ashton	307.4	738.0*
Carroll	151.5	255.4
Cherokee	180.3	219.1
Correctionville	229.8	227.0*
Denison	218.0	320.1*
Emmetsburg	150.2	225.0*
Ida Grove	128.6	177.0*
Le Mars	264.0	375.4
Onawa	298.3	504.1
Pocahontas	229.6	298.5*
Rock Rapids	185.4	443.7*
Rockwell City	199.4	259.7
Sac City	211.1	434.1*
Sioux City-Hamilton	241.1	504.7*
Sioux City-Leeds	221.0	376.0*
Sloan	126.6	163.7
Spencer	171.5	297.3*
Spirit Lake	176.7	424.9
Storm Lake	163.8	243.1

* Test run duplicates on multiple road segments

CHAPTER 4. PRACTICAL CONSTRAINTS

This chapter introduces four practical constraints that are considered in formulating and solving the CARP in this study. Three of them are related to snowplow trucks, the last one is related to road segment service cycle time.

4.1 Heterogeneous Capacity

The winter road maintenance trucks are capable of plowing snow and spreading materials simultaneously or separately. The materials that are spread could be sand (to increase roadway friction), salt, or a mixture of both. Figure 4.1 shows a typical winter road maintenance truck with a 12' front plow, a 10' right-wing plow, an 11' underbody scraper, a dump body, and pre-wet tanks (Snow Plow Truck, NDDOT). The dump body stores dry materials (i.e., salt, sand, or a mixture of the two). Dry materials are discharged by a spinner. The pre-wet tanks store salt brine that is sprayed onto the salt/sand when it leaves the sander body. This could facilitate the melting process and help the dry material stick to the road surface rather than be blown off the road.

Figure 4.1 *Winter road maintenance truck (Picture from Snow Plow Truck, NDDOT)*

There are two types of trucks used by the Iowa DOT: heavy duty and medium duty (Iowa DOT Office of Maintenance, 2018). The medium-duty trucks have a solid material capacity of 12,000 lbs. The heavy-duty trucks have a solid material capacity of 16,000 lbs.

4.2 Fleet Size

The truck inventory of each depot in District 3 is shown in Table 4.1. Each depot has a limited number of snowplow trucks of each type. These two types of trucks can be used interchangeably on any routes if the demand for solid material is less than the truck capacity.

Table 4.1 *Truck inventory of District 3 depots*

Depot name	**Medium duty (single-axle trucks)**	**Heavy-duty (tandem-axle trucks)**	**Total**
Alton	4	5	9
Ashton	7	8	15
Carroll	3	2	5
Cherokee	5	2	7
Correctionville	3	3	6
Denison	5	3	8
Emmetsburg	2	3	5
Ida Grove	2	2	4
Le Mars	2	7	9
Onawa	3	6	9
Pocahontas	3	4	7
Rock Rapids	2	3	5
Rockwell City	2	5	7
Sac City	2	4	6
Sioux City-Hamilton	6	3	9
Sioux City-Leeds	3	4	7
Sloan	2	4	6
Spencer	1	5	6
Spirit Lake	5	4	9
Storm Lake	3	4	7

4.3 Road-Truck Dependency

Although the front plow is 12 feet in width, which is the typical lane width of U.S. Interstate roadways, it should be tilted at an angle in usage so that snow is pushed off-road. Thus, the front plow clears 10 feet of the roadway. The underbody scraper removes compacted snow and ice and can discharge it onto the wing plow. The wing plow can help clear a wider path when used in conjunction with the front plow or underbody scraper, allowing the operator to push snow along the road shoulder. A 10-foot wing plow clears 8 feet of roadway.

In Iowa, all trucks are equipped with a front plow. Most trucks are equipped with either a left-wing plow or a right-wing plow. Some of the trucks are equipped with an underbody scraper.

Road-truck dependency arises when snow and ice must be pushed at a predefined direction on some roadways, and most trucks only have one directional wing plow. Figure 4.2 and Figure 4.3 illustrate this relationship. On the undivided two-lane or multi-lane streets, snow and ice must be pushed to the right shoulder. The truck might need to run in echelon on undivided multi-lane streets. If pushing to the left, snow and ice are pushed to other roadway spaces, which is not allowed. However, on the divided multi-lane streets, if the median strip has enough space to hold snow and ice, a truck on the inner-lane can push to the left whereas a truck on other lanes should push to the right. In this way, trucks are not required to run in echelon. Therefore, the routes can be more flexible.

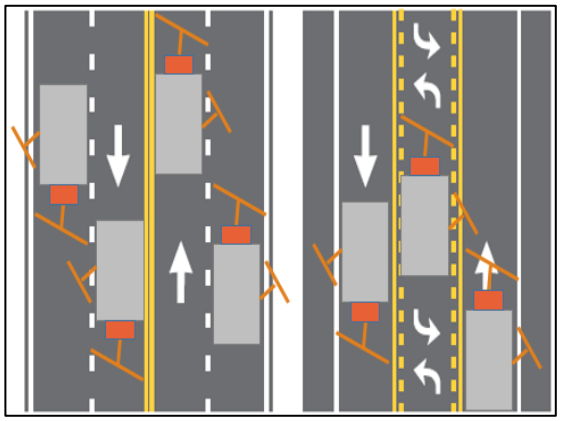

Figure 4.2 *Undivided multi-lane road, all truck with a right-wing plow (roadway diagram picture from the Internet)*

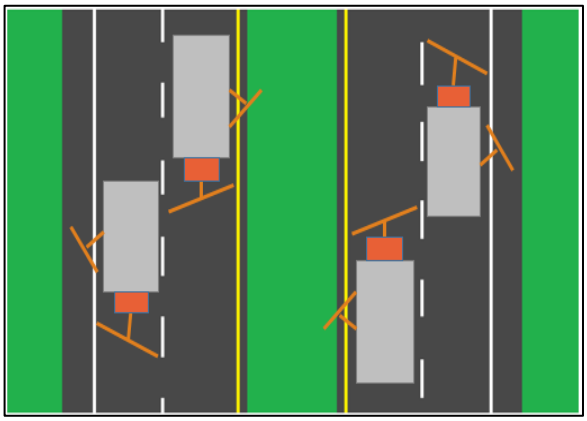

Figure 4.3 *Divided multi-lane road, an inner-lane truck with left-wing plow, an outer-lane truck with a right-wing plow (roadway diagram picture from the Internet)*

4.4 Road Segment Service Cycle Time

The road segment service cycle time represents the service frequency of the road level. Typically, high-traffic road levels should be serviced at a higher frequency, so that the overall impact of adverse weather reduces to a minimum. Some urban roads are required to be serviced every hour. Interstate roadways are serviced every 1.25 hours, most U.S. highways are serviced every 2 hours, and most Iowa roads are serviced every 2.5 hours. Table 4.2 summarizes the road level service cycle time.

Table 4.2 *Service cycle time*

Index	Road Level	Cycle Time (Hours)
Metro	Urban Area	1
A	Interstate	1.25
B	U.S. Highway	2
C	Iowa Road	2.5

4.5 Assumptions

The following assumptions are made in this study.

- The depot network built represents the actual road network.

- The spreading rate is determined before designing routes and the same for all routes. Thus, the spreading length of each type of truck can be calculated. The spreading rate is set as 300 lbs./lane mile.
- The service speed and deadhead speed in urban or rural areas are fixed. Thus, the route travel time can be calculated once a route is given. Urban area service speed is 22 mph, and rural area service speed is 26 mph. Urban area deadhead speed is 32 mph, and rural area deadhead speed is 40 mph.
- This study considers one pass for the depot networks; repeated service of the networks and its consequences is not considered.
- The shortest distance path is used as the deadhead path since the objective is minimizing total travel distance.

CHAPTER 5. SINGLE DEPOT WINTER ROAD MAINTENANCE ROUTING PROBLEM

5.1 Introduction

In this chapter, the SDWRMRP is formulated and solved. The problem is formulated as a directed CARP to minimize total traveling distance, and restrictions including road segments service cycle time, heterogeneous vehicle capacity, fleet size, deadhead and service speed difference, and road-vehicle dependency. Section 5.2 formulates the problem as a mixed integer program. To obtain solutions to instances of practical size, a metaheuristic approach has been developed. Section 5.3 introduces the MA and the parallel scheme of the algorithm. Section 5.4 discusses the results of the case study conducted for 20 depots in District 3.

5.2 Mathematical Model

Given a connected directed graph $G = (V, A)$, where V is a set of nodes and A is a set of arcs, let A_R represent the set of service arcs. Let node v_0 represent the depot node. For each arc (i, j) in G, let c_{ij} denote the cost (distance) of traversing the arc. Let $q_{ij} \geq 0$ be the demand for (i, j). If $q_{ij} > 0$, the arc is said to be service required. Otherwise, it is a non-service demand arc. A heterogeneous truck $h \in H_1 \cup H_2$ has a capacity of Q_h, where H_1 and H_2 represent the two types of vehicles. Let m_1 and m_2 represent the fleet size of the two types of vehicles, separately. Let f_{ij} represent the service cycle time for each arc. Let f_k stand for route cycle time for route k. Let $\mathcal{F}$ represent the set of service cycle times. Let t'_{ij} represent the service time on arc (i, j), and let t_{ij} represent the deadhead time on arc (i, j). Let r_{ij} represent the arc plow direction on arc (i, j). $r_{ij} = 1$ stands for a right-wing plow, and $r_{ij} = -1$ stands for a left-wing plow.

Variables are defined as follows:

$$x_{ij}^{k} = \begin{cases} 1 & if\ route\ k\ service\ (i,j)\ from\ i\ to\ j \\ 0 & otherwise \end{cases}$$

$$y_{ij}^{k} = \begin{cases} 1 & if\ route\ k\ traverse\ (i,j)\ from\ i\ to\ j \\ 0 & otherwise \end{cases}$$

$$u_{kh} = \begin{cases} 1 & if\ the\ hth\ truck\ service\ route\ k \\ 0 & otherwise \end{cases}$$

$$f_k \in \mathcal{F}$$

The formulation is shown below:

$$\sum_{(i,j)\in A, k\in K} c_{ij}(x_{ij}^{k} + y_{ij}^{k}) \tag{1}$$

$$\sum_{(i,j)\in A} (x_{ij}^{k} + y_{ij}^{k}) - \sum_{(j,i)\in A} (x_{ji}^{k} + y_{ji}^{k}) = 0 \qquad \forall k \in K, i \in V \tag{2}$$

$$\sum_{(0,i)\in A} (x_{0i}^{k} + y_{0i}^{k}) = 1 \qquad \forall k \in K, i \in V \tag{3}$$

$$\sum_{(i,0)\in A} (x_{i0}^{k} + y_{i0}^{k}) = 1 \qquad \forall k \in K, i \in V \tag{4}$$

$$\sum_{\forall k\in K} x_{ij}^{k} = 1 \qquad \forall (i,j) \in A_R \tag{5}$$

$$\sum_{(i,j)\in A} q_{ij}\, x_{ij}^{k} \le \sum_{h\in H_1\cup H_2} Q_h u_{kh} \qquad \forall k \in K \tag{6}$$

$$\sum_{h\in H_1\cup H_2} u_{kh} = 1 \qquad \forall k \in K \tag{7}$$

$$\sum_{k\in K} u_{kh} \le 1 \qquad \forall h \in H_1 \cup H_2 \tag{8}$$

$$\sum_{k\in K, h\in H_1} u_{kh} \le m_1 \tag{9}$$

$$\sum_{k\in K, h\in H_2} u_{kh} \le m_2 \tag{10}$$

$$f_k \le f_{ij} x_{ij}^{k} \qquad \forall k \in K, (i,j) \in A_R \tag{11}$$

$$\sum_{(i,j)\in A}(x_{ij}^k t'_{ij} + y_{ij}^k t_{ij}) \le f_k \qquad \forall k \in K \qquad (12)$$

$$x_{ij}^k r_{ij} = x_{i'j'}^k r_{i'j'} \qquad \forall k \in K, (i,j) \in A_R, (i',j') \in A_R \qquad (13)$$

$$x_{ij}^k, y_{ij}^k, u_{kh} \in \{0,1\} \qquad \forall k \in K, h \in H_1 \cup H_2, \forall (i,j) \in A \qquad (14)$$

The objective is to minimize the total travel cost. Constraint (2) is the flow conservation equations for each route. Constraint (3) and (4) state that all routes must start and end at the depot. Constraint (5) ensures the network is fully serviced exactly once. Constraint (6) guarantees that the capacity of each vehicle is never exceeded. Constraint (7) states that each route is served by one vehicle. Constraint (8) states that each vehicle services one route at most. Therefore, some vehicle can be idling at the depot. Constraint (9) and (10) are the fleet size constraints. Constraint (11) states that the service cycle time of a route is determined by the minimum service cycle time of any arc in the route. Constraint (12) ensures each route traveling time never exceeds route cycle time. Constraint (13) guarantees that the arc plow direction (road-truck dependency) within a route is the same. Constraint (14) is the variables constraints.

5.3 Solution Algorithm

The memetic algorithm is similar to the genetic algorithm (GA), such that in MA local search is employed instead of the mutation operator in GA. Each solution is represented by a chromosome, and a set of chromosomes consists of a population set. The evolution process of the population depicts the improvement of the population set concerning the fitness value of chromosomes. At every iteration of the MA algorithm, new chromosomes (children) are produced based on selected old chromosomes (parents), and the desired new chromosomes replace some old chromosomes. The procedure of selecting parents to reproduce, generating children, and replacing some parents by children is called the selection

operator, reproduction operator, and replacement operator, separately. The strategies employed by the operators are often problem-specific.

5.3.1 Algorithm Overview

In this study, incremental MAs are employed, where the population is updated by one chromosome per iteration. Another way of updating the population is the generational MA, where the number of new chromosomes generated per iteration equals the size of the population. Lacomme et al. (2004) show that incremental MA performs better than generational MA on the mixed graph CARP.

The framework of the MA is shown in Figure 5.1. After the initialization of the population, the MA generates a new chromosome by the crossover operator. The local search is performed on the new chromosome at a fixed probability of P_{ls}. Then, the new chromosome replaces an old chromosome in the population. The iterative local search is employed in the replacement function at line 12. The whole population goes through a MS operator every $msFreq$ iterations. The main search process (line 5-16) will be looped $maxRestart$ times. In the restart phase, the local search probability P_{ls} is updated, and new chromosomes are generated.

5.3.2 Solution Representation

5.3.2.1 Chromosome

Solutions are represented as a sequence of arcs. The sequence includes all service demand arcs by assuming the vehicle has unlimited capacity. This sequence is a RPP solution and is known as a giant tour. The giant tour indicates the service order of the service demand arcs with the deadhead arcs omitted from the task sequence. The path that connects consecutive tasks in the giant tour can be derived using the Floyd-Warshall algorithm (Floyd, 1962), which guarantees the shortest path (regarding distance) is used for deadhead. The

Floyd-Warshall algorithm typically only provides the lengths of the paths between all pairs of nodes. However, with simple modifications, it is possible to reconstruct the actual path between any two nodes. Hence, the deadhead time of the shortest distance deadhead path between any two nodes can be calculated.

```
1   pop := Initialization
2   Procedure EvolveIncremental(pop, P_ls, maxIter, maxRestart)
3           useRestartRate := false
4           for nRestart = 1 to maxRestart do
5                   for nIter = 0 to maxIter do
6                           child := Crossover(pop)
7                           Split&Evaluate(child)
8                           if U[0,1] < P_ls
9                                   LocalSearch(child)
10                                  Split&Evaluate(child)
11                          end if
12                          Replacement(child, pop)
13                          if nIter % msFreq = 0
14                                  MergeSplit(pop)
15                          end if
16                  end for
17                  if nRestart ≤ maxRestart
18                          if not useRestartRate = True
19                                  P_ls = RestartLocalSearchRate
20                          end if
21                          Restart(pop)
22                          Split&Evaluate(new chromosomes in pop)
23                  end if
24          end for
25  return best solution
```

Figure 5.1 *Pseudo code of MA*

5.3.2.2 Split procedure and solution evaluation

The split procedure is the critical element of giant tour-based heuristics. Given a giant tour S with t tasks, the split procedure partitions the service sequence into routes. Then, the fitness value of a chromosome can be evaluated. The fitness value, also known as the cost, is

the total traveling distance of all routes, including the deadhead distance between consecutive arcs, the deadhead distance between arcs and depots, and the service distance of arcs.

Figure 5.2 provides one example for a task sequence with four tasks (solid arrows) and demands in brackets. Dashed arrows indicate deadhead shortest paths. For simplicity, the numbers next to arrows represent travel distance, service time for tasks, and deadhead time for deadhead path. Assume the truck capacity $Q = 9$, and all tasks should be serviced every $D = 60$ time units.

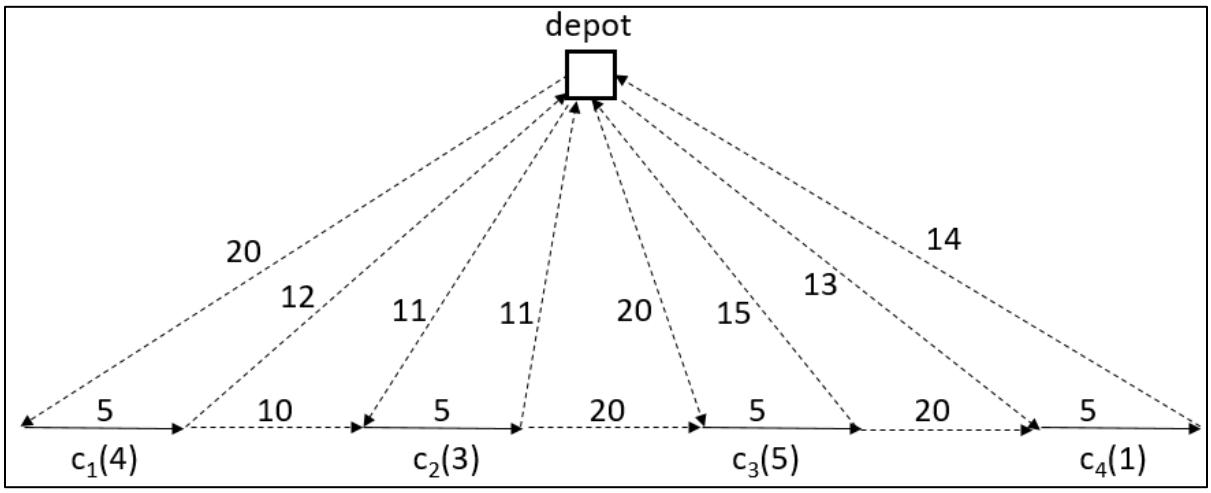

Figure 5.2 *Chromosome giant tour with four tasks*

The split procedure builds an auxiliary graph H with $t + 1$ nodes indexed from 0 to t. Figure 5.3 shows examples of the auxiliary graph under different constraints. Each subsequence of tasks $(c_i, \dots, c_j)$ that can be treated as one route is modeled by one arc that connects nodes $(i - 1, \dots, j)$ in the auxiliary graph H. Each arc in H indicates the cost of that feasible route. Each node in H records the minimum cost of visiting the node, with the corresponding predecessor node.

The first graph considers only the capacity constraint. For example, arc c_1 indicates the travel time and travel distance is $37 = 20 + 5 + 12$, which is the sum of deadhead from

the depot to c_1, the time/distance of c_1, and the deadhead from c_1 back to the depot. The arc $c_1c_2 = 20 + 5 + 10 + 5 + 11 = 51$. There is no arc $c_1c_2c_3$ because the demand for such route exceeds Q. By looping through the t tasks, the minimum cost to reach the $t + 1$ nodes in H can be found. The shortest path from node 0 to node $t + 1$ in Figure 5.3-I (in **boldface**) corresponds to the optimal split of the chromosome. Thus, the optimal routes are $[c_1c_2]$ and $[c_3c_4]$.

The second graph considers both capacity and duration constraints. The road segment service cycle time is guaranteed by the route duration. Any route duration should not exceed the minimum cycle time of any arc in that route. This means the route duration limit is determined by the highest cycle time of arcs in the route. Adding more constraints means eliminating the violated arcs in H. In Figure 5.3-II, route $c_2c_3c_4$ and route c_3c_4 are no longer feasible because they exceed route duration D. The optimal routes change to $[c_1c_2]$, $[c_3]$, and $[c_4]$.

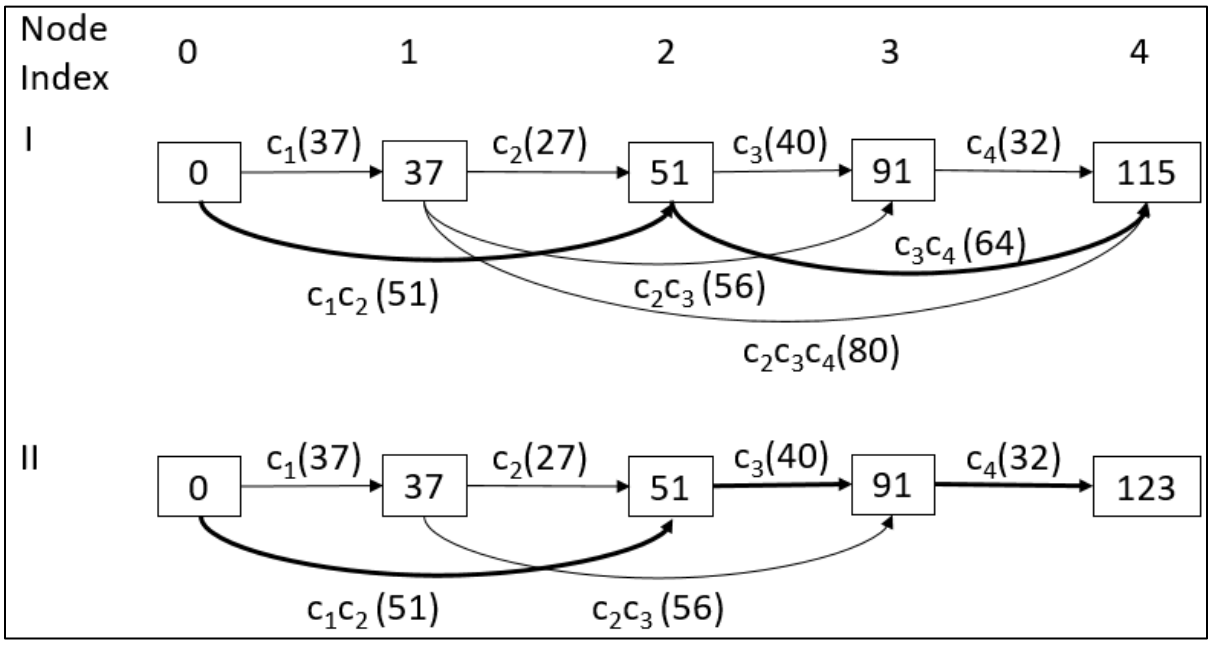

Figure 5.3 *Auxiliary graph and shortest path. I—only capacity constraint, II—capacity and duration constraints*

The road-vehicle dependency constraint can also be incorporated in the split procedure. To be more specific, if the task plow direction is added to each task in the giant tour in Figure 5.2, then in the auxiliary graph H, the arc that plowing tasks with different wing plow directions will be removed. However, it is more efficient to execute the program twice, once servicing the right-wing plow and once servicing the left-wing plow. It can save a great amount of time, especially for the algorithm of local search and the penalty function (in section 5.3.4). The only work is to set the right-wing roadways not service demanded when solving the problem for the left-wing plow, and vice versa. Hence, the road-vehicle dependency constraint will not be mentioned in the following sections.

The split procedure discussed above assumes the homogenous capacity and unlimited fleet size. Duhamel et al. (2011, 2012) and Prins et al. (2014) developed a split procedure for heterogeneous capacity and limited fleet size. The problem of partitioning the acyclic auxiliary graph H with a limited number of heterogeneous vehicles belongs to the resource-constrained shortest path problem, which is also NP-hard (Prins et al., 2014). The idea of building the auxiliary and finding the shortest path from back to start is the same as above. The constraints of capacity of each type of vehicle, the fleet size, as well as the route duration should be taken into consideration while generating arcs in the auxiliary graph. The difference is since multiple types of capacity are involved, each node in the auxiliary graph now maintains a set of labels in addition to the one with the least cost. The set of labels are actually the Pareto frontier regarding the number of vehicles in use for each type of vehicle.

A Pareto front node (PFN) contains the information of the cost (travel distance), time, predecessor node index, predecessor PFN number, current PFN number, the total single-axle truck used, and total tandem-axle truck used. For example, a PFN node with value {50, 30,

$10,\ 6,\ 3,\ 2,\ 4\}$ indicates that the cost of reaching to current node index using this route is 50, the time is 30. The predecessor of the route is the 6th PFN in node $index\_i = 10$. The current PFN number is a sequential number that tracks the index of PFN nodes generated for node $index\_j$. The current node is the 3rd PFN that ever generated at $index\_j$. Total of 2 single-axle truck and 4 tandem-axle truck is used.

The newly generated label will be checked of its dominant status (Prins et al. 2014) of the existing labels in the node. After the check, PFN that remain in the node are non-dominated by other PFNs in the node.

The split algorithm is shown in Figure 5.4. The PFN of the auxiliary graph remained in a 2-D vector $VCost$ with the outer index indicating the auxiliary node index and the inner index representing the current PFN number. The algorithm uses $VCost$ to keep track of the route cost and time starting from the depot source node. Line 4 and line 8 use two loops to track the cost and time of servicing the tasks from $arcIndex\_i$ to $arcIndex\_j$, building the auxiliary graph, which is similar to Lacomme et al. (2004). Line 10 to 12 check the cycle time of servicing the $arcIndex\_j$, and the route travel time must not exceed the minimum cycle time that is serviced in the route. Line 14 to 30 updates the travel distance and time of the current route. Line 31 to 36 updates the PFN nodes of node at Index_j in the auxiliary graph, which is similar to Prins et al. (2014).

Figure 5.5 represents the PFN update procedure. Line 2 first checks if the route satisfies all constraints. Line 3 to 8 guarantee that the number of trucks in use is less than the available fleet size of the depots. Line 10 represents that if all the constraints are met, the current route is feasible, and the loop (line 8 in Figure 5.4) for $arcIndex\_j$ can proceed on the next $arcIndex\_j = arcIndex\_j + 1$ at least for the current route. Lines 11 to 28 update

the information for the current PFN. Line 29 uses the dominate rule (Prins et al., 2014) to update the Pareto frontier nodes for auxiliary node $Index\_j$.

```
1   Procedure SplitSDW(child)
2   arcMax = the size of the giant tour
3   PFN initialNode = {0, 0, 0, 0, 0, 0, 0}, VCost[0][0]  = initialNode
4   for arcIndex_i = 0 to arcMax - 1
5       arcIndex_j = arcIndex_i
6       loadK = 0, LOSK = inf, costK= 0, timeK= 0
7       stopLoop = false
8       do  // next arcIndex_j
9           loadK = loadK + demand[arcIndex_j]
10          if LOSK > LOSMins[arcIndex_j]
11              LOSK = LOSMins[arcIndex_j]
12          endif
13          stopLoop = true
14          if arcIndex_j = arcIndex_i
15              costK += d_{sourceNode, arcIndex_i_begin}
16              costK += d_{arcIndex_i_end, sourceNode}
17              costK += distance[arcIndex_i]
18              timeK += t_{sourceNode, arcIndex_i_begin}
19              timeK += t_{arcIndex_i_end, sourceNode}
20              timeK += time[arcIndex_i]
21          else
22              costK -= d_{arcIndex_j-1_end, sourceNode}
23              costK += d_{arcIndex_j-1_end, arcIndex_j_begin}
24              costK += d_{arcIndex_j_end, sourceNode}
25              costK += distance[arcIndex_j]
26              timeK -= t_{arcIndex_j-1_end, sourceNode}
27              timeK += t_{arcIndex_j-1_end, arcIndex_j_begin}
28              timeK += t_{arcIndex_j_end, sourceNode}
29              timeK += time[arcIndex_j]
30          endif
31          PFNSizeofArcIndex_i = size of VCost[arcIndex_i]
32          for PFNIndexofArcIndex_i = 0 to PFNSizeofArcIndex_i – 1
33              for vehType = (# of fleet type -1) to 0
34                  updatePFNnodes( )
35              endfor  // vehicle type
36          endfor  // Pareto Front Node of arcIndex_i
37          arcIndex_j = arcIndex_j + 1
38      while (arcIndex_j < arcMax) && (stopLoop == false)  // arcIndex_j service the next arc
39  endfor  // arcIndex_i
```

Figure 5.4 *Split procedure for SDWRMRP*

```
1   Procedure updateSDWPFNnodes( )
2   if (TimeK < LOSK) && (loadK <= capacityHetero[vehType])
3       if ( ( (vehtype = 0) && (VCost[arcIndex_i][PFNIndexofArcIndex_i].SGL < fleetSize[vehtype]) )
4       || ( (vehtype = 1) && (VCost[arcIndex_i][PFNIndexofArcIndex_i].TDM <  fleetSize[vehtype]) ) )
5           // fleet size satisfied, do nothing
6       else
7           continue
8       endif
9       // update current PFN
10      stopLoop = false
11      currentNode.Cost = VCost[arcIndex_i][PFNIndexofArcIndex_i].Cost + costK
12      currentNode.Time = VCost[arcIndex_i][PFNIndexofArcIndex_i].Time + timeK
13      currentNode.SGL = VCost[arcIndex_i][PFNIndexofArcIndex_i].SGL
14      currentNode.TDM = VCost[arcIndex_i][PFNIndexofArcIndex_i].TDM
15      currentNode.PredecessorNodexIndex = arcIndex_i
16      currentNode.PredecessorPFNNumber = VCost[arcIndex_i][PFNIndexofArcIndex_i].currentPFNNumber
17      if vehType = 0
18          currentNode.SGL += 1
19      else
20          currentNode.TDM += 1
21      endif
22      // track the last PFN node of arcIndex
23      PFNSizeArcIndex_j = size of VCost[arcIndex_j + 1]
24      if PFNSizeArcIndex_j != 0
25          currentNode.currentPFNNumber = VCost[arcIndex_j + 1][PFNSizeArcIndex - 1].currentPFNNumber + 1
26      else
27          currentNode.currentPFNNumber = 0
28      endif
29      update the dominates nodes of node arcIndex_j in the auxiliary graph
30  endif
```

Figure 5.5 *Update PFN procedure for SDWRMRP*

After the algorithm terminates, the optimal split can be retrieved by tracking the predecessor's PFN node and number from the last node in the auxiliary graph.

5.3.3 Initial Solution

The population is initialized by path-scanning (PS). The path-scanning procedure is first introduced by Golden et al. (1983) for CARP on an undirected graph and has been extended by several authors. For example, Lacomme et al. (2004) provided a PS algorithm

for mixed graph CARP, Xing et al. (2010) proposed a random PS algorithm for MDCARP, and Willemse and Joubert (2016) created a PS algorithm for CARPTIF.

The PS procedure builds one route at a time. All service demand arcs consist of a set $freeArcs$. In constructing each route, the algorithm searches all arcs in the $freeArcs$ to find the closest one to the current route's end. The most promising arc is appended to the current route's end with all constraints satisfied (capacity of the current truck, route service duration), and this arc is removed from the $freeArcs$. If no arcs can be appended to the current route, the current route ends and the next route begins searching from the depot. If a tie appears, tie-break rules apply when several arcs are incident to the route, including:

1. maximize $dist(u)/demand(u)$;
2. minimize $dist(u)/demand(u)$;
3. maximize return cost from $endnode(u)$ to the depot;
4. minimize return cost from $endnode(u)$ to the depot;
5. if the truck is less than half-full, then apply rule 3, otherwise, rule 4; and
6. randomly choose the tie-break rule from 1 to 5 by a uniform distribution

where $dist(u)$ represents the arc length of u, $demand(u)$ represent service demand of u, $endnode(u)$ represent the end node of arc u.

A new PS algorithm is developed in this study to incorporate the constraint of road level cycle time. When constructing the routes, higher road level arcs are serviced first. In this study, for example, the PS will start by searching the arc set of "urban," that is, the $freeArcs$ is loaded as the arc set of "urban." After $freeArcs$ is empty and the "urban" routes have been built, the $freeArcs$ is loaded with the arc set of "interstate," and so on with "highway" and "Iowa road" until all service demand arcs have been serviced. The population

is generated by the new PS algorithm. The first five chromosomes use the tie-breaking rules of the first five types, respectively, and the rest of the chromosomes use the sixth tie-breaking rule.

5.3.4 Operators

5.3.4.1 Crossover

Two parent chromosomes P1 and P2 are randomly selected, and then the longest common substring crossover (LCSX) produces the next offspring. The idea of the LCSX is that adequate chromosomes often share common substrings. Such substrings might already have the deadhead minimized between consecutive tasks. The longest common substring can be found using dynamic programming in $O(t^2)$.

Figure 5.6 shows an example of the LCSX. The longest common substrings between P1 and P2 are arcs (1, 7) and (3, 8). These two substrings are copied to the child chromosome C1 and C2, respectively, and without changing positions. Then, the rest of the P1 tasks are left shifted to fill the empty positions in C2, and the same procedure is performed on P2 to C1. Finally, only one child is randomly selected and kept.

P1:	1	7	6	4	2	3	8	5
P2:	4	5	6	1	7	2	3	8
C1:	1	7	4	5	6	3	8	2
C2:	6	4	2	1	7	5	3	8

Figure 5.6 *Example of LCSX*

5.3.4.2 Local search

The local search is performed at a probability of P_{ls}. All pairs of tasks try the following five types of moves, resulting in a time complexity of $O(t^2)$. During each iteration of the local search, all types of moves are scanned in sequence, and the first improving move

updates the current solution. Then, the next iteration is continued. The local search stops when all pairs of tasks are scanned.

L1: Move task u after task v.

L2: Move two consecutive tasks (u, v) after task w.

L3: Swap task u and v.

L4: Swap task u and consecutive tasks (v, w).

L5: Swap the consecutive tasks (u, v) and consecutive tasks (w, z).

The local search performs on the routes rather than the chromosome. Because if the local search performs on the chromosome directly without route delimiters, the split procedure must be employed after each run of the local search to evaluate the solution, and more time is spent on evaluating the chromosome instead of improving the chromosome. By performing a local search on routes, evaluation of the local search can be completed in constant time.

During the local search, infeasible solutions (violation of capacity or route duration) are allowed and penalized by their violations. The feasible solution might be isolated in the solution space. It is difficult for the local search to move from one feasible region to another feasible region unless they are very close to each other. Infeasible solutions act as bridges to connect different feasible regions. If the local search is allowed to accept infeasible intermediate solutions, the search space is much larger, and the search path between two feasible solutions is likely to be connected by infeasible solutions.

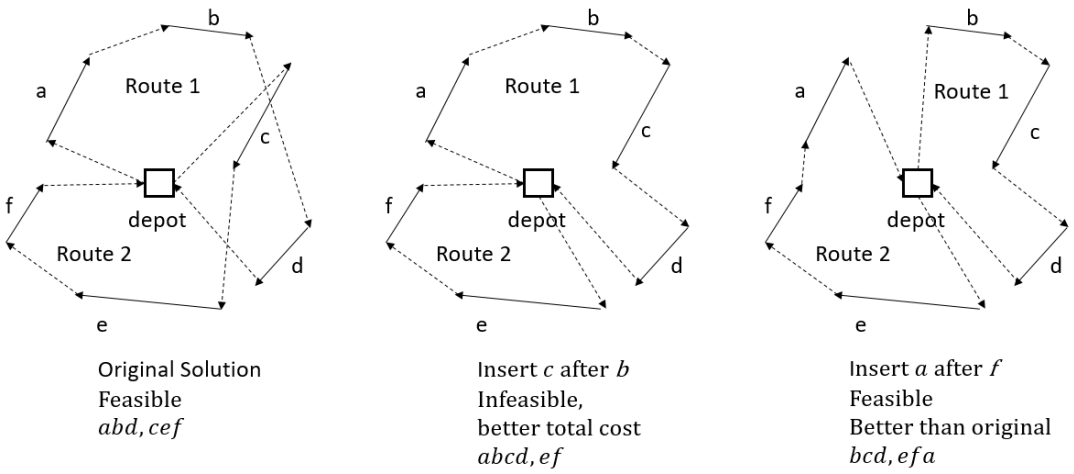

Figure 5.7 *Example of local search – swap operation*

Figure 5.7 illustrates an example of the local search. The original chromosome has two routes. Assume the capacity is three arcs, and only consider the capacity constraint here. By inserting arc c after arc b, now route 1 violates the capacity constraint, but the total travel distance of all routes decreased. So, the total traveling and penalty cost of this solution $abcd, ef$ can be lower than the original cost. Hence, the local search accepts this intermediate solution and performs the next iteration. Then, in the right graph in Figure 5.7, arc a is inserted after arc f, resulting in a feasible solution. This solution updates the old chromosome, and the next local search iteration begins. Accepting an infeasible intermediate solution within local search could largely expand the search space during a local search.

If the solution $abcd, ef$ does not violate any restrictions, it is a feasible solution. However, it can be imagined that this might not be the optimal partition of this chromosome. The routes abc, def might embrace a better split, depending on the deadhead distance between the arcs and the depot. Thus, performing the split procedure after the return of the local search can sometimes provide better results. Figure 5.8 shows the local search procedure. The new route cost and penalty value are updated in the $PerformLocalSearch$ function.

```
1   Procedure LocalSearch(oldRoutes)
2           currentRoutes := oldRoutes
3           localSearchType := 1
4           repeat
5                   fitnessImproved, penaltyImproved := false
6                   calculate penalty parameters α, β for currentRoutes
7                   for i = 1 to t do
8                           for j = 1 to t do
9                                   (newRouteFitness, newRoutePenalty) := PerformLocalSearch
10  (currentRoutes, localSearchType, i, j)
11                                  if (newRouteFitness < currentRouteFitness) and (newRoutePenalty = 0)
12                                          oldRoutes = newRoute
13                                          currentRoutes = newRoute
14                                          fitnessImproved = true
15                                          break
16                                  else if newRouteFitness + newRoutePenalty < currentRouteFitness
17                  + currentRoutePenalty
18                                          currentRoutes = newRoute
19                                          penaltyImproved = true
20                                          break
21                                  endif
22                          endfor
23                          if (fitnessImproved = true) or (penaltyImproved = true)
24                                  break
25                          endif
26                  endfor
27                  if (fitnessImproved = true) or (penaltyImproved = true)
28                          localSearchType = 1
29                          continue
30                  else
31                          localSearchType = localSearchType + 1
32                  endif
33          until (fitnessImproved = false) and (penaltyImproved = false)
34  return oldRoutes
```

Figure 5.8 *Local search procedure*

5.3.4.3 Penalty function

The quality of a solution is evaluated by a weighted sum of the violated constraints and the fitness cost, given in (16)

$$f(S) = TC(S) + \alpha * CapVio(S) + \beta * DurVio(S) \qquad (16)$$

where $TC(S)$ is the total cost of S, $CapVio(S)$ is the total capacity violation of S, $DurVio(S)$ is the total route duration violation of S. The penalty parameters that balance the tradeoff between cost and violations are α and β. The solution with the lower value of $f(S)$ is considered the better solution obtained by local search.

The penalty parameters should be adaptive during a local search so that the magnitude of penalization is updated according to current solution quality. It should also normalize between the cost and violation terms, making them within the same magnitude. When performing the local search around solution S, the penalty parameters α and β are first calculated for the current solution. The capacity penalty parameter α is calculated as

$$\alpha = \frac{TC(S_{best})}{Q} * (\frac{TC(S_{best})}{TC(S)} + \frac{CapVio(S)}{Q} + 1)$$

where S_{best} is the best feasible solution found so far. $TC(S_{best})/Q$ is the normalization factor. $TC(S_{best})/\,TC(S)$ makes α decrease while $TC(S)$ increases. Therefore, if the $TC(S)$ is close to $TC(S_{best})$, the solution S is heavily penalized, but if the $TC(S)$ is much worse (much higher fitness value) than $TC(S_{best})$, the solution S is lightly penalized. $CapVio(S)/Q$ makes α increase while total capacity violations increase. The term "1" is added here to ensure a sufficiently large α in case $TC(S)$ is a feasible solution with a very high cost. During a local search, α is halved if a feasible solution is generated in five consecutive iterations, and is doubled if an infeasible solution is found in five consecutive iterations.

The duration penalty parameter β is calculated as

$$\beta = \frac{TC(S_{best})}{\mathcal{F}_{avg}} * (\frac{TC(S_{best})}{TC(S)} + \frac{DurVio(S)}{\mathcal{F}_{avg}} + 1)$$

where the $\mathcal{F}_{avg}$ is the average of the cycle time in the demand network.

5.3.4.4 Replacement

In the replacement procedure, the child chromosome replaces one parent chromosome in the population. Figure 5.9 illustrates the pseudo code for the replacement procedure. If the child chromosome is unique, which means the population does not contain the same fitness value, binary tournament selection is employed to select the parent chromosome. Two chromosomes are randomly picked from the population, and the chromosome with higher cost value is chosen to be replaced. If the child chromosome is not unique, one double swap (local search move type 5) is performed on the child with the swapped arcs randomly selected. Then, if it is still a clone, it replaces the clone. Otherwise, it replaces the worse parent chosen by the binary tournament selection. Iterative local search is performed if the cost of the new solution is close enough to the old solutions. In ILS, the child chromosome receives the single swap and double swap local search iteratively. A better solution is kept during the ILS procedure, and the best solution is returned.

5.3.4.5 Merge-Split

At every $msFreq$ iteration, the whole population undergoes the MS operation. Figure 5.10 illustrates the pseudo code. The algorithm iteratively selects two routes in a chromosome. It first merges all arcs in the two routes $i\ and\ j$ into the $freeArcs$ set, then applies the PS algorithm on this $freeArcs$. After the PS, the newly generated routes replace the original two routes in the chromosome if the new partial routes' distance is less than the sum of the distance of the original two routes. Then, the split procedure partitions the new giant tour and results in new routes. Since the split procedure generates the optimal split of the giant tour, the fitness is guaranteed to be less than or equal to the previous fitness value. The program checks each pair of the routes by using an increasing value of $i\ and\ j$. The

i *and* j will not be reset even if a better chromosome is found, which can restrict the time complexity to $O(r^2)$, where r equals the number of routes in the chromosome.

```
1   Procedure Replacement(child, pop)
2       worseParent := BinaryTourmentSelection(pop)
3       Insert all chromosome's fitness value into popFitnessSet, except the worseParent's fitness
4       // Iterative Local Search
5       if (childFitness ≥ bestFitness) and (childFitness ≤ 1.005*bestFitness)
6           newChild := child;   newChild' := child
7           for iter = 1 to nILS do
8               SingleSwap(newChild, i, j)
9               Split&Evaluate(newChild)
10              PerformLocalSearch (newChild, i, j)
11              Split&Evaluate(newChild)
12              if (newChildFitness < popBestFitness( ))
13                  pop[worseParent] := newChild
14                  child = newChild
15              endif
16              DoubleSwap(newChild', i, j)
17              Split&Evaluate(newChild')
18              PerformLocalSearch (newChild', i, j)
19              Split&Evaluate(newChild')
20              if (newChild'Fitness < popBestFitness( ))
21                  pop[worseParent] := newChild'
22                  child = newChild'
23              endif
24          endfor
25      endif
26      // replace an old chromosome
27      if (childFitness not in popFitnessSet)
28          pop[worseParent] := child
29      else
30          newChild'' := DoubleSwap(child, i, j)
31          Split&Evaluate(newChild'')
32          if (newChild'Fitness not in popFitnessSet)
33              pop[worseParent] := newChild''
34          else
35              pop[clone] := newChild''
36          endif
37      endif
38  return pop
```

Figure 5.9 *Replacement procedure*

```
1   Procedure MergeSplit(pop)
2       for k = 1 to popSize
3           oldChromosome = pop[k]
4           newChromosome = oldChromosome
5           freeArcs = Ø
6           i = 0
7           while i + 1 < number of routes in the newChromosome
8               j = i + 1
9               while j + 1 < number of routes in the newChromosome
10                  insert arcs of route i and j into freeArcs
11                  partialRoutes = pathScanning(freeArcs)
12                  if (partialRoutes.fitness < route distance i + route distance j)
13                      remove route  i in oldChromosome
14                      remove route  j in oldChromosome
15                      newChromosome = insert partialRoutes in oldChromosome
16                      Split&Evaluate(newChromosome)
17                      oldChromosome = newChromosome
18                  endif
19                  j = j + 1
20              endwhile
21              i = i + 1
22          endwhile
23          pop[k] = tempChromosome
24      endfor
25  return pop
```

Figure 5.10 *MS procedure*

5.3.4.6 Restart phase

In the restart phase, half of the population is replaced. The population is first sorted ascendingly, then the even order chromosomes (i.e., the second, fourth …) are replaced by new chromosomes generated by the PS algorithm with the 6th tie-break rule.

5.3.5 Parallel Metaheuristic

A parallel metaheuristic approach is developed to addresses the solution quality and computation efficiency of the algorithms. In this study, the parallel metaheuristic is designed on the iteration level with a master-worker model. An elite chromosome set is maintained by a master CPU. Other worker CPUs work on a population set and perform the MA algorithm.

The worker CPU communicates with the master CPU at a predefined frequency. The elite set replaces the partial worker population. The workers return newly evaluated solutions to the master. Figure 5.11 shows the parallel MA diagram.

The workers send their best chromosome immediately after every MS operation and replace the worst chromosome in the worker's population by the chromosome sent by the master. The master holds only one chromosome, and this chromosome is sent to the worker immediately after the master receives the communication of a worker. Then, the master compares the two chromosomes: the one it is currently holding and the other received from the worker. Only the best chromosome is left in the master. In this way, all workers remain largely independent of each other but kept updates to the best known. In addition, all chromosomes in the worker still may be unique in the population.

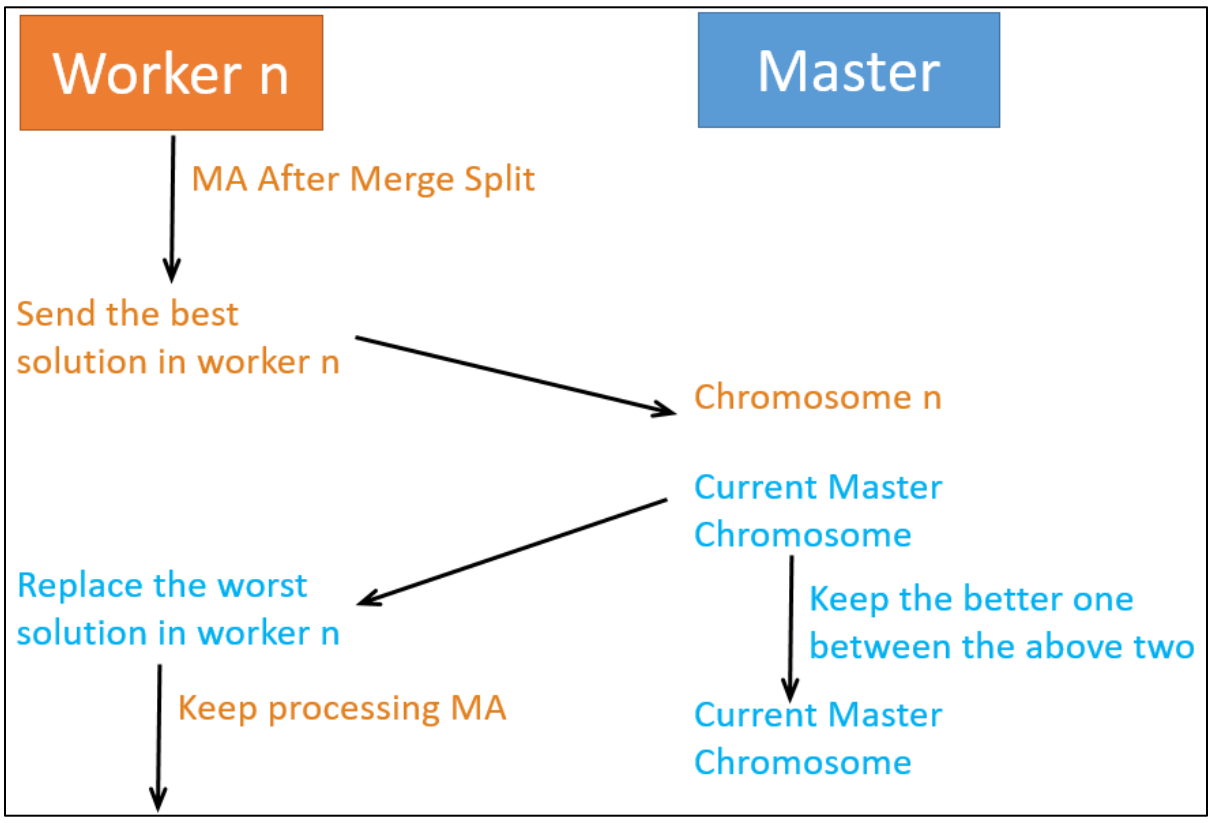

Figure 5.11 *Parallel MA scheme*

5.4 Computational Experiments

5.4.1 Experiments Setups

The MA algorithm proposed in this chapter is implemented in C++ and executed on an Intel® Xeon® CPU E5-1620 v2 3.7 GHz. The network data structure is a directed graph implemented by using LEMON, an open source C++ graph template library. Throughout the experiments, MA adopted the parameters shown in Table 5.1.

Table 5.1 *Summary of the parameters of MA*

Name	Description	Value
maxIter	number of iteration in each phase	10,000
maxRestart	number of restarts	10
msFreq	MS every $msFreq$ iteration	1000
$\boldsymbol{P_{ls}}$ **main phase**	local search rate in the main phase	0.1
$\boldsymbol{P_{ls}}$ **restart phase**	local search rate after restart phase	0.2
psize	population size	30
nILS	number of iteration of ILS	3

5.4.2 Test Instances

The traffic network for the 20 depots in District 3 is used as the test instance of this study. The networks are built by ArcGIS and exported by Python as edge lists. Two networks, “Alton” and “Rock Rapids,” have two depots. So, the routing problem for Alton and Rock Rapids is actually a multi-depot problem, which is studied in the next chapter. Results for the single depot winter maintenance operation are shown for the remaining 18 depots.

5.4.3 Computation Results

5.4.3.1 Result summary

The MA algorithm was run for each network. Key results are presented in Table 5.2. The service distance and the test run distance are the same as in Table 3.2. The optimized

distance is the optimized travel distance using the proposed MA algorithm. The fleet size is the total number of trucks in the current fleet, as in Table 4.1.

The optimized fleet is the number of trucks needed in the optimized solution. The result is obtained by comparing the optimized routes with the current routes. “Change boundary” or “Add number of lanes” means the service network changed during the study period of this research. “Change route due to time constraint” indicates that some of the current routes might violate the service cycle time. “Change route due to capacity constraint” indicates that some of the current routes might violate the capacity constraint. Hence, the optimized routes may result in longer deadhead distance than the current routes. “Saving deadhead” or “Saving truck” means the current routes can be improved. In other words, the optimized routes have less deadhead distance compared to the current routes. “Currently efficient” means the current routes satisfy all constraints and are efficient.

Eight of eighteen depots’ test run distance does not have duplicate service. Hence, their test run distance can be used as a baseline of the current operation. The total of the test run distance of these eight depots is 2445.3 miles, and corresponding optimized distance is 2122.8 miles. Therefore, the deadhead distance saving of the optimized distance comparing to the current test run distance is 13.2%.

A detailed arc-by-arc service route map is drawn after the solution is provided. Since each polyline in the RAMS network represents all traffic lanes in one direction, the service route map should indicate the number of lanes to be serviced. After all routes for the depot are drawn, the collection of routes can be compared to the current routes.

Table 5.2 *Depot service lane miles, test run travel distance and optimized distance in miles, current and optimized fleet sizes*

Depot Name	Service Distance	Test Run Distance	Optimized Distance	Fleet Size	Optimized Fleet Size	Result
Ashton	307.4	738.0*	499.6	15	10	Change boundary, Change route due to time constraint
Carroll	151.5	255.4	205.4	5	4	Saving deadhead and truck
Cherokee	180.3	219.1	212.01	7	6	Saving deadhead
Correctionville	192.9	227.0*	254.7	6	6	Add # of lanes, Change route due to capacity constraint
Denison	237.4	320.1*	252.4	8	6	Change boundary
Emmetsburg	150.2	225.0*	169.5	5	4	Saving deadhead
Ida Grove	128.6	177.0*	139.3	4	3	Saving truck
Le Mars	264.0	375.4	335.1	9	7	Saving deadhead
Onawa	298.3	504.1	475.7	9	10	Saving deadhead, Change route due to time constraint
Pocahontas	229.6	298.5*	259.6	7	6	Saving deadhead
Rockwell City	199.4	259.7	276.2	7	6	Change route due to time constraint
Sac City	185.6	434.1*	313.8	6	7	Add # of lanes
Sioux City Hamilton	189.7	504.7*	332.6	9	10	Change route due to time constraint
Sioux City Leeds	157.0	376.0*	251.1	7	8	Change route due to time constraint
Sloan	126.6	163.7	162.9	6	5	Currently efficient
Spencer	171.5	297.3*	207.0	6	5	Currently efficient
Spirit Lake	176.7	424.9	223.8	9	5	Saving deadhead and truck
Storm Lake	163.8	243.1	231.7	7	5	Saving deadhead and truck

* Test run duplicates on multiple road segments

As an example of deadhead distance saving, Figure 5.12 illustrates the current operation routes for the Emmetsburg depot. Routes are highlighted by different colors. The depot is located in the center of the map. The purple route “A31411” belongs to the “Spencer” depot. Figure 5.13 shows the optimized Emmetsburg routes. The saving comes from route 2, illustrated by the arrow. It services the routes east of the depot as a whole. In contrast, the current orange route “A33311” and blue route “A31320” share this area.

As an example of route change, Figure 5.12 illustrates the current operation routes for Rockwell City depot. Routes are highlighted by a different color. The depot is located in the center of the map. The turquoise route “A32018” and blue route “A33486” service East U.S. 20 of Rockwell City. Since these road segments are divided roads, one of the routes must be a left-wing plow route and the other a right-wing plow route, and the current right-wing route was found too long to be serviced. Figure 5.13 shows the optimized Rockwell City routes. Route 5 is a left-wing plow route and is capable of servicing the route. Alternatively, routes 1 and 3 share the right-wing plow route. Because these road segments have multiple turning lanes, the route is too long to be serviced by using only route 1, if not violating the service cycle time constraint. Hence, route 3 should service turning lanes left by route 1. Deadhead distance can be reduced by making route 2’s service begin from the depot.

By comparing the current routes and the optimized routes, some observations can be drawn about the result. First, most of the depots are well located in their service map. They are almost in the center of the map and close to the nearest service demand road segment.

Second, several general rules can be used to help manual route design: 1) The route service beginning point should be placed as close to the depot as possible to avoid deadhead. For example, the route 2 in the Emmetsburg depot. Emmetsburg adopted the current route

that had been designed for a different service responsibility and had not changed it ever since. This indicates that result of the optimized route can help the state agencies to identify insufficient plans.

2) For multiple routes traversed on the same road segments, if the road segment has more than one lane in any direction, all routes should take a share of the service to decrease deadhead.

5.4.3.2 Practical concerns

The real operation condition could influence the choice of best route design. In this section, the concern of operating speed is discussed first, followed by a sensitivity analysis of spreading rate, and some other practical issues are discussed last.

5.4.3.2.1 Speed and service cycle time

Average service and deadhead speed are used in this study. However, if the driver drives a little faster, some routes might just be feasible. Table 5.2 shows that the current plan of five depots violates the cycle time constraint. Table 5.3 illustrate the number of routes in the current plan that violate the cycle time constraint. This means the truck that servicing the "infeasible" route needs to drive faster than the assumed speed if the state agency wants to keep their current plan.

As an example, Figure 5.16 illustrates the optimized result of Onawa depot. Route 2, 4, 9, 10 services a 4 lane Interstate roadway. Each of them services only one lane one direction of the roadway and has to deadhead on the other direction to make the route meet cycle time constraint, which is 75 minutes. The current plan operations on Skyhawk, however, uses only two routes, where each them service one lane in both directions. Consequently, the travel time of the current routes are longer, which are around 80 to 85 minutes.

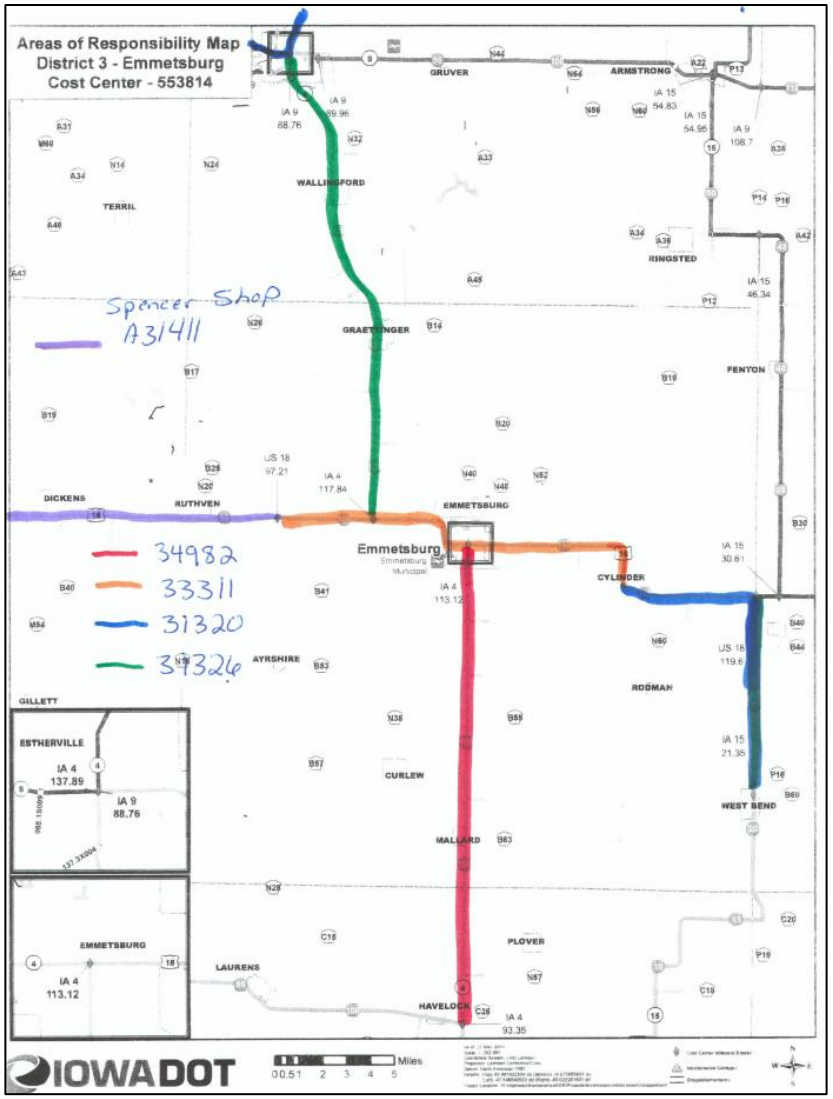

Figure 5.12 *Current Emmetsburg routes.*

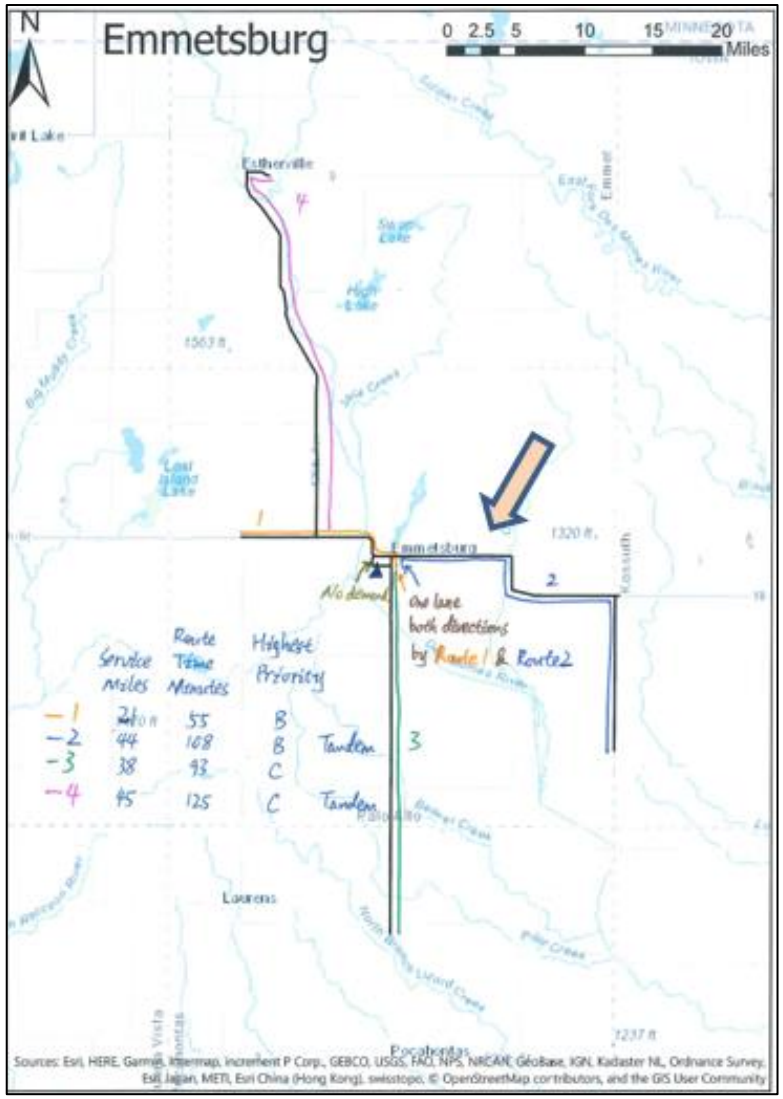

Figure 5.13 *Optimized Emmetsburg routes.*

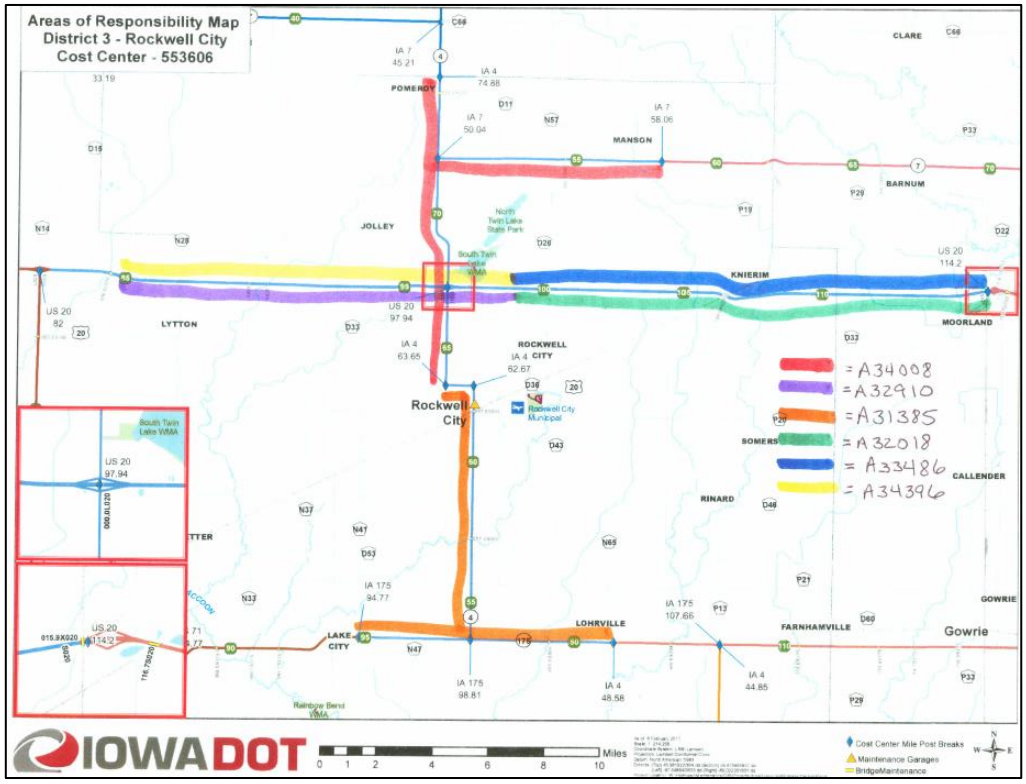

Figure 5.14 *Current Rockwell City routes*

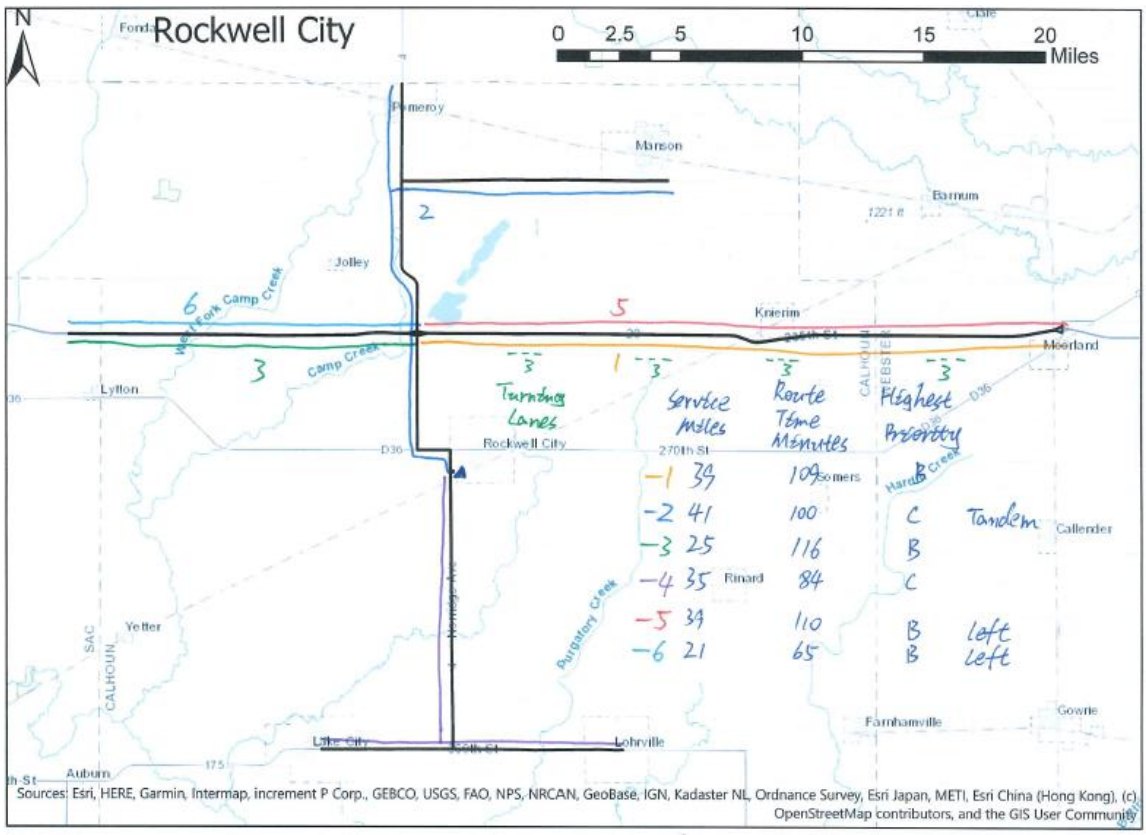

Figure 5.15 *Optimized Rockwell City routes*

The service cycle time is a guideline that given by Iowa DOT. Minor violation of it during the real operation does not have significant consequence. A better way to incorporate this character could use a penalty on the objective functions. Future research could focus on this problem.

Table 5.3 *Number of routes violate cycle time constraint*

Depot Name	# of current routes violated cycle time constraint
Ashton	1
Onawa	2
Rockwell City	1
Sioux City Hamilton	1
Sioux City Leeds	1

5.4.3.2.2 Spreading rate

A sensitivity analysis of the spreading rate is conduct in this study. The spreading rate is set as 150, 200, 250 lbs. per lane mile and solved the SDWRMRP for all instances. Two of the depots found a better result if using a lower spreading rate instead of 300 lbs. per lane mile. Table 5.4 presents the travelling distance under different spreading rates. It is found that under low spreading rate, Correctionville and Pocahontas can have significantly better solution routes. The savings of the Correctionville and Pocahontas routes both have a cycle time of 2.5 hours.

The result shows that only the routes that service road level “C”, which represent Iowa roads with a service cycle time of 2.5 hours, are sensitive to spreading rate. The reason is the cycle time constraint makes a tighter bound on other road levels instead of the capacity constraint. The service speeds are 22 mph for urban area and 26 mph for rural area. So even a truck only service the rural area and has zero deadhead, in two hours it services only 52 lane

miles. However, at 300 lbs./lane mile spreading rate, a tandem axle truck, which has 16,000 pounds of capacity, can cover 53.3 lane miles. Therefore, routes that have a road cycle time of 1 hour (level "Metro"), 1.25 hours (level "A"), and 2 hours (level "B") will return to the depot with materials left.

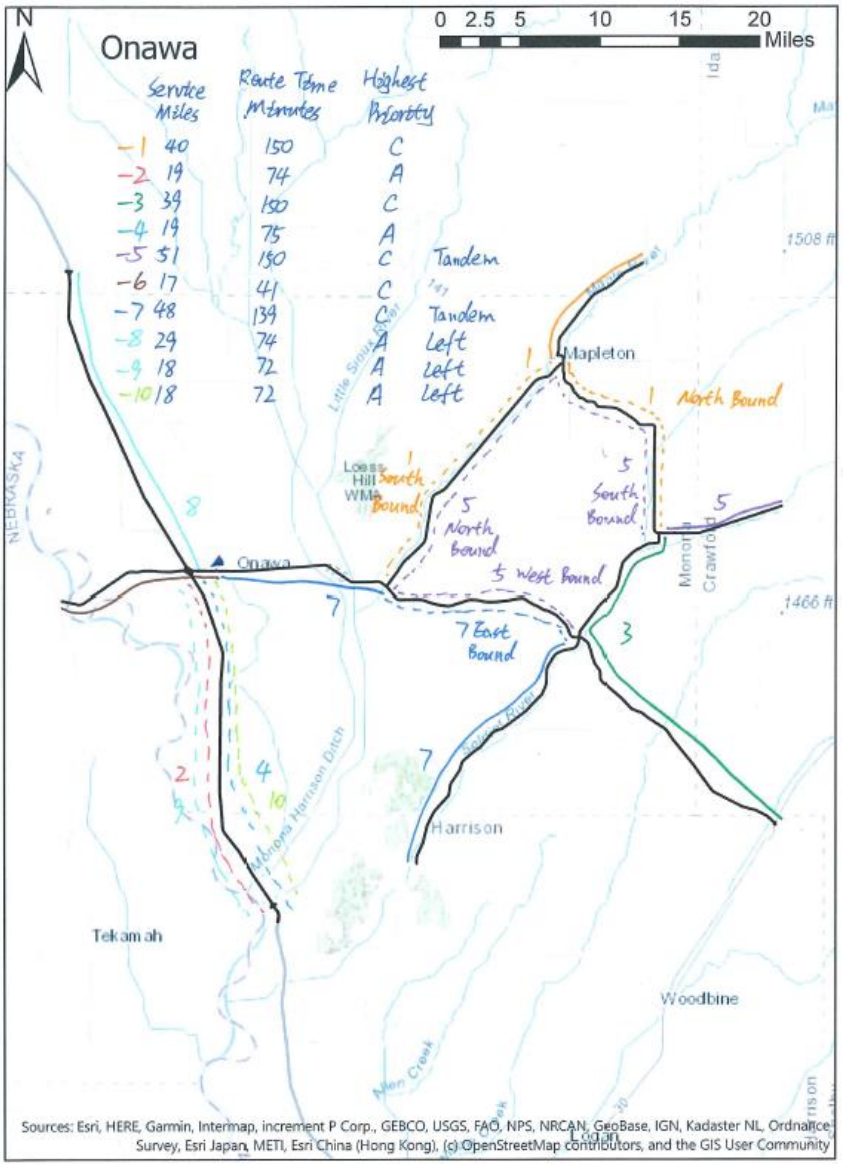

Figure 5.16 *Optimized Onawa routes*

Other depot networks have almost the same value regarding the total travel distance. Those results that various less than 1% of the travel distance might only be the different local

optimal found by the metaheuristic algorithm, thus not related to the different spreading rate used. It does not worth the truck drivers to follow different route plans for such insignificant changes.

Correctionville is the only network that is infeasible because of the capacity constraint in all current plans. Figure 5.17 shows the optimized result of spreading rate at 150 lbs./lane mile on the left and 300 lbs./lane mile on the right. The current plan is the same as the left graph, where route 3 and 4 both start from the depot. However, if the spreading rate is set as 300 lbs./lane mile, route 3 is no longer feasible. The total lane miles from the depot to Mapleton is about 58 miles. So at 300 lbs./lane mile this road needs 17,400 pounds of materials, which exceeds the capacity of a tandem truck.

Table 5.4 *Sensitivity analysis summary of travelling distance under different spreading rates*

Depot	150 lbs./ln mile	200 lbs./ln mile	250 lbs./ln mile	300 lbs./ln mile
Ashton	500.3	499.6	499.6	**499.6**
Carroll	205.4	205.4	205.4	**205.4**
Cherokee	213.6	212.0	213.6	**212.0**
Correctionville	237.9	237.9	237.9	**254.7**
Denison	253.1	253.1	252.3	**252.4**
Emmetsburg	169.5	169.5	169.5	**169.5**
Ida Grove	139.3	139.3	139.3	**139.3**
Le Mars	335.1	335.1	335.1	**335.1**
Onawa	475.7	475.7	475.7	**475.7**
Pocahontas	247.0	247.0	247.0	**259.6**
Rockwell City	276.2	276.2	276.2	**276.2**
Sac City	313.8	314.2	313.8	**313.8**
Sioux City Hamilton	332.5	332.6	332.4	**332.6**
Sioux City Leeds	251.3	251.1	251.1	**251.1**
Sloan	162.9	162.9	162.9	**162.9**
Spencer	207.0	207.0	207.0	**207.0**
Spirit Lake	223.8	223.7	223.7	**223.8**
Storm Lake	231.7	231.7	231.7	**231.7**

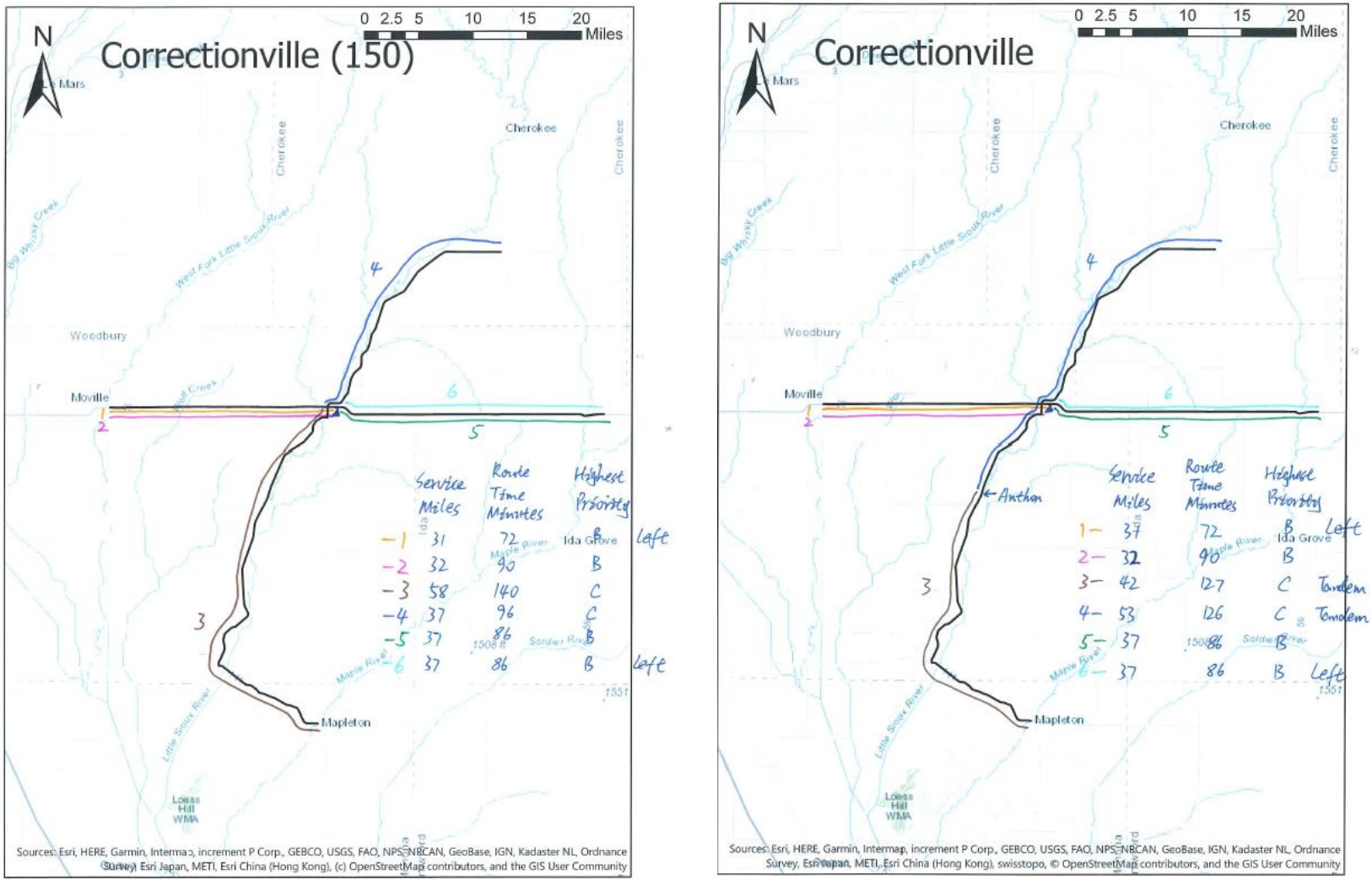

Figure 5.17 *Optimized Correctionville routes (spreading rate at 150 lbs./lane mile on the left, 300 lbs./lane mile on the right)*

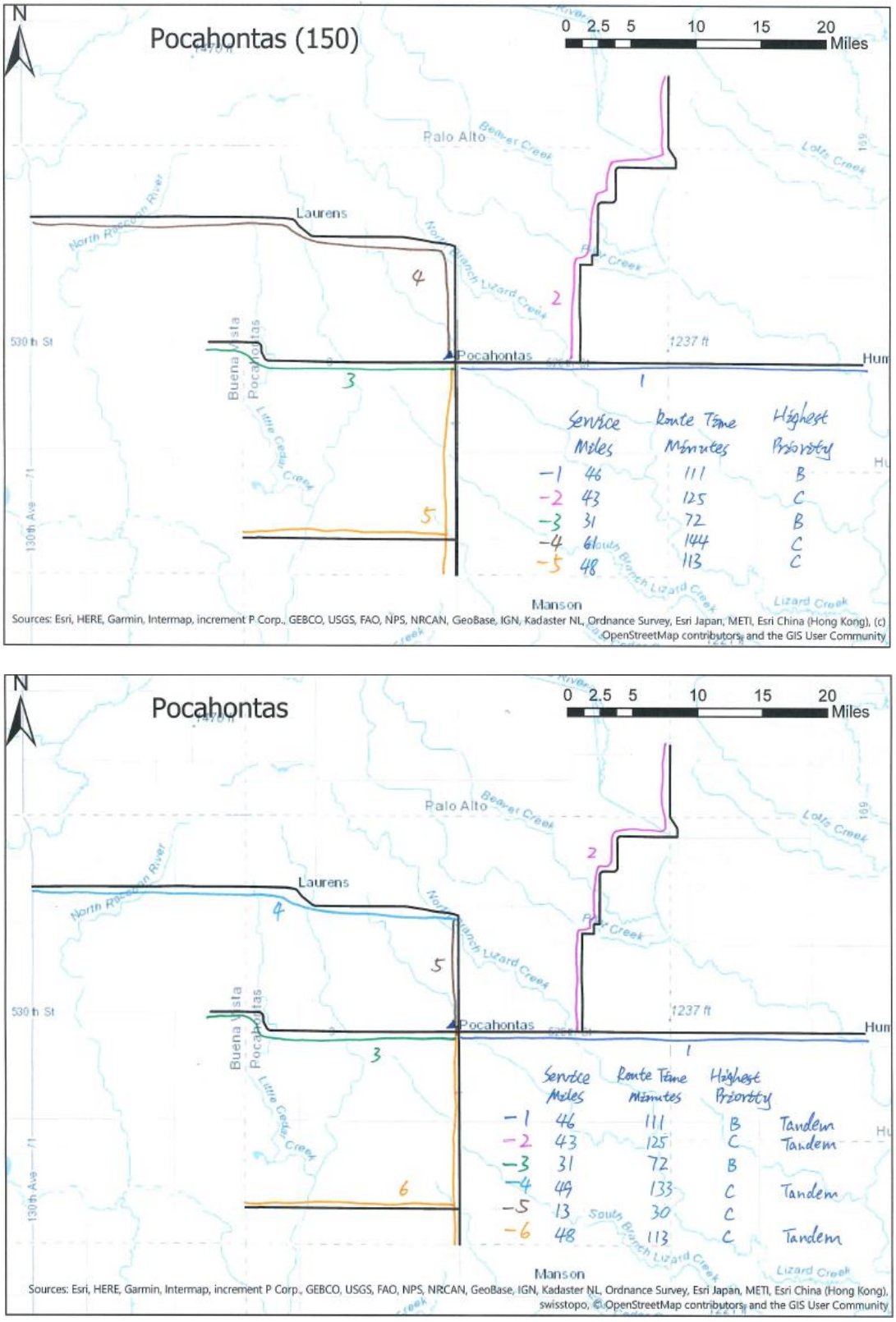

Figure 5.18 *Optimized Pocahontas routes (spreading rate at 150 lbs./lane mile at the top, 300 lbs./lane mile at the bottom)*

Pocahontas can save deadhead distance if a lower spreading rate is used. Figure 5.18 shows the optimized result of spreading rate at 150 lbs./lane mile at top and 300 lbs./lane mile at the bottom. Route 4 on the top graph of Figure 5.18 service 61 lane miles. At 300 lbs./lane mile this would cost 18,300 pounds of material. But at other lower spreading rates, this route can be serviced by only one truck.

5.4.3.2.3 Snow drifting

Practical concerns must consider site-specific situations. Hence the actual operation routes may not always follow the optimized routes in this study. The following issue concerns snow drifting. Figure 5.19 shows the Pocahontas current and optimized routes, separately. It is almost identical to Figure 5.18 (bottom graph) except route 5 and 6. Route 6 service the south of Pocahontas as a whole route, whereas in the current plan, the "red" route A32551 service Iowa 7 by itself. Although route 6 is feasible and can save deadhead for A32551, the A32551 is used here intentionally. Road segments on Iowa 7 have drifting snow issue, so a truck needs to be left there and service the roadways at a higher frequency.

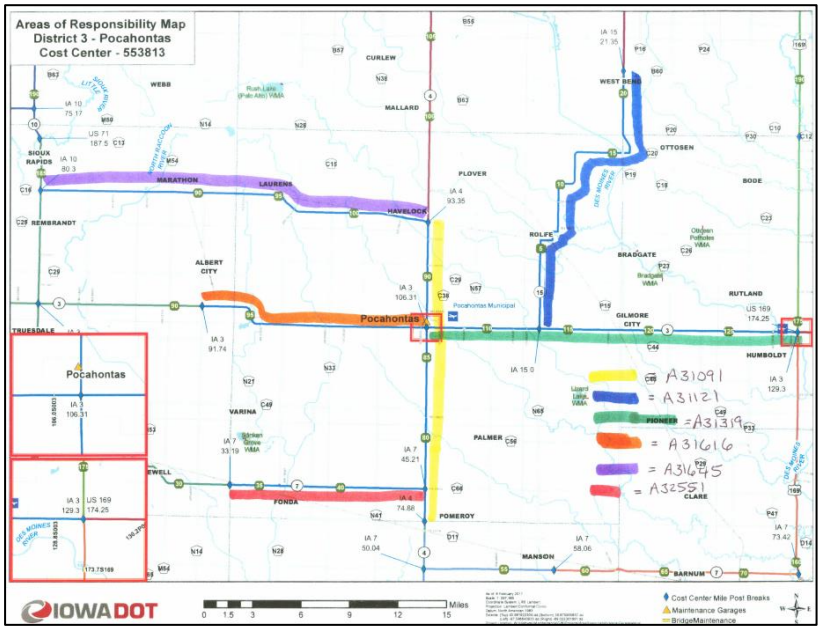

Figure 5.19 *Current Pocahontas routes*

5.4.3.3 Parallel computation

Table 5.5 represents the results of 10 runs of the single CPU on the 18 instances. The minimum, median, and maximum total fitness are represented with the standard deviation and relative standard deviation (RSD) of the ten runs. The RSD larger than 1% is colored red. It can be seen in the table that 11 instances have RSD less than 1%, which means every single run of the algorithm provides almost the same quality as the solution. This indicates the proposed algorithm is stable on these instances, and parallel computation might not have a significant improvement in these instances. Thus, only instances with RSD larger than 1% are tested for parallel computing.

Table 5.5 *Result of 10 Runs of Single CPU*

Depot	Min. Total Fitness	Median Total Fitness	Max. Total Fitness	Std. dev. of Total Fitness	RSD Total Fitness
Ashton	540.59	557.36	569.31	9.00	1.62%
Carroll	205.46	211.32	233.71	13.13	6.01%
Cherokee	212.01	213.58	213.58	0.50	0.23%
Correctionville	254.68	254.68	254.68	0.00	0.00%
Denison	252.38	253.11	253.11	0.23	0.09%
Emmetsburg	169.52	169.52	169.62	0.03	0.02%
Ida Grove	139.31	139.31	140.95	0.52	0.37%
Le Mars	335.13	337.19	338.64	1.34	0.40%
Onawa	489.09	518.08	525.70	12.38	2.41%
Pocahontas	259.55	259.55	259.55	0.00	0.00%
Rockwell City	276.19	276.19	276.24	0.02	0.01%
Sac City	313.83	314.79	314.79	0.38	0.12%
Sioux City-Hamilton	331.93	347.48	382.84	20.70	5.86%
Sioux City-Leeds	251.13	266.59	270.22	6.04	2.28%
Sloan	162.87	162.87	162.87	0.00	0.00%
Spencer	206.98	213.37	245.26	12.55	5.78%
Spirit Lake	223.76	226.18	240.91	4.95	2.18%
Storm Lake	231.65	231.65	231.89	0.07	0.03%

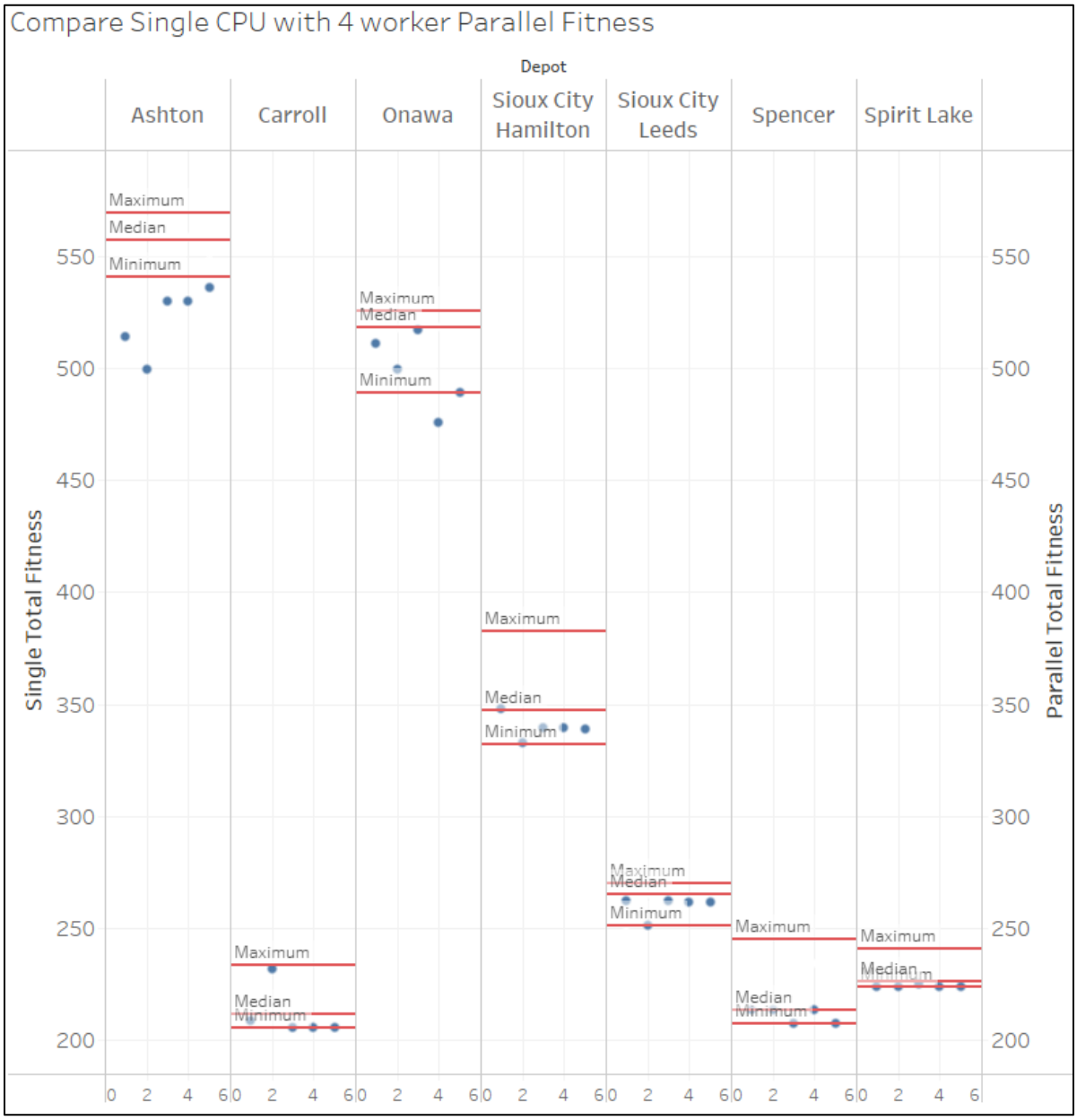

Figure 5.20 *Compare parallel computation result of total fitness value*

In this study, four worker CPU is used to test the parallel computation. Parallel computing was run five times. Figure 5.20 illustrates the result of comparing the total fitness value between the parallel and the single core CPU. Each blue dot represents each run of the parallel computation. Red lines in each depot column represent the minimum, median, and maximum values from the ten runs of the single CPU.

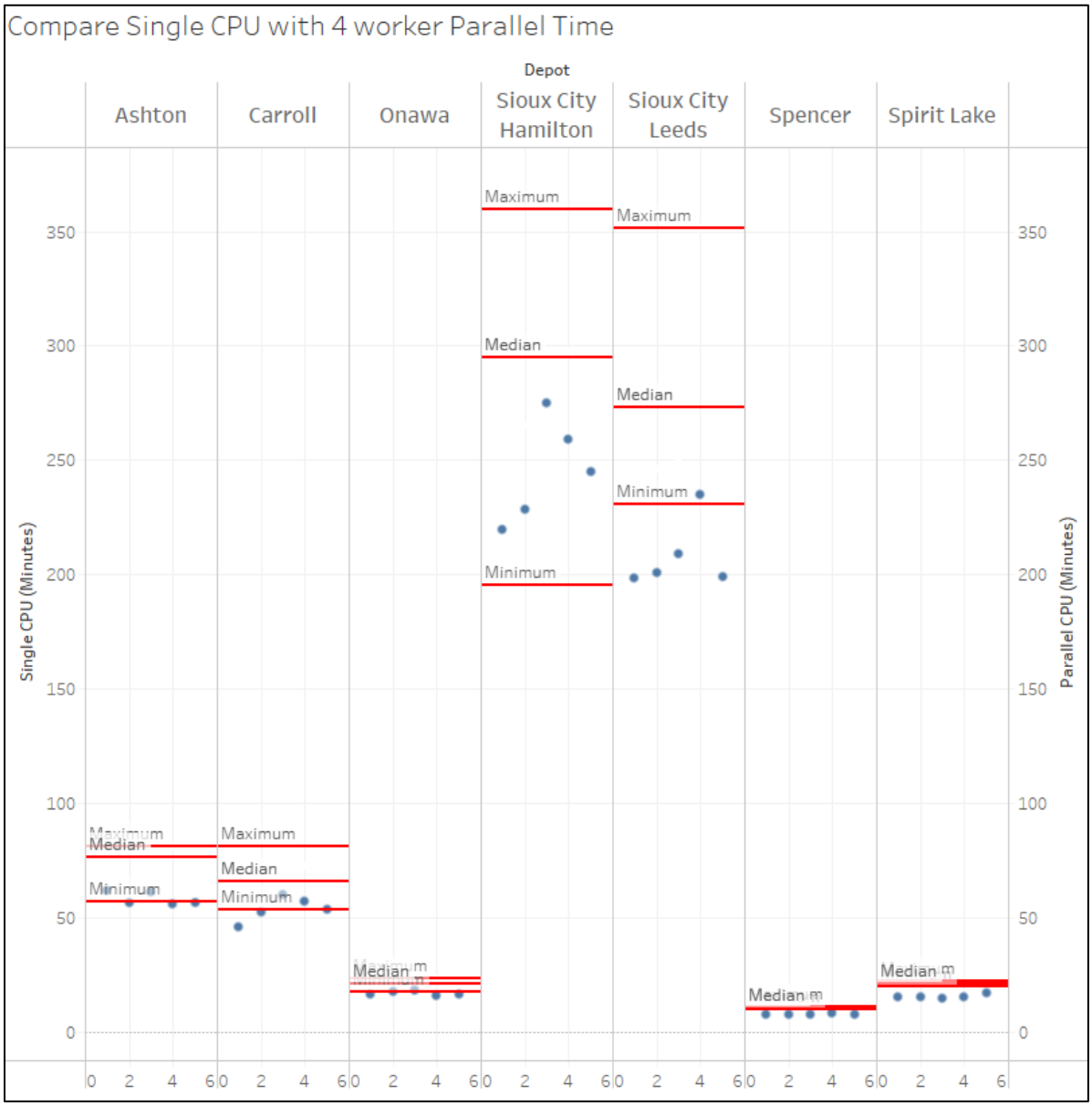

Figure 5.21 *Compare parallel computation result of computation time*

It can be seen that 2 of the 7 depots find a better solution by parallel computation. Eighteen of 35 times the parallel computation finds a solution better or equal to the best one that single CPU had found. Only one parallel computation did not find results less than or equal to the median. These results indicate the parallel computation enhanced solution quality.

Figure 5.21 represents the computation time comparison of the single versus parallel CPU. The stopping criterion in this study is by a fixed number of iterations. Hence, it is not expected to have a huge improvement in the computation time. Nevertheless, almost all parallel computation takes less time than the median time of the single CPU of each depot. Several parallels' computation time smaller than the single CPU. These results indicate the parallel computation enhanced computation efficiency.

CHAPTER 6. MULTI-DEPOTS WINTER ROAD MAINTENANCE ROUTING PROBLEM WITH INTERMEDIATE FACILITIES

6.1 Introduction

In this chapter, the problem of MDWRMRPIF is formulated and solved. The depot service boundaries among the multiple depots can be redesigned. Each route must start and end at its home depot, but they can reload at other depots or reload stations. Constraints introduced in Chapter 4 also hold in this chapter. Section 6.2 formulates the problem with a mathematical model. The MA discussed in Chapter 5 is adjusted to incorporate the multi-depots and reload feature. Section 6.3 introduces algorithms that need to be adjusted for multi-depot routing. Section 6.4 discusses the case study of District 3.

6.2 Mathematical Model

The definition of the coefficients and variables represents the same meaning as in section 5.2 unless otherwise stated below. The road segments traversed by a truck between facilities is called a *sub-route* in this chapter. The set of all sub-routes assigned to a truck is called a *rotation* the start and end node of which must be the truck's home depot. If a rotation only consists of one sub-route, which means the rotation does not use any reload, the rotation is simply a *route* under the definition of Chapter 5. The total travel time of each sub-route must be less than the minimum road service cycle time of any service road segment in this sub-route. The total travel time of the rotation should be less than the work span, which is the work shift duration of a truck driver. Now, let a node set $B \subset V$ represent the depot locations. Let m_1^l and m_2^l, where $l \in B$ represents the fleet size of the two types of vehicles of each depot, separately. Let t_R be the reload time. Let W be the work span limit for each truck driver's rotation.

Variables are defined as follows:

$$x_{ij}^{kp} = \begin{cases} 1 & if\ subroute\ k\ of\ rotation\ p\ service\ (i,j)\ from\ i\ to\ j \\ 0 & otherwise \end{cases}$$

$$y_{ij}^{kp} = \begin{cases} 1 & if\ subroute\ k\ of\ rotation\ p\ traverse\ (i,j)\ from\ i\ to\ j \\ 0 & otherwise \end{cases}$$

$$u_{ph} = \begin{cases} 1 & if\ the\ hth\ truck\ service\ rotation\ p \\ 0 & otherwise \end{cases}$$

The formulation is shown below:

$$\sum_{(i,j)\in A, k\in K, p\in P} c_{ij}(x_{ij}^{kp} + y_{ij}^{kp}) \tag{1}$$

$$\sum_{(i,j)\in A}(x_{ij}^{kp} + y_{ij}^{kp}) - \sum_{(j,i)\in A}(x_{ji}^{kp} + y_{ji}^{kp}) = 0 \qquad \forall k \in K, p \in P, i \in V \tag{2}$$

$$\sum_{(l,i)\in A, k\in K}(x_{li}^{kp} + y_{li}^{kp}) = 1 \qquad \forall p \in P, i \in V, l \in B \tag{3}$$

$$\sum_{(i,l)\in A, k\in K}(x_{il}^{kp} + y_{il}^{kp}) = 1 \qquad \forall p \in P, i \in V, l \in B \tag{4}$$

$$\sum_{\forall k\in K, p\in P} x_{ij}^{kp} = 1 \qquad \forall (i,j) \in A_R \tag{5}$$

$$\sum_{(i,j)\in A} q_{ij}\, x_{ij}^{kp} \le \sum_{h\in H_1\cup H_2} Q_h u_{ph} \qquad \forall k \in K, p \in P \tag{6}$$

$$\sum_{h\in H_1\cup H_2} u_{ph} = 1 \qquad \forall p \in P \tag{7}$$

$$\sum_{p\in P} u_{ph} \le 1 \qquad \forall h \in H_1 \cup H_2 \tag{8}$$

$$\sum_{p\in P, h\in H_1} u_{ph} \le \sum_{l\in B} m_1^l \tag{9}$$

$$\sum_{p\in P, h\in H_2} u_{ph} \le \sum_{l\in B} m_2^l \tag{10}$$

$$f_k \le f_{ij} x_{ij}^{kp} \qquad \forall k \in K, p \in P, (i,j) \in A_R \tag{11}$$

$$\sum_{(i,j)\in A}(x_{ij}^{kp} t'_{ij} + y_{ij}^{kp} t_{ij}) \le f_k \qquad \forall k \in K, p \in P \tag{12}$$

$$\left[\sum_{(i,j)\in A,k\in K}(x_{ij}^{kp}t'_{ij}+y_{ij}^{kp}t_{ij})\right]+(|K|-1)*t_R\le W \qquad \forall p\in P \qquad (13)$$

$$x_{ij}^{kp}r_{ij}=x_{i'j'}^{kp}r_{i'j'} \qquad \forall p\in P,(i,j)\in A_R,(i',j')\in A_R \qquad (14)$$

$$x_{ij}^{kp},y_{ij}^{kp},u_{ph}\in\{0,1\} \qquad \forall k\in K,p\in P,h\in H_1\cup H_2,\forall(i,j)\in A \qquad (15)$$

The objective is to minimize the total travel cost. Constraint (2) is the flow conservation equations for each sub-route. Constraint (3) and (4) state that all rotations must start and end at a depot and each rotation ends where it started. Constraint (5) ensures the network is fully serviced exactly once. Constraint (6) guarantees that the capacity of each vehicle is never exceeded. Constraint (7) states that each rotation is served by one vehicle. Constraint (8) states that each vehicle services one rotation at most. Therefore, some vehicle can be idling at the depot. Constraint (9) and (10) are the fleet size constraints. The trucks can be reassigned to other depots within the sector, but the total number of trucks are also limited by what is available in the sector. Constraint (11) states that the service cycle time of a sub-route is determined by the minimum service cycle time of any arc in the sub-route. Constraint (12) ensures each sub-route traveling time never exceeds sub-route cycle time. Constraint (13) guarantees that the rotation time is less than the work span. Constraint (14) guarantees that the arc plow direction (road-truck dependency) within a rotation is the same. Constraint (15) is the variable constraints.

6.3 Solution Algorithm

The framework of MA stays the same. Most of the functions employed in Chapter 5 can be applied for the problem in this chapter directly, including the crossover and the replacement operator. Some functions need slight changes. The local search and penalty cost evaluation should consider the moves around reload stations. Some other functions need

more endeavor. The PS algorithm initializing the solution is redesigned. The split procedure now must consider multi-depots and the reload station.

6.3.1 Initial Solution

Since the truck now can start from different depots, the PS algorithm should feature this characteristic. A new PS procedure is developed. It iterates among the multiple depots. First, the depot capacities (number of trucks that a depot can hold) are sorted in descending order. Then, each iteration performs the PS for all depots following that order. For each depot, the number of rotations should not exceed the current fleet size. The rotation built also satisfies all constraints and feature reload opportunities.

6.3.2 Split Procedure for Multiple Depots and Intermediate Facilities

The split procedure for multiple depots is shown below. For simplicity, only the capacity constraint is applied to illustrate the split procedure. Adding more constraints is similar to the split procedure provided in Chapter 5. The truck capacity is still $Q = 9$. Figure 6.1 illustrates an example of two depots and four tasks. Figure 6.2 indicates the auxiliary graph and the shortest path to complete the tasks. Numbers and arrows represent the same meaning as in the previous chapter. In Figure 6.2, the arrows and numbers on the upper part indicate the costs related to depot 1, whereas the lower part indicates the costs related to depot 2. The shortest path in Figure 6.2 is the optimal split for the task sequence. Arc c_1 is serviced by depot 1 as one route, and arc $c_2c_3c_4$ is serviced by depot 2 as the other route.

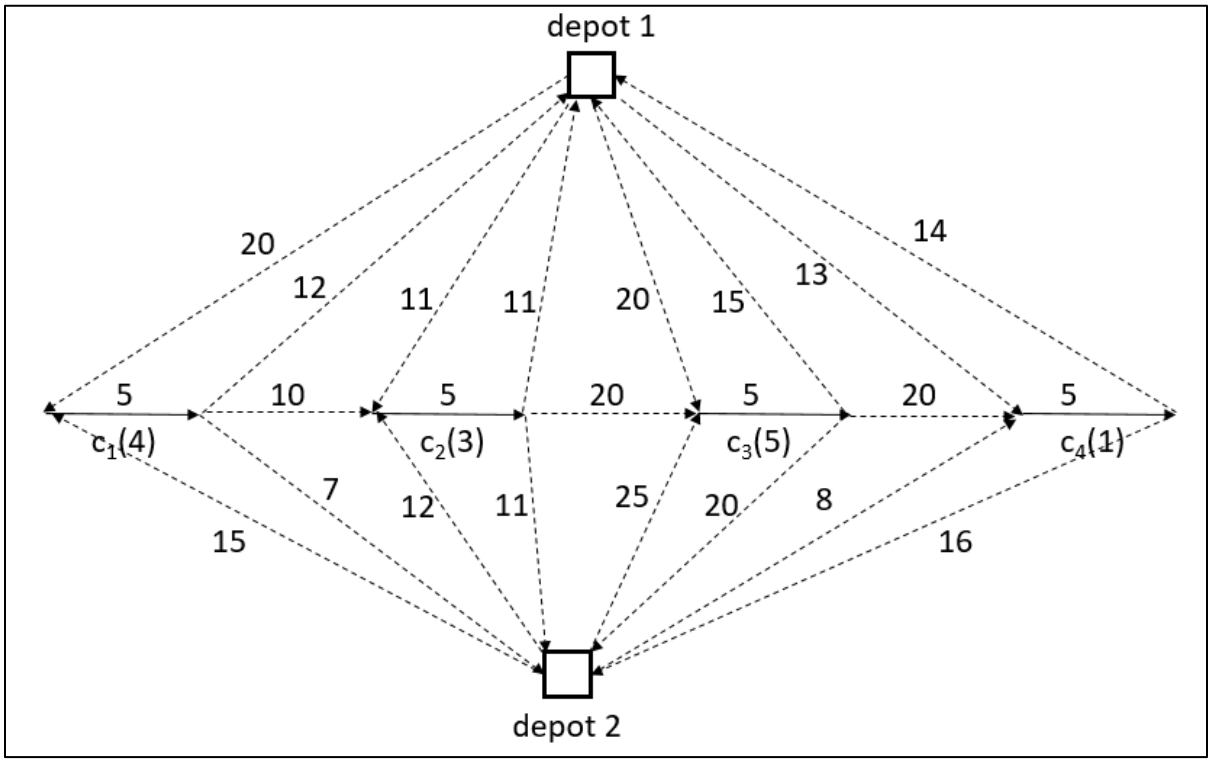

Figure 6.1 *Chromosome giant tour with four tasks and two depots*

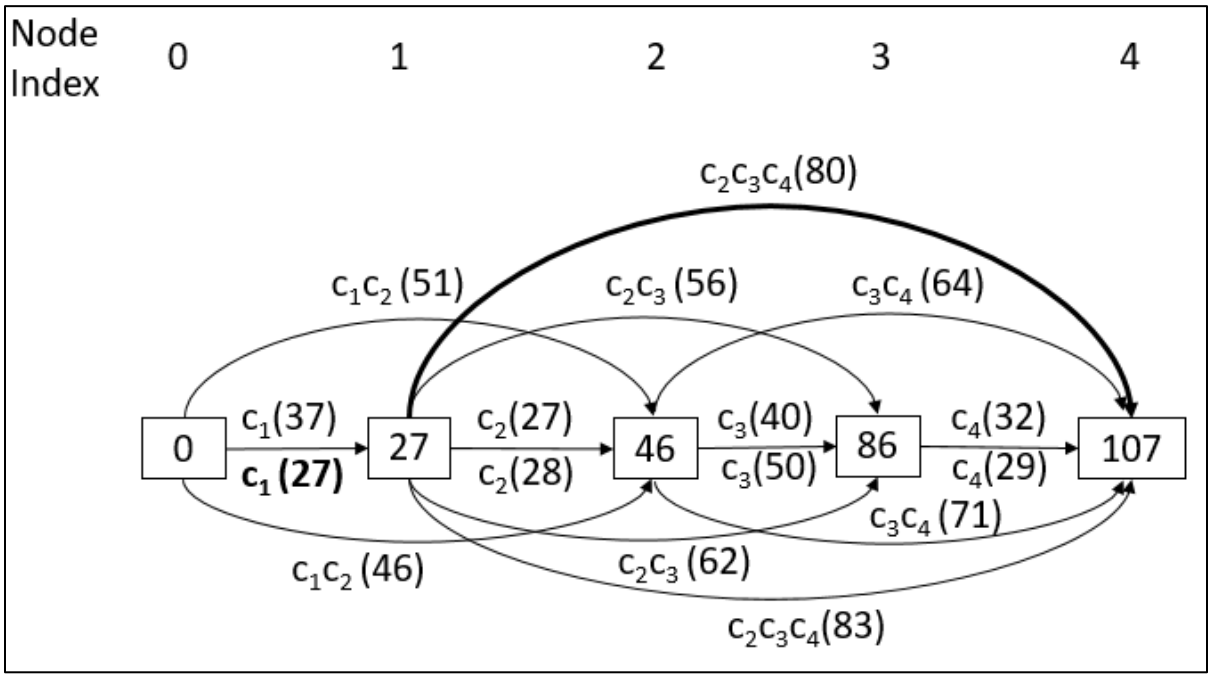

Figure 6.2 *Auxiliary graph and shortest path only considering the capacity constraint*

The split procedure for intermediate facilities is more complicated than multi-depots. Willemse et al. (2016b) showed that the optimal split of a giant tour for the CARPTIF has a time complexity of $\mathcal{O}(n^3)$, whereas the nearly optimal split can generate solutions at similar solution quality with time complexity of $\mathcal{O}(n^2)$. The average gap between the optimal and the heuristic split is less than 1% when the splitting is applied on chromosome instances generated by simple constructive heuristics. For a giant tour size of about 1000, the optimal split takes approximately 6 seconds, whereas the nearly optimal split only takes 0.06 seconds. Considering the sector instances, size is approximately 500-1000 arcs in this study. The optimal split is not computation affordable when this procedure is embedded in the MA procedure.

Similar to the SDWRMRP split procedure, the MDWRMRPIF split procedure uses the idea of building an auxiliary graph and finding the shortest path from the end node to the begin node. For each node of the auxiliary graph, a Pareto frontier remains while the algorithm builds the rotation from node $index\_i$ to $index\_j$. A PFN contains the information of the cost (travel distance), time, predecessor node index, predecessor PFN number, current PFN number, total single-axle truck used, total tandem-axle truck used, single-axle truck used per depot, tandem-axle truck used per depot, and the home depot node. For example, a PFN node with value $\{50,\ 30,\ 10,\ 6,\ 3,\ 2,\ 4,\ [1,\ 1],\ [0,\ 4],\ 8\}$ indicates that the cost of reaching the current node index using this rotation is 50, the time is 30. The predecessor of the rotation is the 6th PFN node in node $index\_i = 10$. The current PFN number is a sequential number that tracks the index of PFN nodes generated for node $index\_j$. The current node is the 3rd PFN that ever generated at $index\_j$. Two single-axle trucks and 4 tandem-axle trucks are used with the first depot using a single-axle truck and no tandem-axle

trucks, the second depot uses one single-axle truck and all four tandem-axle trucks. The source node of this rotation starts from node 8 in the network.

The heuristic split procedure that considers all constraints with multiple depots and intermediate facilities is developed in this study and shown in Figure 6.3. The structure of the algorithm uses many loops. Lines 5 and 11 loop through all nodes in the auxiliary graph indexed by $arcIndex\_i$ and $arcIndex\_j$. Line 17 loops through all depots as the rotation home depot $sourceNode$. Line 38 loops through all existing PFN of $arcIndex\_i$, and line 39 checks all vehicle types.

The PFN of the auxiliary graph remains in a 2-D vector $VCost$; the outer index indicates the auxiliary node index, and the inner index represents the current PFN number of the node. The algorithm uses vectors to keep track of the rotation cost and time that starts from a different depot source node. Line 9 reset these two vectors to 0 for all depot source nodes. Since servicing the next task from any depot leads to the same load increase, there is no need to use a vector for different depots. Lines 13 to 15 keep the lowest service cycle time of all tasks in the current rotation. Lines 19 to Line 35 calculate the cost and time of servicing the tasks from $arcIndex\_i$ to $arcIndex\_j$; this is similar to Lacomme et al. (2004). Then, line 36 checks if the current rotation time is less than the work span. If not, no PFN node will be updated. The loop for $arcIndex\_j$ will terminate and, the algorithm will iterate with the next value of $arcIndex\_i$. Line 40 finds the current truck capacity of the current vehicle type under concern.

```
1   Procedure SplitMDW(child)
2   arcMax = the size of the giant tour
3   PFN initialNode = {0, 0, 0, 0, 0, 0, 0, [0, 0, …], [0, 0, …], 0}
4   VCost[0][0]  = initialNode
5   for arcIndex_i = 0 to arcMax - 1
6       arcIndex_j = arcIndex_i
7       sourceNodeSize = # of depots
8       loadK = 0, LOSK = inf
9       set all costK[0 to sourceNodeSize - 1] = 0, timeK[0 to sourceNodeSize - 1] = 0
10      stopLoop = false
11      do  // next arcIndex_j
12          loadK = loadK + demand[arcIndex_j]
13          if LOSK > LOSMins[arcIndex_j]
14              LOSK = LOSMins[arcIndex_j]
15          endif
16          stopLoop = true
17          for sourceNodeIndex = 0 to sourceNodeSize – 1
18              sourceNode = sourceNodeMulti[sourceNodeIndex].
19              if arcIndex_j = arcIndex_i
20                  costK[sourceNodeIndex] += d_{sourceNode, arcIndex_i_begin}
21                  costK[sourceNodeIndex] += d_{arcIndex_i_end, sourceNode}
22                  costK[sourceNodeIndex] += distance[arcIndex_i]
23                  timeK[sourceNodeIndex] += t_{sourceNode, arcIndex_i_begin}
24                  timeK[sourceNodeIndex] += t_{arcIndex_i_end, sourceNode}
25                  timeK[sourceNodeIndex] += time[arcIndex_i]
26              else
27                  costK[sourceNodeIndex] -= d_{arcIndex_j-1_end, sourceNode}
28                  costK[sourceNodeIndex] += d_{arcIndex_j-1_end, arcIndex_j_begin}
29                  costK[sourceNodeIndex] += d_{arcIndex_j_end, sourceNode}
30                  costK[sourceNodeIndex] += distance[arcIndex_j]
31                  timeK[sourceNodeIndex] -= t_{arcIndex_j-1_end, sourceNode}
32                  timeK[sourceNodeIndex] += t_{arcIndex_j-1_end, arcIndex_j_begin}
33                  timeK[sourceNodeIndex] += t_{arcIndex_j_end, sourceNode}
34                  timeK[sourceNodeIndex] += time[arcIndex_j]
35              endif
36              if timeK[sourceNodeIndex] < workspan
37                  PFNSizeofArcIndex_i = size of VCost[arcIndex_i]
38                  for PFNIndexofArcIndex_i = 0 to PFNSizeofArcIndex_i – 1
39                      for vehType = (# of fleet type -1) to 0
```

Figure 6.3 *Heuristic split procedure for MDWRMRPIF*

```
40                          capacity = capacityHetero[vehType]
41                          subRouteTimeK = 0, subRouteLOSK = inf, reloadedCounts = 0
42                          deltaCostK = 0, deltaTimeK = 0
43                          if (loadK <= capacity) && (timeK[sourceNodeIndex] < LOSK)
44                              // do nothing
45                          else // reloaded
46                              reloaded()
47                          endif
48                          updatePFNnodes( )
49                      endfor  // vehicle type
50                  endfor  // Pareto Front Node of arcIndex_i
51              endif  // rotation time of current truck
52          endfor  // depot
53          arcIndex_j = arcIndex_j + 1
54      while (arcIndex_j < arcMax) && (stopLoop == false)  // arcIndex_j service the next arc
55  endfor  // arcIndex_i
```

Figure 6.3 (continued)

Lines 41 to 47 is the heuristic reload split for intermediate facilities visited during the rotation. First, line 41 checks whether the current rotation is less than capacity and cycle time constraint. If so, the rotation is simply a feasible route. Thus, nothing needs to be done. If not, either the load is more than capacity or the time of the route is more than the cycle time. Hence, a reload is needed within the rotation to divide the rotation into several sub-routes. In this study, the heuristic split assumes the reload is performed only when the capacity is violated. As stated above, this split generates a nearly optimal solution in an affordable time.

The reload procedure is represented in Figure 6.4. The algorithm determines the index of the arc after which a reload is needed by a loop through all the arcs under the current $arcIndex\_i$ to $arcIndex\_j$. The index of this arc is called the $cutoffLoadArcIndex$. A rotation might have multiple reloads, and each of the services between reloads/depot is called a $subRoute$ in the algorithm. Cost and time changes to the rotation are updated by $deltaCostK$, and $deltaTimeK$.

```
1   Procedure reloaded( )
2   cutoffLoad = 0
3   for cutoffLoadArcIndex = arcIndex_i to arcIndex_j
4       sourceNode = sourceNodeMulti[sourceNodeIndex].
5       if cutoffLoadArcIndex = arcIndex_i
6           subRouteTimeK = t_{sourceNode, cutoffLoadIndex_{begin}}
7           subRouteTimeK += t_{cutoffLoadIndex_{end}, sourceNode}
8           subRouteTimeK += time[cutoffLoadIndex]
9       else
10          subRouteTimeK -= t_{cutoffLoadIndex-1_{end}, sourceNode}
11          subRouteTimeK+= t_{cutoffLoadIndex-1_{end}, cutoffLoadIndex_{begin}}
12          subRouteTimeK += t_{cutoffLoadIndex_{end}, sourceNode}
13          subRouteTimeK += time[cutoffLoadIndex]
14      endif
15      tempSubRouteLOSK = subRouteLOSK
16      if (tempSubRouteLOSK) > LOSMins[cutoffLoadArcIndex])
17          tempSubRouteLOSK = LOSMins[cutoffLoadArcIndex]
18      endif
19      // if reload is needed after service this cutoff arc
20      if ((cutoffLoad + demand[cutoffLoadArcIndex] > capacity) || (subRouteTimeK > tempSubRouteLOSK))
21          tempRouteTimeK = subRouteTimeK
22          if (cutoffLoadArcIndex == arcIndex_i)  // the first arc in the route is too far to service
23              deltaCostK = 0, deltaTimeK = 0
24              subRouteLOSK = LOSMins[arcIndex]
25              break
26          else  // reload between two arcs
27              reloadNode = IFUsage[cutoffLoadIndex − 1_{end}][ cutoffLoadIndex_{begin}]
28              deltaCostK = - d_{cutoffLoadIndex-1_{end}, cutoffLoadIndex_{begin}}
29              deltaCostK += IFDist_{cutoffLoadIndex-1_{end}, cutoffLoadIndex_{begin}}
30              deltaTimeK = - t_{cutoffLoadIndex-1_{end}, cutoffLoadIndex_{begin}}
31              deltaTimeK += IFTime_{cutoffLoadIndex-1_{end}, cutoffLoadIndex_{begin}}
32              tempRouteTimeK -=  t_{cutoffLoadIndex-1_{end}, cutoffLoadIndex_{begin}}
33              tempRouteTimeK -=  t_{cutoffLoadIndex_{end}, sourceNode}
34              tempRouteTimeK -= time[cutoffLoadIndex]
35              tempRouteTimeK +=  t_{cutoffLoadIndex-1_{end}, reloadNode}
36          endif
37          // after reload the truck, the subRoute is refreshed
38          if (tempRouteTimeK < subRouteLOSK)
```

Figure 6.4 *Update reload procedure*

```
39              subRouteTimeK = t_{reloadNode, cutoffLoadIndex_begin}
40              subRouteTimeK += time[cutoffLoadIndex]
41              subRouteTimeK += t_{cutoffLoadIndex_end, sourceNode}
42              subRouteLOSK = LOSMins[cutoffLoadArcIndex]
43              cutoffLoad = demand[cutoffLoadArcIndex]
44              costK[sourceNodeIndex] += deltaCostK
45              timeK[sourceNodeIndex] += deltaTimeK
46              reloadedCounts += 1
47          endif
48      else  // reload is not needed after service this cutoff arc
49          if (SubRouteLOSK) > LOSMins[cutoffLoadArcIndex])
50              SubRouteLOSK = LOSMins[cutoffLoadArcIndex]
51          endif
52          cutoffLoad += demand[cutoffLoadArcIndex]
53      endif
54  endfor
```

Figure 6.4 (continued)

The reload time and distance that deadhead between any two arcs can be pre-computed and are fixed numbers, which are stored in $IFDist$, and $IFTime$ matrix, separately. Also, the nearest intermediate facility for reloading between and two arcs can be found from the $IFDist$ matrix; thus, the best reload station node to visit between any two arcs is stored in the $IFUsage$ matrix.

Lines 5 to 18 update the time and cycle limit of the sub-route. Line 20 determines whenever a reload is needed by testing the capacity and cycle time constraint. If so, line 21 to 47 update the reload to the sub-route and the rotation correspondingly. If not, line 49 to 52 simply update the load and cycle time limit of the current sub-route.

In line 22, the algorithm first eliminates the situation that the first service arc of the rotation is too far from the current depot. The reload procedure breaks, and the split procedure moves on to the next home depot. Hence, lines 22 to 25 prevent the rotation using a non-promising depot as its home depot. Lines 27 to 35 update the $deltaCostK$,

$deltaTimeK$ and the travel time of the new sub-route. Note that after reload, the sub-route travel time might be changed, so line 38 ensures the current sub-route time still less than the sub-route cycle time limit. If so, the sub-route is refreshed by resetting the travel time and cycle time limit. The changes to rotation are applied, and the reload counts increase by 1.

Figure 6.5 shows the details of updating the PFN procedure. The procedure (at line 2) first checks if the sub-route satisfies all constraints. Since the heuristic split is performed only for a capacity violation, the new sub-route time $subRouteTimeK$ might not necessarily less than the most frequent cycle time of tasks in the sub-route. Also, a reload visit usually means the rotation needs a longer deadhead time from/to the reload station between sub-routes. Hence, the work span should be checked again. A 15-minute reload time is assumed in this study.

Lines 3 to 8 in Figure 6.5 guarantee that the number of trucks in use is less than the available fleet size from all depots. An alternative way is to restrict the number of trucks in use for each depot and each truck type, but these will lead to high computation time and memory usage. Line 10 represents that if all the constraints are met, the loop for $arcIndex\_j$ can proceed on the next $arcIndex\_j = arcIndex\_j + 1$. Lines 11 to 34 update the information for the current PFN. Line 35 uses the dominate rule (Prins et al., 2014) to update the Pareto frontier nodes for auxiliary node $Index\_j$.

The dominate rule in this study only compares the cost, the total single-axle truck in use, and the total tandem-axle truck in use. By doing so, the number of PFN is much less than if comparing the fleet size of each depot. Thus, the computation time and memory usage are much less. The split solution can be extracted by using the predecessor's node and number.

```
1   Procedure updateMDWPFNnodes( )
2   if (subRouteTimeK < subRouteLOSK) && (timeK[sourceNodeIndex] < workspan - 15* reloadedCounts)
3       if ( ( (vehtype = 0) && (VCost[arcIndex_i][PFNIndexofArcIndex_i].totalSGL < fleetSize[vehtype]) )
4       || ( (vehtype = 1) && (VCost[arcIndex_i][PFNIndexofArcIndex_i].totalTDM <  fleetSize[vehtype]) ) )
5           // fleet size satisfied, do nothing
6       else
7           continue
8       endif
9       // update current PFN
10      stopLoop = false
11      currentNode.Cost = VCost[arcIndex_i][PFNIndexofArcIndex_i].Cost + costK[sourceNodeIndex]
12      currentNode.Time = VCost[arcIndex_i][PFNIndexofArcIndex_i].Time + timeK[sourceNodeIndex]
13      currentNode.Time += 15* reloadedCounts
14      currentNode.SGL = VCost[arcIndex_i][PFNIndexofArcIndex_i].SGL
15      currentNode.TDM = VCost[arcIndex_i][PFNIndexofArcIndex_i].TDM
16      currentNode.totalSGL = VCost[arcIndex_i][PFNIndexofArcIndex_i].totalSGL
17      currentNode.totalTDM = VCost[arcIndex_i][PFNIndexofArcIndex_i].totalTDM
18      currentNode.PredecessorNodexIndex = arcIndex_i
19      currentNode.PredecessorPFNNumber = VCost[arcIndex_i][PFNIndexofArcIndex_i].currentPFNNumber
20      currentNode.depotNode = sourceNode
21      if vehType = 0
22          currentNode.SGL[sourceNodeIndex]  += 1
23          currentNode.TotalSGL +=1
24      else
25          currentNode.TDM[sourceNodeIndex]  += 1
26          currentNode.TotalTDM +=1
27      endif
28      // track the last PFN node of arcIndex
29      PFNSizeArcIndex_j = size of VCost[arcIndex_j + 1]
30      if PFNSizeArcIndex_j != 0
31          currentNode.currentPFNNumber = VCost[arcIndex_j + 1][PFNSizeArcIndex - 1].currentPFNNumber + 1
32      else
33          currentNode.currentPFNNumber = 0
34      endif
35      update the dominates nodes of node arcIndex_j in the auxiliary graph
36  endif
```

Figure 6.5 *Update PFN procedure for MDWRMRPIF*

6.4 Computational Experiments

6.4.1 Computation Results

The traffic network of District 3 is used as the test instances in this study. The depot sector boundary is shown in Figure 3.2. The reload time t_R is set to be 15 minutes, and the work span is 8 hours.

The MA algorithm was run for each sector. Algorithm parameters are the same as in section 5.4.1. Table 6.1 summarizes the total distance and fleet size of the optimized routes for each sector. The third column sums up the optimized distance for each depot in the SDWRMRP scenario in miles. The fourth column shows the optimized distance while solving the MDWRMRPIF scenario for the sector. The fifth column represents the optimized fleet size for the SDWRMRP scenario, while the last column represents the optimized fleet size for the MDWRMRPIF scenario. Rock Valley performs like a depot in the Alton and Rock Rapids network. Hence, the Alton and Rock Rapids are solved as multiple depot problem, and their result is put under the SDWRMRP scenario.

It can be observed from Table 6.1 the optimized distance of MDWRMRPIF is less than the sum up the value of the SDWRMRP. Some depots demand a larger fleet size while some others need less. The reason is the road segment service responsibility for these depots has changed, and the routes are optimized under new circumstances.

The total optimized travel distance of all sectors under the SDWRMRP scenario is 4919 miles. The total optimized travel distance of all sectors under MDWRMRPIF scenario is 4859.8 miles. Therefore, the deadhead distance saving of MDWRMRPIF compared to SDWRMRP is 1.2%.

Table 6.1 *Total optimized distance of single depots within each sector, and optimized sector distance in miles, current and optimized fleet sizes*

Sector	Garage Name	Total Optimized Single Depot Distance	Optimized Sector Distance	Optimized Single Depot Fleet Size	Optimized Sector Fleet Size
Ashton	Ashton	717.2	712.2	10	10
	Rock Rapids			4	4
	Rock Valley (for Rock Rapids)			1	2
Le Mars	Alton	1072.7	1061.1	9	10
	Rock Valley (for Alton)			1	0
	Correctionville			6	6
	Le Mars			7	7
Onawa	Denison	1030.3	1016.3	6	7
	Ida Grove			3	4
	Onawa			10	9
	Sloan			5	5
Sac City	Carroll	1054.9	1053.5	4	4
	Pocahontas			6	7
	Rockwell City			6	6
	Sac City			7	6
Storm Lake	Cherokee	1043.9	1016.7	6	6
	Emmetsburg			4	3
	Spencer			5	6
	Spirit Lake			5	4
	Storm Lake			5	5

As an example, Figure 6.6 illustrates the current responsibility of the Onawa sector and color-coded by depots. Figure 6.7 illustrates the optimized routes for Onawa sector. Comparing these two graphs, the responsibility of route 4 and 11 have changed. It is found that route 11 is closer to Denison instead of Onawa. Optimized routes for other parts of the new Onawa network are changed accordingly. Other routes are the same as in the SDWRMRP scenario. This example shows that the algorithm can find better sector partitions comparing to the current route plans.

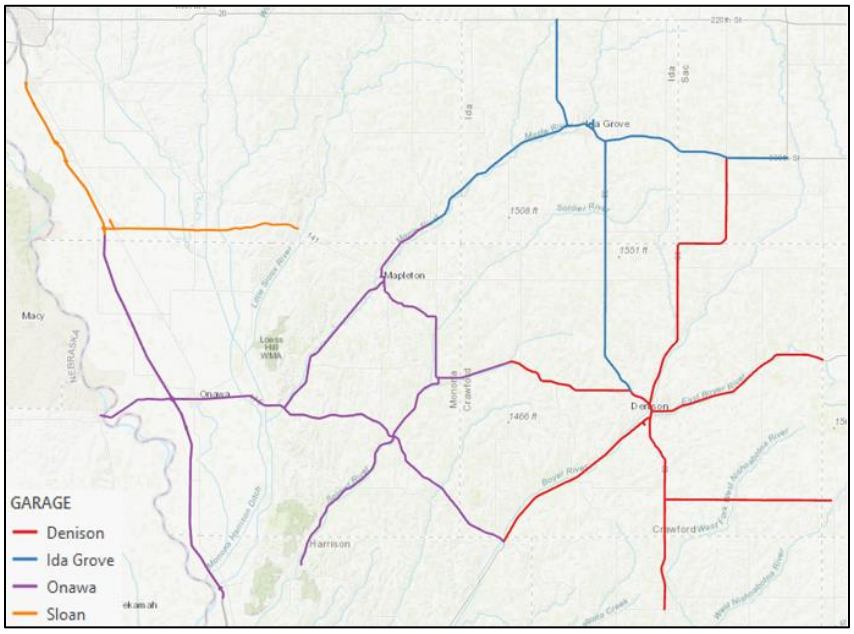

Figure 6.6 *Current Onawa sector, color-coded by depot responsibility*

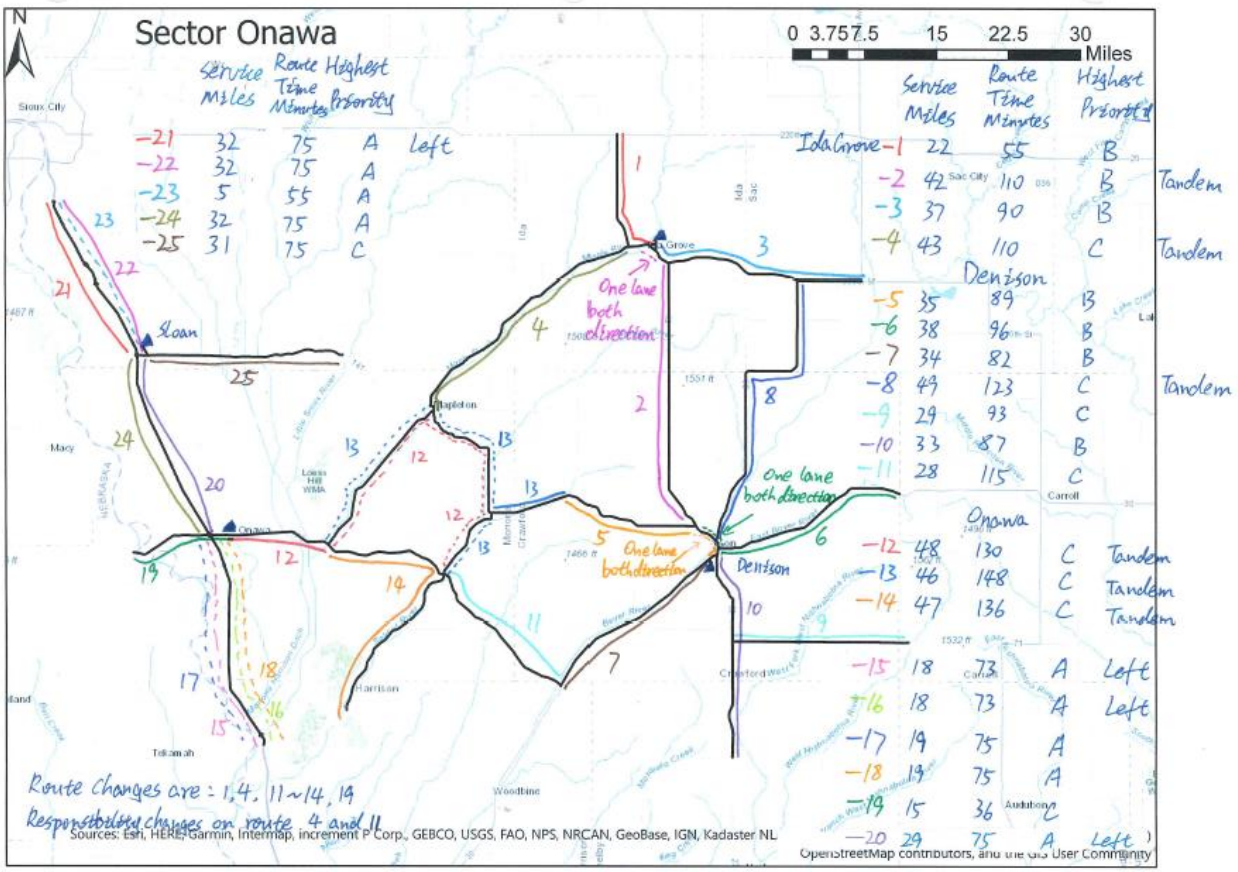

Figure 6.7 *Optimized routes for Onawa sector*

The reason MDWRMRPIF does not make a significant improvement to the SDWRMRP in Table 6.1 could be due to the network structure of the instance in this study. Figure 6.8 illustrates the situation when a reload is needed. The triangle represents a depot with reload capability; the trapezoid represents a reload station. Each arc has a specific demand for material and lets the length of the arc be the same for simplicity. Assume a truck capacity of 30.

Figure 6.8-1b illustrates the first situation when a reload is needed. Figure 6.8-1a only needs one route, since the truck can service both the arcs in one run. Figure 6.8-1b is when reload is needed and can reduce deadhead. The arcs now have a demand of 20. Starting from either depot, the truck will not have enough capacity to service both arcs if not reloaded. Hence, the total travel distance is 80 if no reload is used, and two truck routes are needed. Alternatively, if reload is used after servicing the first arc in the truck route, the capacity replenished, then the truck can service the other arc without any deadheading. The total travel distance is 40 with only one truck route.

In essence, to make the reload opportunity helpful to the truck route, at least one arc in the route must has a demand more than half of the truck capacity; or regarding the cycle time constraint, at least one arc in the route must have a service time more than half of the cycle time of this route. However, the current network structure does not meet the situation above. The traffic network exists; plenty of intersections, ramps, and U-turn locations exist. So, the road segment arc in the traffic network is never long enough to need more demand than half-truck capacity.

Figure 6.8-2a illustrates the second situation when reload is needed. Now, there are four arcs in the graph, and an intersection is in the middle of the graph. The truck route can replenish at the reload station and finish all service in one run without any deadhead.

Figure 6.8-2b illustrates when making the reload station as a depot, hence, a multiple depot problem, the reload is not needed. The graph also can be serviced by two routes without any deadhead.

The current network structure and depot location do not have a long haul between any depots. This is reasonable upon second thought. Since the current operation needs to make the SDWRMRP work for any single depot, the whole district network was partitioned into individual depots under the scenario where no reload is assumed. Hence, the depots are placed near the center in their responsibility graph and without any road segment too far to be serviced.

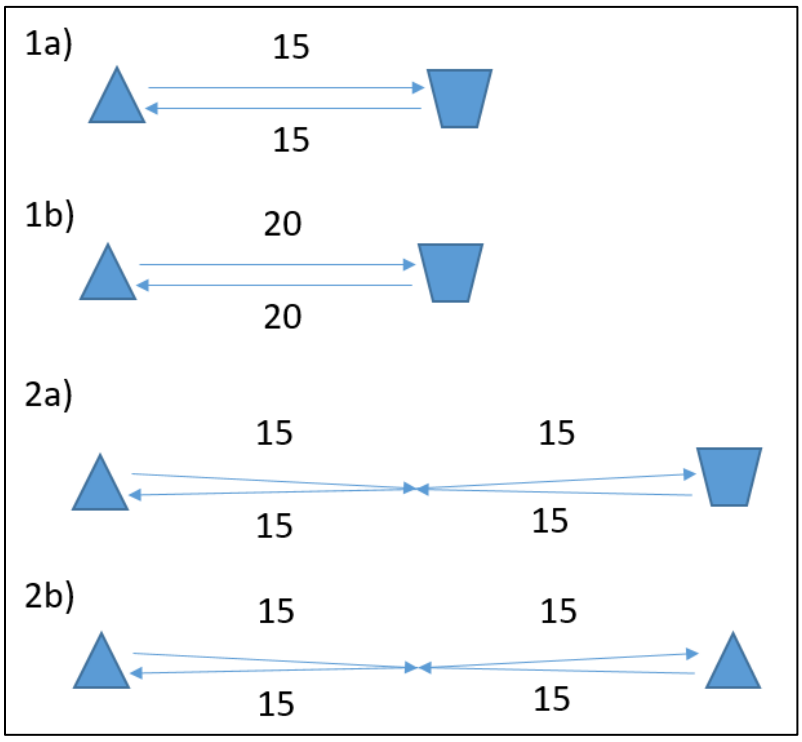

Figure 6.8 *Reload situation*

Given the discussion above, the improvement of the MDWRMRPIF comparing to SDWRMRP is due to the reassignment of road segment responsibility, but not due to the reload opportunity.

6.4.2 Case Study

To demonstrate that the reload opportunity could benefit the result and the effectiveness of the algorithm, an instance network is created and tested. The instance is made by using the Ashton sector network, and assuming the depot at Rock Valley can only perform as a reload station, but not as a depot. Figure 6.9 shows the Ashton sector color-coded by the current service responsibility by depots. The black triangles represent home depots, and the black trapezoid represents the reload station.

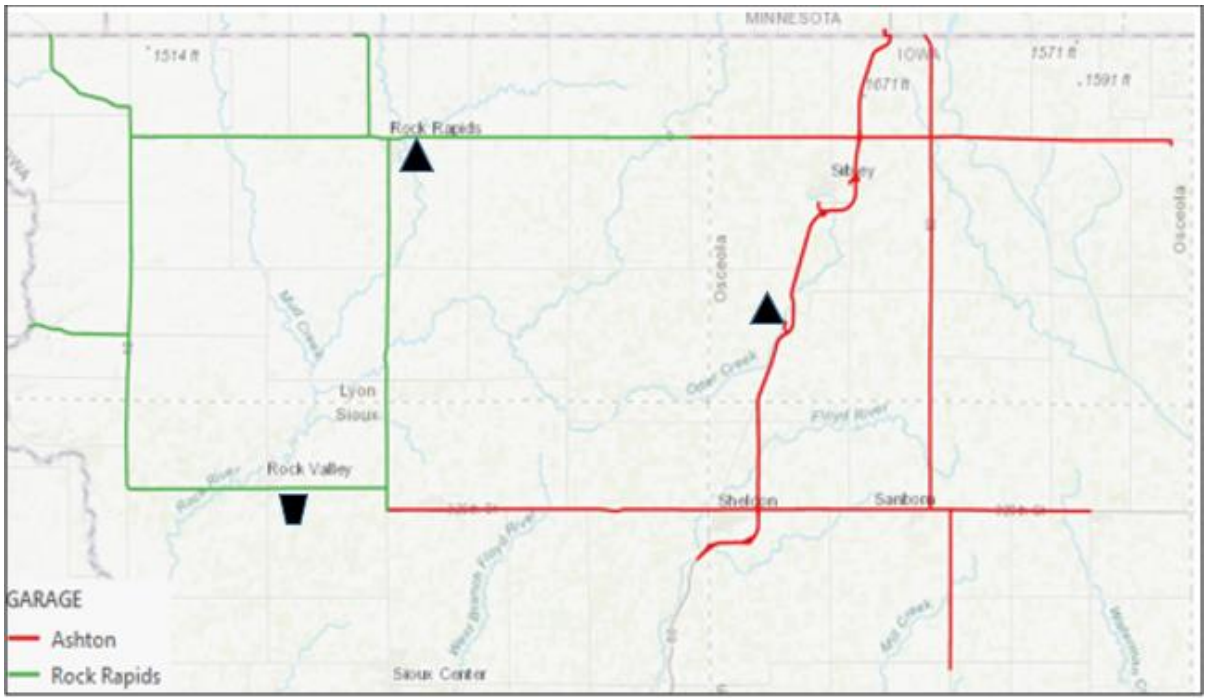

Figure 6.9 *Sector Ashton test instance—current responsibility*

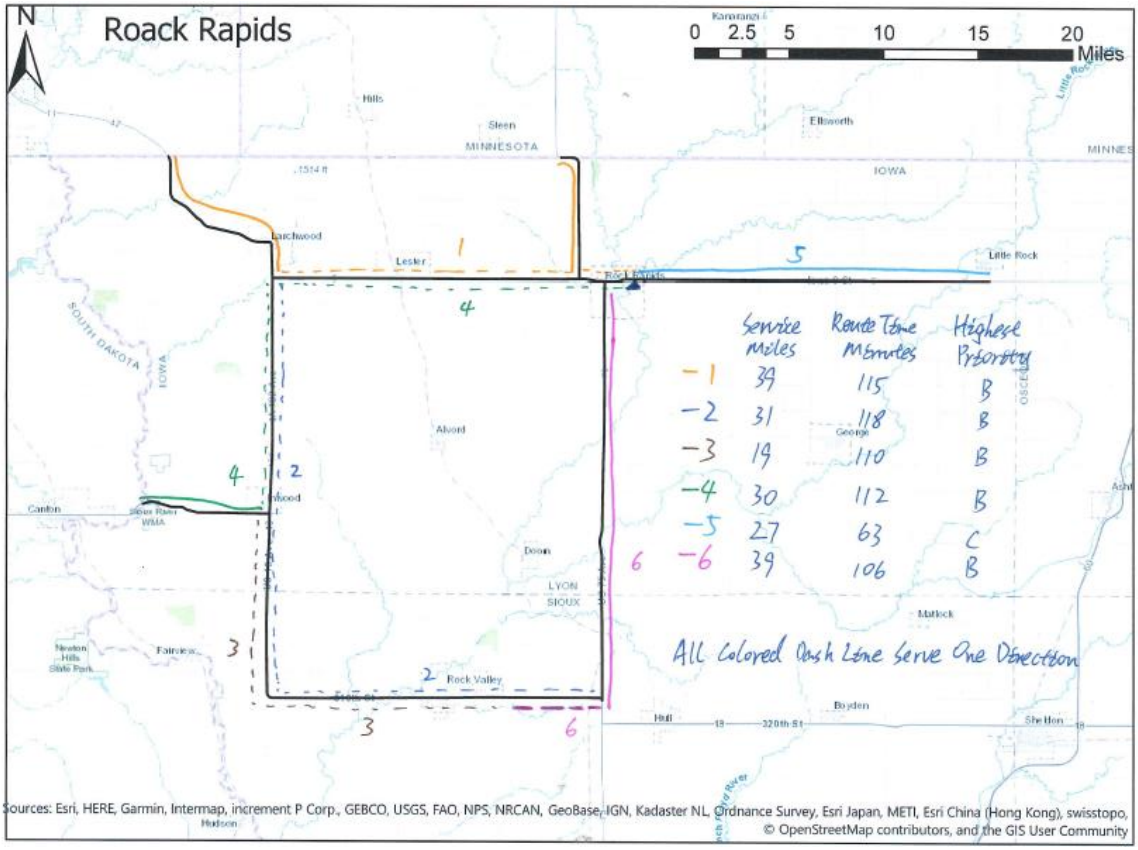

Figure 6.10 *Rock Rapids routes under SDWRMRP scenario*

In the SDWRMRP scenario, the Rock Rapids does not have a reload station at Rock Valley. Figures 6.10 and Figure 6.11 show the routes for Rock Rapids and Ashton under the SDWRMRP scenario, separately. The travel distance of Rock Rapids is 307.3 miles, and the total work completion time is 624 minutes. Six trucks are used in Rock Rapids. The travel distance of Ashton is 499.6 miles, and the total work completion time is 1006 minutes. Ten trucks are used in Ashton. Hence, the two networks have a combined travel distance of 806.9 miles and operation time of 1630 minutes.

In the MDWRMRPIF scenario, the sector is solved as a whole network with two depots and one reload station. Figure 6.12 shows the optimized routes. First, route 11 and 13 both reload at the Rock Valley. Route 11a services southbound from Rock Rapids to Rock Valley, and after a reload, route 11b services the northbound of Iowa 182 and returns to its

home depot. Route 13a and 13b service in the opposite direction. Any two of these routes are too long to be serviced as a whole because of the time constraint. Also, route 13b with route 8 changes the service responsibility between Ashton and Rock Rapids, which can decrease the deadhead distance by about 5 miles.

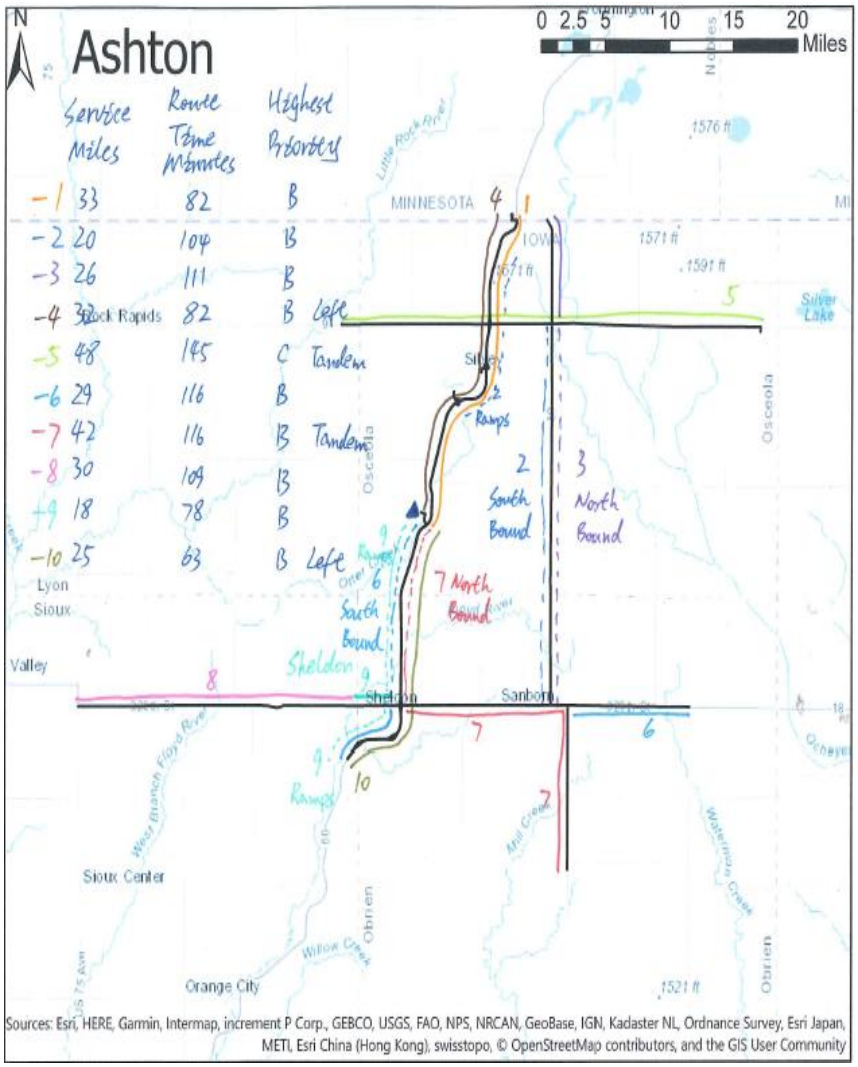

Figure 6.11 *Ashton routes under SDWRMRP scenario*

The total traveling distance of the Ashton sector is 723.2 miles with an operation time of 1540 minutes (2 reloads take an additional 30 minutes). Compared to the SDWRMRP, the MDWRMRPIF saved 10.4% of the travel distance and 5.5% of the operation time.

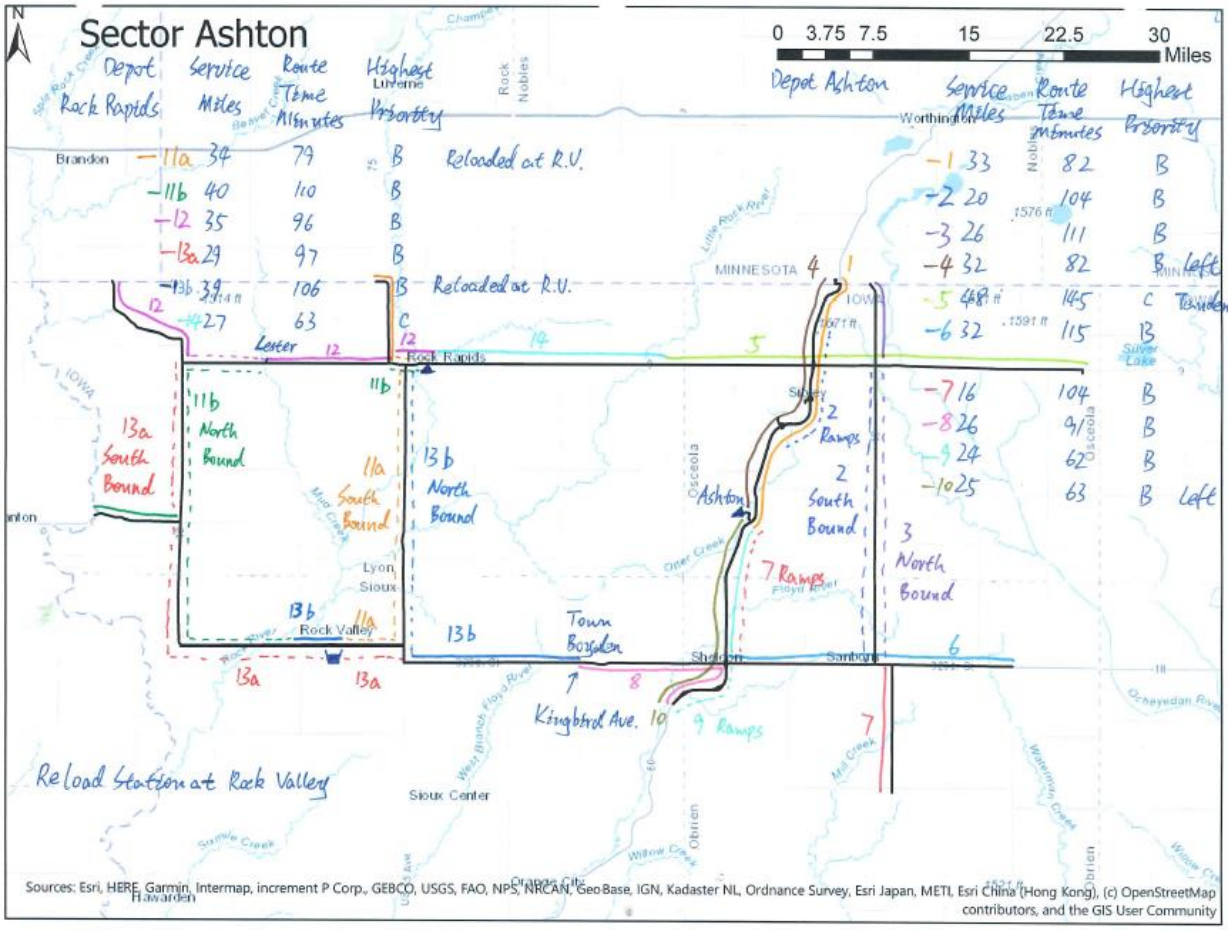

Figure 6.12 *Sector Ashton with Rock Valley as a reload station—current responsibility*

CHAPTER 7. CONCLUSION

7.1 Summary

This study developed methods to design the routes for winter road maintenance operations. Two types of winter road maintenance routing problems are studied, considering practical constraints. The proposed solution algorithms are applied to real-world networks. The result shows the proposed method can reduce deadhead distance.

The key findings are as follows. First, the SDWRMRP incorporated real-world winter road maintenance constraints, including road segment service cycle time, heterogeneous vehicle capacity, fleet size, and road-vehicle dependency. The problem is solved by the MA approach. This is the first study that formulated SDWRMRP and developed the route split procedure within the MA framework to solve SDWRMRP. In addition, a parallel metaheuristic algorithm is proposed to enhance the solution quality and computation efficiency. The result shows the optimized route reduced deadhead distance by 13.2% compared to the current practice. The deadhead saving percentage could be even larger, given that optimized routes strictly satisfy all constraints, whereas the current operation might not.

Second, the MDWRMRPIF considers work duration constraint, in addition to road segment service cycle time, heterogeneous vehicle capacity, fleet size, and road-vehicle dependency. This is the first study that formulate the MDWRMRPIF and developed the route split procedure for MDWRMRPIF. Due to the network structure and current depot locations, the difference between the optimized routes based on MDWRMRPIF and SDWRMRP is insignificant. A hypothetical instance is created to evaluate the effectiveness of the proposed

MDWRMRPIF algorithm. The result shows 10.4% deadhead distance can be saved by using MDWRMRPIF instead of SDWRMRP.

Third, the sensitivity analysis of the spreading rate parameter shows that this parameter only impacts routes that service roadways with service cycle level "C". This is because the cycle time is a tighter bound for service levels "Metro", "A", and "B". Trucks service these roadways will use up the operation time before it uses up the material. For service cycle level "C", trucks will use up the material before it uses up the operation time if the deadhead time of the route is relatively short comparing to service time. Since the spreading rate is highly related to the snowfall level, the state agency can choose the best plan accordingly for those networks.

Fourth, this study proposed methods for state agencies to optimize their winter maintenance routes. The result can be used as a reference to guide the route design and sector partition. Inefficiencies in current operations can be discovered by comparing the current plan with the optimized plan.

Note that the optimized routes are calculated based on the assumed speed and spreading rate under current fleet size, truck capacity, service cycle time, and plow direction for the current network. If any of the factors above changed, the state agency should recalculate the routes. Otherwise, the optimized routes can be used as a static plan.

7.2 Limitations and Future Research

The present study formulates and solves winter maintenance routing problems based on fixed speed and spreading rate and under current fleet size, truck capacity, service cycle time, and plow direction constraints. However, there are several caveats and limitations.

First, the cycle time is treated as a hard constraint in this study. The service cycle time is a guideline set by Iowa DOT. In practice, the cycle time is not strictly enforced. For some

routes, exceeding the cycle time by a few minutes might result in significantly improved operational efficiency. Therefore, in future study the cycle time constraint can be incorporated as a penalty in the objective function, or as a soft constraint.

Second, the fixed service speed and deadhead speed are assumed in this study. However, in real operation, the speeds could vary depending on the driving habit, road conditions, and traffic conditions. Therefore, the speed could be incorporated in the model as a random parameter which follows a specific distribution. A stochastic programming approach can be explored in the future research.

Third, the optimized routes are designed by assuming the maximum spreading rate. However, as shown in Table 1.1, different spreading rate should be applied for different temperature, precipitation and road surface conditions. The sensitivity analysis with regards to spreading rates (see section 5.4.3.2.2) suggests that the optimized routes could be different if a lower spreading rate is used on certain networks. Using a conservative estimate of spreading rate might result in longer deadhead distance for such networks and inefficient use of resources.

Fourth, the mathematical models may not represent all the practical considerations in real-world operations. Although the optimized routes may reduce deadhead distance, a different plan might be used in real-world operations for practical reasons. For example, to address the drifting snow issue on certain road section, as discussed in section 5.4.3.2.3, one truck is assigned to the problematic road section on purpose. One should consult with district maintenance manager regarding these practical concerns and adjust routes accordingly.

Lastly, since this study uses the metaheuristic algorithm approach, it is not guaranteed that the optimal result was found. The metaheuristic algorithm can solve the problem in

relatively short computation time, but usually generates local optimal solution. The exact algorithm, on the other hand, is guaranteed to find the global optima, but can only solve small-sized problems. By carefully tailoring the metaheuristic algorithm to the problem, the local optimal solution found by metaheuristic could approach to global optima in a statistical sense.

REFERENCES

1. Bartolini, E., Cordeau J.-F., Laporte, G., 2011. Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*:1-44.

2. Benavent, E., Campos, V., Corberán A., Mota, E., 1990. The capacitated arc routing problem. A heuristic algorithm. *Qüestiió* 14 (1–3): 107–122.

3. Beullens, P., Muyldermans, L., Cattrysse, D., Van Oudheusden, D., 2003. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research* 147 (3): 629–643.

4. Bode, C., Irnich S., 2012. Cut-First Branch-and-Price-Second for the Capacitated Arc-Routing Problem. *Operations Research* 60 (5) 1167-1182.

5. Bode, C., Irnich, S., 2012b. In-Depth Analysis of Pricing Problem Relaxations for the Capacitated Arc-Routing Problem. Technical Report LM-2012-06 (revised), Chair of Logistics Management, Johannes Gutenberg University, Mainz, 2012. in: *Transportation Science* 49 (2), 2015, 369-383.

6. Brandão, J., Eglese, R., 2008. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research* 35 (4): 1112–1126.

7. Cai, W.P., Liu, G., Cao, W.S., 2009 A Study of Vehicle and Materials Depot Location Problems for Winter Road Maintenance. Proceedings of the 9th International Conference of Chinese Transportation Professionals, *ICCTP* 2009: Vol. 358. Critical Issues in Transportation System Planning, Development, and Management, pp. 1530-1535.

8. CARP benchmarks. Lower and upper bound values and integer solutions for CARP benchmark instances. <https://logistik.bwl.uni-mainz.de/forschung/benchmarks/> (accessed June, 2019)

9. Crevier, B., Cordeau, J-F., Laporte, G., 2007. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176, pp.756-773

10. Doulabi, S.H.H., Seifi, A., 2013. Lower and upper bounds for location-arc routing problems with vehicle capacity constraints. *European Journal of Operational Research*, 224 (1), pp. 189-208

11. Duhamel, C., Lacomme, P., Prodhon, C., 2011. Efficient frameworks for greedy split and new depth first search split procedures for routing problems. *Computers & Operations Research* 38, 723–739.

12. Duhamel, C., Lacomme, P., Prodhon, C., 2012. A hybrid evolutionary local search with depth first search split procedures for the heterogeneous vehicle routing problem. *Engineering Applications of Artificial Intelligence*. 25 (2), 345–358.

13. Dussault, B., Golden, B., Groer, C., Wasil, E., 2013. Plowing with precedence: a variant of the windy postman problem. *Computers and Operations Research*. 40: 1047–1059.

14. Edmons, J., Johnson, E.L., 1973. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5, pp. 88-124.

15. Eglese, R.W. 1994. "Routing winter gritting vehicles." *Discrete Applied Mathematics* 48 (3): 231–244.

16. Eglese, R.W., Li, L.Y.O., 1996. A tabu search based heuristic for arc routing with a capacity constraint and time deadline. *Meta-heuristics: Theory and Applications*, I.H. Osman and J.P. Kelly, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 633-649

17. Floyd, R. W. 1962. Algorithm 97: shortest path. In: *Communications of the ACM* 5.6, p. 345.

18. Frequently asked questions. Iowa DOT Office of Maintenance. <https://iowadot.gov/maintenance/faq> (accessed October, 2018).

19. Gendreau M., Laporte G., Séguin R., 1996. Stochastic vehicle routing. *European Journal of Operational* Volume 88, Issue 1: 3-12

20. Ghiani G., Guerriero F., Laporte G., Musmanno R., 2004. Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Mathematical Modelling and Algorithms*; 3(3):209–23.

21. Ghiani G., Improta G., Laporte G., 2001. The capacitated arc routing problem with intermediate facilities. *Networks*; 37(3): 134–43.

22. Ghiani G., Laganá D., Laporte G., Mari F., 2010. Ant colony optimization for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Heuristics*; 16(2):211–33.

23. Golden B.L., Wong R.T., 1981. Capacitated arc routing problems. *Network*, 11, pp.305-315.

24. Golden, J.S., Dearmon J.S., Baker E.K., 1983. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research* 10 (1): 47–59.

25. Gundersen A.H., Johansen M., Kjær B.S., Andersson H., Stålhane M., 2017. Arc Routing with Precedence Constraints: An Application to Snow Plowing Operations. In: Bektaş T., Coniglio S., Martinez-Sykora A., Voß S. (eds) *Computational Logistics. ICCL 2017*. Lecture Notes in Computer Science, vol 10572. Springer, Cham

26. Haghani A., Qiao H., 2001. Decision Support System for Snow Emergency Vehicle Routing. *Transportation Research Record*. 1771: 172-178.

27. Haghani A., Qiao H., 2002. Snow Emergency Vehicle Routing with Route Continuity Constraints. *Transportation Research Record*. 1783: 119-124

28. Hajibabai, L., Nourbakhsh, S., Ouyang, Y., and Peng, F., 2014. Network Routing of Snowplow Trucks with Resource Replenishment and Plowing Priorities. Transportation Research Record: *Journal of the Transportation Research Board*. 2440: 16–25.

29. Hajibabai, L., Ouyang, Y., 2016. Dynamic Snow Plow Fleet Management Under Uncertain Demand and Service Disruption. *IEEE Transactions on Intelligent Transportation Systems*. 17(9): 2574–2582.

30. Handa, H., Chapman, L., Yao, X., 2005. Dynamic salting route optimization using evolutionary computation. In 2005 *IEEE Congress on Evolutionary Computation*. 1: 158–165.

31. Hertz, A., Laporte G., Mittaz M., 2000. A tabu search heuristic for the capacitated arc routing problem. *Operations Research* 48 (1): 129–135.

32. Hertz, A., Mittaz M., 2001. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science* 35 (4): 425–434.

33. Hu H., Liu T., Zhao N., Zhou Y., Min D., 2013. A hybrid genetic algorithm with perturbation for the multi-depot capacitated arc routing problem. *Journal of Applied Sciences* 13(16):32–39

34. Kansou A, Yassine A., 2009. A two ant colony approaches for the multi-depot capacitated arc routing problem. In: *IEEE International Conference on Computers and Industrial Engineering*, 2009. CIE 2009, pp 1040–1045

35. Kansou, A., Yassine A., 2010. New upper bounds for the multi-depot capacitated arc routing problem. *International Journal of Metaheuristics*, 1(1), 81–95.

36. Kinable, J., van Hoeve, W.-J., and Smith, S.F. 2016. Optimization Models for a Real-World Snow Plow Routing Problem. *In Proceedings of Integration of AI and OR Techniques in Constraint Programming: 13th International Conference*, CPAIOR 2016, Banff, AB, Canada, 29 May – 1 June 2016. Edited by C.-G. Quimper. Springer International Publishing Cham. pp. 229–245.

37. Krushinsky D., Woensel. T.V., 2015. An approach to the asymmetric multi-depot capacitated arc routing problem. *European Journal of Operational Research* 244(1):100–109

38. Lacomme, P., Prins C., Ramdane-Cherif W., 2001. A genetic algorithm for the capacitated arc routing problem and its extensions. In Boers, E.J.W. ed. *Applications of evolutionary computing*. Springer, 473–483.

39. Lacomme P., Prins C., Ramdane-Cherif W., 2004. Competitive memetic algorithms for arc routing problem, *Annals of Operations Research*, vol. 131, no. 1–4, pp. 159-185.

40. Lenstra J.K., Rinnooy Kan A.H.G., 1976. On general routing problems. *Network*, 6, pp. 273-280.

41. Liu G., Ge Y, Qiu T.Z., Soleymani H.R., 2014. Optimization of Snow Plowing Cost and Time in an Urban Environment: A Case Study for the City of Edmonton. Canadian *Journal of Civil Engineering*. 41(7): 667-675.

42. Liu T., Jiang Z., Geng N., 2013. A memetic algorithm with iterated local search for the capacitated arc routing problem. *International Journal of Production Research*. Volume 51. Issue 10 Pages 3075-3084.

43. Lystlund, L., Wohlk, S. 2012. The service-time restricted capacitated arc routing problem. <https://pure.au.dk/portal/files/35929389/Lystlund_Woelk_Service_Time.pdf> (accessed July, 2018).

44. Jang W., Noble JS., Hutsel T., 2010. An integrated model to solve the winter asset and road maintenance problem. *IIE Transactions*, 42:675-89.

45. Martinelli, R., Poggi M., Subramanian A., 2011. Improved Bounds for Large Scale Capacitated Arc Routing Problem. *Math. Program*., Ser. A 137:409–452

46. Mei Y, Tang K, Yao X., 2009. A global repair operator for capacitated arc routing problem. *IEEE Transactions on Systems*, Man and Cybernetics; 39(3):723–34.

47. Mourão, M.C., Amado, L., Prins C., 2009. Heuristic method for a mixed capacitated arc routing problem: A refuse collection application. *European Journal of Operational Research* 160 (1), 139–153.

48. Muyldermans L, Cattrysse D, Oudheusden DV, Lotan T., 2002. Districting for salt spreading operations. *European Journal of Operational Research* 139(3):521–532

49. Muyldermans L, Cattrysse D, Van Oudheusden D., 2003. District design for arc-routing applications. *Journal of Operational Research Society* 54:1209–1221

50. New, web-based system to better interconnect Iowa dot data, 2017. Transportation Matters Blog. <http://www.transportationmatters.iowadot.gov/2017/07/new-web-based-system-to-better-interconnect-iowa-dot-data-.html> (accessed October, 2018).

51. Omer M., 2007. Efficient routing of snow removal vehicles. Master Dissertation. College of Engineering and Mineral Resources, West Virginia University.

52. Perrier N, Langevin A, Amaya CA., 2008a. Vehicle routing for urban snowplow operations. *Transportation Science*. 42:44-56.

53. Perrier N., Langevin A., Campbell J.F., 2006. A Survey of Models and Algorithms for Winter Road Maintenance. Part I: system design for spreading and plowing. *Computers & Operations Research*, 33, 209-238.

54. Perrier N., Langevin A., Campbell J.F., 2006. A Survey of Models and Algorithms for Winter Road Maintenance. Part II: system design for snow disposal. *Computers & Operations Research*, 33, 239-262.

55. Perrier N., Langevin A., Campbell J.F., 2007. A Survey of Models and Algorithms for Winter Road Maintenance. Part III: vehicle routing and depot location for spreading. *Computers & Operations Research*, 34, 211-257.

56. Perrier N., Langevin A., Campbell J.F., 2007. A Survey of Models and Algorithms for Winter Road Maintenance. Part IV: vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research*, 34, 258-294.

57. Perrier, N., Langevin, A., Campbell, J.F., 2008b. The sector design and assignment problem for snow disposal operations. *European Journal of Operational Research* 189 (2), 508–525.

58. Polacek, M., Doerner K.F., Hartl R.F., V. Maniezzo. 2008. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics* 14 (5): 405–423.

59. Prins C., Calvo R.W., 2005. A fast GRASP with path relinking for the Capacitated Arc Routing Problem. Proceeding of *INOC* 2005 (third international network optimization conference), University of Lisbon (2005), pp. 289-295

60. Prins C., Labadi, N., Reghioui, M., 2009. Tour splitting algorithms for vehicle routing problems. International Journal of Production Research 47, 507–535.

61. Prins C., Lacomme P., Prodhon C., 2014. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C,* 40, pp. 179-200

62. Quirion-Blais, O., Langevin, A., Trepanier, M., 2017. A case study of combined winter road snow plowing and de-icer spreading. *Canadian Journal of Civil Engineering*. 44(12): 1005-1013.

63. Roadway Diagram Picture. Undivided roadway from <https://www.dot.state.mn.us/trafficeng/safety/threelane.html>. Divided roadway diagram from <https://commons.wikimedia.org/wiki/File:2-lane_interstate_highway_road.png>. (accessed Nov, 2018).

64. Salazar-Aguilar, M.A., Langevin, A., Laporte, G., 2012. Synchronized arc routing for snow plowing operations. *Computers and Operations Research*. 39(7): 1432–1440.

65. Salt Institute. Snow fighters Handbook. <https://idot.illinois.gov/Assets/uploads/files/Transportation-System/Manuals-Guides-&-Handbooks/T2/L026%20The%20Snowfighters%20Handbook.pdf> (accessed Apr, 2019)

66. Santos, L., Coutinho-Rodrigues J., Current J., 2010. An improved ant colony optimization-based algorithm for the capacitated arc routing problem. *Transportation Research Part B*: Methodological 44 (2): 246–266.

67. Snow And Ice Control Plan. Village of Algonquin Public Works Department. <https://www.algonquin.org/egov/documents/1521725178_38353.pdf> (accessed July, 2018).

68. Snow Plow Truck. NDDOT. <https://www.dot.nd.gov/divisions/maintenance/docs/plowtruck.pdf> (accessed Nov, 2018)

69. Survey of 23 States Shows $1 billion Spent, 8 Million Work Hours Logged and Six Million Tons of Salt Used to Battle Winter Weather, 2015. American Association of State Highway and Transportation Officials. <https://news.transportation.org/Pages/NewsReleaseDetail.aspx?NewsReleaseID=1443> (accessed July, 2018).

70. Tagmouti M., Gendreau M., Potvin J-Y., 2007. Arc routing problems with time-dependent service costs. *European Journal of Operational Research*. 181(1): 30-39.

71. Tagmouti M., Gendreau M., Potvin J-Y., 2010. A variable neighborhood descent heuristic for arc routing problems with time-dependent service costs. *Computers and Industrial Engineering*. 59, pp.954-963.

72. Tagmouti M., Gendreau M., Potvin J-Y., 2011. A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C*, Emerging Technologies, 19, pp. 20-28.

73. Tang K., Mei Y., Yao X., 2009. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*. 13(5): 1151-1166.

74. Ulusoy, G., 1985. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research* 22 (3): 329–337.

75. Usberti, F.L., França P.M., and França A.L.M., 2012. Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers & Operations Research*, 40, pp. 3206-3217.

76. Usman T., Fu L., Miranda-Moreno L.F., 2010. Quantifying safety benefit of winter road maintenance: accident frequency modeling. *Accident Analysis and Prevention*; 42:1878–87.

77. Willemse, E. J., Joubert, J. W., 2016a. Constructive heuristics for the mixed capacity arc routing problem under time restrictions with intermediate facilities. *Computers & Operations Research*, 68:30-62.

78. Willemse, E. J., Joubert, J. W., 2016b. Splitting procedures for the mixed capacitated arc routing problem under time restrictions with intermediate facilities. *Operations Research Letters*, 44(5):569-574.

79. Willemse, E. J., 2016c. Heuristics for large-scale Capacitated Arc Routing Problems on mixed networks. Doctor dissertation. University of Pretoria, Pretoria, South Africa

80. Xing L, Rohlfshagen P, Chen Y, Yao X., 2010. An evolutionary approach to the multidepot capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation* 14(3):356–374

81. Xu, X., Mahmassani, H. S, Hong, Z. Alfelor, R. M., 2017. Dynamic Predictive Strategies for Urban Snowplow Routing. *Transportation Research Board* 96th Annual Meeting. Washington DC.