Routing the vehicles for minimizing traveling distance under green and multi-depot context

by

Weiheng Zhang

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Supply Chain Management

Asper School of Business

University of Manitoba

Winnipeg

Abstract

In this thesis, an Ant Colony System with Variable Neighborhood Search algorithm (ACSVNS) is proposed to solve Multi-depot Vehicle Routing Problem (MDVRP). A new variant of vehicle routing problem, Multi-depot Green Vehicle Routing Problem (MDGVRP), is also formulated. In ACSVNS, two types of ants are used for two different purposes. The first type of ant is used to assign customers to depots while the second type of ant is used to find the routes. ACSVNS applies perturbation scheme, relocate local search and revised swap local search to improve the solution quality. Based on the experiment results of the 23 benchmark instances of MDVRP, ACSVNS can match the existing best known solution for 16 instances, which is the third best in all algorithms. Additionally, ACSVNS found new best solutions for the instance 5, 6 and 7. These results prove that ACSVNS has a good performance on solving MDVRP. Besides, in this thesis, a Multi-depot Green Vehicle Routing Problem (MDGVRP) is also formulated. Based on ACSVNS, a Two-stage Ant Colony System (TSACS) algorithm is proposed to find solutions for this problem. The solution for MDGVRP is useful for companies which employ the Alternative Fuel-Powered Vehicles (AFVs) to deal with the obstacles brought by the limited number of the Alternative Fuel Stations (AFSs). This thesis adds vehicles capacity and tank capacity constraints to make it more meaningful and closer to real-world case. The numerical experiment is performed on randomly generated problem instances to evaluate the performance of proposed algorithms.

Acknowledgements

I would like to express the deepest gratitude to my advisor Dr. S. S. Appadoo and co-advisor Dr. Yuvraj Gajpal. Their guidance helped me in all the time of research and writing of this thesis. My sincere thanks also go to my committee members, Dr. Sara Hajmohammad and Dr. Ruppa Thulasiram for their insightful comments and feedbacks. I would also thank the Province of Manitoba to provide my fellowship and my family to support me. Finally, I thank all people who helped me for my research work and writing this thesis.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Introduction on Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is a very popular research topic in supply chain management (SCM). VRP is developed based on the Traveling Salesman Problem (TSP) proposed by Dantzig and Ramser (1959), which focuses on minimizing the total traveling distance of salesman during the process of visiting all customers. Due to the importance of reducing transportation cost for companies and related industry, the studies of VRP are gradually growing and have been an important topic in SCM and operations research. In the past several decades, based on the different transportation scenarios in real world, many variant VRP problems were proposed, such as Time Dependent Vehicle Routing Problem (TDVRP), Green Vehicle Routing Problem (GVRP), etc.

The classic VRP problem is depicted as the following graph-theoretic problem. Let $G = (V, A)$, where $V = \{0,1, \dots, n\}$ is the set of nodes and A is the set of arcs. Excepting for the depot node $(i = 0)$, rest of the nodes $(i = 1,2,3, \dots, n)$ represent the client who has the delivery demand $(D_i > 0)$ and can be visited by vehicle only once. The route from node k to node j is represented as the arc between these two nodes and each one has a weight $C_{kj} > 0$, which represents the cost (could be distance or time, etc.) of delivering cargos through this route between node k and j (Laporte & Osman, 1995). Figure 1 shows a classical vehicle routing problem which has only one depot (D_1) and five customers $(C_1, C_2, C_3, C_4, C_5)$.

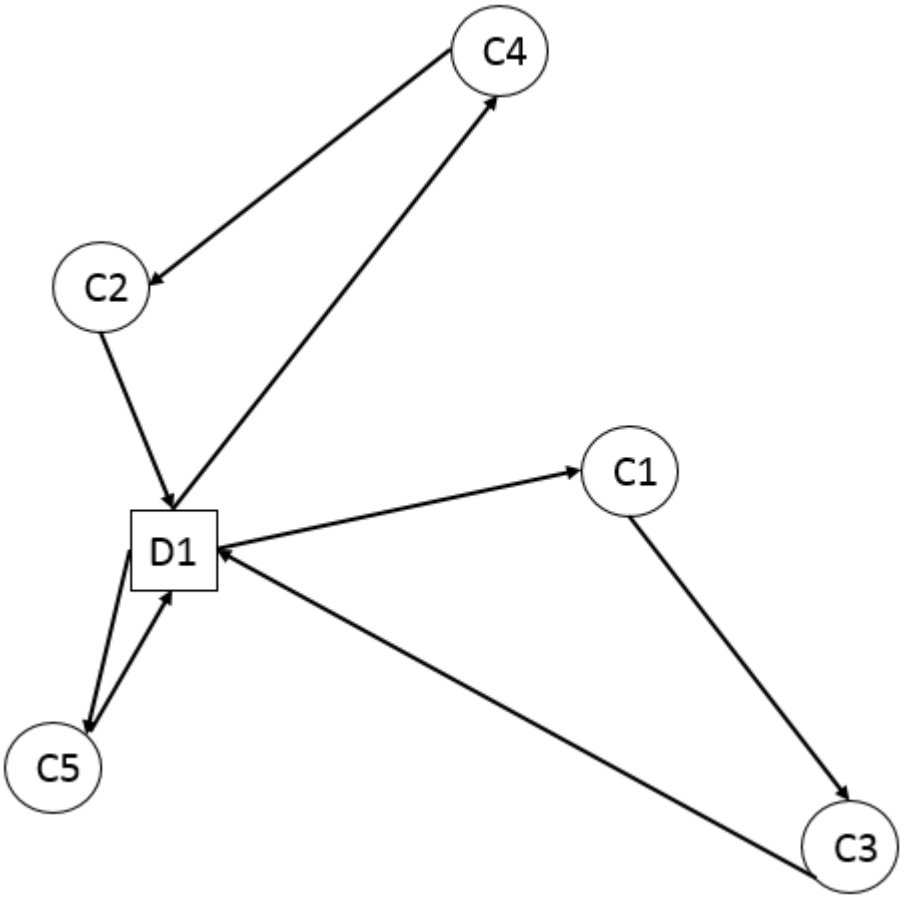

Fig.1. Classical Vehicle Routing Problem

The literature on VRP can be categorized into two main streams. One stream of VRP studies focus on proposing new models of VRP variants by considering more constraints. For instance, based on the classic VRP, Vehicle Routing Problem with Time Window (VRPTW) adds the constraint that different customer can only be served within a specific time window. The studies of those variants of classic VRP are important and necessary because they represent the real world cases. The VRP problems are NP-hard, which means that if we apply exact algorithms to solve a VRP problem, with the size of problem increase, the solve time will increases exponentially . Therefore, another stream of VRP studies focus on designing efficient algorithms to solve the existing VRP problems faster and better.

1.2 Introduction on Green Vehicle Routing Problem (GVRP)

Due to the growing environmental concern worldwide, green logistics is attracting people's attention. Under this context, Green Vehicle Routing problem (GVRP) has attracted the attention of many researchers (Erdoğan & Miller-Hooks, 2012; Lin, Choy, Ho, Chung, & Lam, 2014; Schneider, Stenger, & Goeke, 2014). The classical Vehicle Routing Problem (VRP) only focuses on minimizing the total transportation cost (or distance) generated in the process of distribution services, while the GVRP emphasizes on addressing environmentally sustainable issues in delivery distribution of supply chains along with the optimal economic cost of delivery (Lin et al., 2014). According to the research of Erdoğan and Miller-Hooks (2012), the design of GVRP requires the use of the Alternative Fuel-powered Vehicles (AFV), which relies on greener fuel source such as electricity, natural gas, hydrogen, etc. However, Erdoğan and Miller-Hooks (2012) pointed out two main obstacles encountered while replacing the conventional vehicles with the AFVs: 1) the limited capacity of fuel tank or batteries of AFVs, and 2) the scarcity of Alternative Fuel Stations (AFSs). These obstacles make the problem formulation and algorithm design of GVRP more complex than those of classic VRP.

1.3 Recent Issues and Motivation

When designing transportation routes and network in a single-depot context, more and more practitioners have been considering both financial and environmental aspects. The research and findings on Green Vehicle Routing problem (GVRP) can guide those practitioners to satisfy their demands because GVRP emphasizes on addressing environmentally sustainable

issues in delivery distribution of supply chains along with the optimal economic cost of delivery in a single-depot context.

The Multi-Depot Vehicle Routing Problem (MDVRP) is another branch of VRP. In MDVRP, the fleet of vehicles serves customers from several depots and returns to the same depot. Research about the MDVRP is meaningful for companies that have a wide range of business establishments and have more than one depot. Due to the importance of MDVRP, it has attracted many researchers and many algorithms are proposed to solve this problem.

However, in recent years, many large-scale multinational companies such as UPS, Coca-Cola and GM have especially paid special attention to their environmentally sustainable performance. They are exhausting their ability to keep balance between economic performance and environment protection. For these companies, the design of transportation routes based on GVRP or MDVRP may not provide the desired optimal solution.

Therefore, in this thesis, a new variant of problem called the Multi-Depot Green Vehicle Routing Problem (MDGVRP), is addressed. In MDGVRP, the AFVs start from different depots, serve customers, and at the end return to their original depots. Due to the limited capacity of fuel tank of AFVs and the scarcity of AFSs, each AFV is required to visit AFS for refueling. The objective of MDGVRP is to minimize the route distance of the AFV fleets. Thus, compared with MDVRP or GVRP, MDGVRP has more constraints and subsequently, is different to formulate and solve.

MDGVRP is an extension of MDVRP because it turns to MDVRP when fuel tank capacity is set to be unlimited. In this thesis, firstly, an Ant Colony System with Variable Neighborhood Search algorithm (ACSVNS) is proposed to solve MDVRP. Based on ACSVNS, revised

Partition Based Algorithm (PBA) and Two-stage Ant Colony System (TSACS) algorithm are proposed to solve MDGVRP. The main reason for using ACS to solve MDVRP and MDGVRP is that it has been successfully used to solve many variants of VRP problem (e.g. Gajpal & Abad, 2009a; Gajpal & Abad, 2009b; Zhang, Gajpal, & Appadoo, 2017). The ACS proposed by Gajpal and Abad (2009a) and Gajpal and Abad (2009b) are still the best performing algorithm for sloving Vehicle Routing Problem with Simultaneous Delivery and Pickup (VRPSDP) and Vehicle Routing Problem with Backhauls (VRPB). The successful application of ACS to solve variant problems of VRP motivates us to use ACS for solving the problems considered in this thesis.

1.4 Contribution and Overview of This Thesis

The main contributions and structure of each chapter in this thesis are provided as follows: In chapter 2 of this thesis, a new algorithm, Ant Colony System with Variable Neighborhood Search algorithm (ACSVNS) is proposed to solve MDVRP. Within ACSVNS, perturbation scheme, relocate local search and revised swap local search are applied to improve the solution quality. These neighborhood search techniques are applied based on Remove Cost (*RC*), which is the total route distance decrease due to removing the customer, and Insertion Cost (*IC*), which is the least total route distance increase due to inserting the customer. In ACSVNS, when generating each ant solution, *RC* and *IC* matrices only need to be calculated once at the beginning, and only the related elements in *RC* and *IC* matrices are need to be updated during the following solution generation process, which can avoid repeated computation. This reuse technique is proposed in this thesis for the first time which dramatically improves the efficiency

of the whole algorithm

Besides, a revised swap local search is developed and applied in ACSVNS. This revised swap local search is also proposed in VRP literature for the first time. The main difference between the revised swap local search and the classic swap local search in literature is that in this new swap local search, the two selected customers are swapped between their routes instead of their positions. Therefore, revised swap local search can search more neighborhood solutions. Combined with the reuse technique, revised swap local search is much more effective and efficient than the classic swap local search. More details about ACSVNS, *RC&IC* matrix calculation, revised swap local search and other local search techniques are introduced in Chapter 2.

In Chapter 3, a new VRP variant problem, Multi-Depot Green Vehicle Routing Problem (MDGVRP) is formulated. Revised Partition Based Algorithm (PBA) and Two-stage Ant Colony System (TSACS) algorithm are proposed to solve MDGVRP. We provide mathematical formulation for MDGVRP. 17 small-sized instances and 60 large-sized instances are generated to test the performance of these two algorithms. All experiment results are also provided in Chapter 3. This MDGVRP model is very important for the companies which apply electric vehicles or other alternative fueled vehicles to design their distribution network under a multi-depot context.

At the end of this thesis, the conclusion is in Chapter 4 and the following is all reference literature.

CHAPTER 2

ANT COLONY SYSTEM WITH VARIABLE NEIGHBORHOOD SEARCH ALGORITHM (ACSVNS) FOR THE MULTI-DEPOT VEHICLE ROUTING PROBLEM

In this chapter, an Ant Colony System with Variable Neighborhood Search algorithm (ACSVNS) is proposed to solve Multi-depot Vehicle Routing Problem (MDVRP). In ACSVNS, two types of ants are used for two different purposes. The first type of ants is used to assign customers to depots while the second type of ants is used to find the routes. ACSVNS applies perturbation scheme, relocate local search and revised swap local search to improve the solution quality. The ACS based algorithm proposed in this chapter lays a foundation of building the algorithm in solving MDGVRP in chapter 3.

2.1. Introduction

Vehicle Routing Problem (VRP) is an important research topic in logistic field because of the relationship between routing network and transportation cost. However, in the classic VRP, it is assumed that there is only one depot and it is obviously oversimplified for many real-life study cases. Most large-scale retail and transportation industries, have much more than one depot. Given this background, the Multi-Depot Vehicle Routing Problem (MDVRP) has attracted considerable attention among researchers and practitioners. In MDVRP, fleet of vehicles serve customers from several depots and returns to the same pre-assigned depot.

MDVRP is a NP-hard problem which means it is impossible to solve large-sized MDVRP problem with exact methods. Besides, there is more than one depot considered in MDVRP, therefore, if the same customer is assigned to different depots, the total distance will change.

Due to these factors, solving MDVRP is very challenging. Researchers have been trying to propose and revise algorithms to solve MDVRP effectively, and some of those algorithms have shown excellent performance in solving MDVRP, for instances, CGL method (Cordeau et al, 1997) , ACO with the ant-weight strategy and the mutation operation (ACO-WM) (Yu et al, 2009), PIACO method (Yu et al, 2011), HGSADC algorithm (Vidal, Crainic, Gendreau, Lahrichi, & Rei, 2012) and ELTG (Escobar, Linfati, Toth, & Baldoquin, 2014). In this thesis, we developed an Ant Colony System Variable Neighborhood Search algorithm (ACSVNS) to solve MDVRP. The main contributions of this chapter to the literature of MDVRP are as follows:

1. A new outstanding algorithm, ACSVNS, is proposed;
2. New best solution for three benchmark instances are found;
3. A new local search technique, called revised swap local search, is developed.
4. The reuse of database is proposed to increase the efficiency of local search techniques.

The structure of this chapter is organized as follows. In chapter 2.2, related literature is reviewed. Chapter 2.3 presents description of MDVRP. Chapter 2.4 presents the details of ACSVNS and how it solves MDVRP. Numerical experiments are presented in chapter 2.5 and are followed by conclusion in chapter 2.6.

2.2. Literature Review

The MDVRP was first described in the research of Cassidy and Bennett (1972). It is a generalization of the standard VRP in which depots are considered to be more than one (Yu, Yang, & Xie, 2011). The research of MDVRP mainly focuses on proposing and developing

new methods and algorithms to solve the problem or proposing new MDVRP variant models (e.g. Cassidy & Bennett, 1972; Dantzig & Ramser, 1959; Escobar, Linfati, Toth, & Baldoquin, 2014; Laporte & Osman, 1995; Montoya-Torres, López Franco, Nieto Isaza, Felizzola Jiménez, & Herazo-Padilla, 2015; Nagy & Salhi, 2005; Salhi & Nagy, 1999; Ting & Chen, 2013; Vidal, Battarra, Subramanian, & Erdoŧan, 2015; Vidal, Crainic, Gendreau, Lahrichi, & Rei, 2012; Yu, Yang, & Xie, 2011).

The work of Montoya-Torres et al. (2015) revealed that most researchers tend to solve the MDVRP by heuristics or meta-heuristics. Some of the algorithms proposed in literature have an excellent performance in solving MDVRP. For instances, Yu et al. (2011) converted the MDVRP into Single-depot VRP (SVRP) by adding a virtual depot in the first step, and then applied an improved Ant Colony Optimization (ACO) to solve the Single-depot VRP (SVRP). Vidal et al. (2012) solved the MDVRP by using a hybrid genetic algorithm and this algorithm has the best overall performance in solving MDVRP. Escobar et al. (2014) designed a hybrid granular tabu search algorithm to solve MDVRP. This algorithm can solve MDVRP instance in a very short time with satisfied solution quality.

There are also some researchers who have proposed new variant of MDVRP models and designed solution methods. Nagy and Salhi (2005) solved MDVRP with pickups and deliveries problem by dividing customers into borderline and non-borderline customers. Jabir, Panicker and Sridharan (2017) proposed a Multi-depot Green Vehicle Routing Problem (MDGVRP) which minimizes the total carbon emissions for all vehicles within multi-depot (their MDGVRP model is developed from Pollution Routing Problem (PRP) which minimizing the total Green House Gas (GHG) emissions and does not consider tank capacity of vehicles, while the

MDGVRP model proposed in Chapter 3 is developed from Green-VRP (G-VRP) which minimizes the total travelling distance and considers tank capacity of Electric Vehicles). They also proposed a heuristics algorithm to solve this problem.

In this chapter, we proposed Ant Colony System with Perturbation and Revised Swap local search (ACSVNS) algorithm to solve MDVRP. The details of ACSVNS and numerical experiment results are introduced in the following sections.

2.3. Problem Description.

A standard MDVRP can be described as designing transportation routes from more than one depot to a set of geographically scattered customers. Each customer is associated with a fixed allocated demand to be delivered. Each customer is visited by the vehicle fleet only once, and the demand of customer is satisfied after each visit. A vehicle starts from a depot, serves customers one by one, and finally returns to its original assigned depot. During the service process, when the remaining cargo of a vehicle cannot satisfy the demand of the next customer, the vehicle returns to depots for reloading. For each depot, the maximum number of routes cannot exceed the vehicle constraint. For each route, the maximum route distance cannot exceed the route duration constraint. The objective of the problem is to minimize the total distance travelled by all vehicles.

All necessary notations which are used to formulate MDGVRP are defined as follows:

N The total number of customers

M The total number of depots

K The total number of vehicles

T_k Route duration limit of vehicle k

P_k Load capacity of vehicle k

Q_i The demand of vertex i

d_{ij} The distance (cost) between vertices i and j

Decision Variables

x_{ijk} is a binary variable taking value 1 if vehicle travels directly from vertex i to j by the k^{th} vehicle. Otherwise, x_{ijk} equals 0.

y_i is an auxiliary variable used to avoid subtour elimination.

A standard mathematic formulation of MDVRP, which is extracted from the work of Kulkarni and Bhave (1985), is provided as follows:

$$\text{Minimize } \sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\sum_{k=1}^{K}(d_{ij}x_{ijk}) \qquad 2.1$$

Subject to

$$\sum_{i=1}^{N+M}\sum_{k=1}^{K}x_{ijk} = 1 \quad j = 1, \dots, \mathrm{N} \qquad 2.2$$

$$\sum_{j=1}^{N+M}\sum_{k=1}^{K}x_{ijk} = 1 \quad i = 1, \dots, \mathrm{N} \qquad 2.3$$

$$\sum_{i=1}^{N+M}x_{ihk} - \sum_{j=1}^{N+M}x_{hjk} = 0 \quad k = 1, \dots, K \quad h = 1, \dots, N+M \qquad 2.4$$

$$\sum_{i=1}^{N+M}Q_i\sum_{j=1}^{N+M}x_{ijk} \le P_k \quad k = 1, \dots, K \qquad 2.5$$

$$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M} d_{ij}x_{ijk} \leq T_k \quad k = 1, \dots, K \tag{2.6}$$

$$\sum_{i=N+1}^{N+M}\sum_{j=1}^{N} x_{ijk} \leq 1 \quad k = 1, \dots, K \tag{2.7}$$

$$\sum_{j=N+1}^{N+M}\sum_{i=1}^{N} x_{ijk} \leq 1 \quad k = 1, \dots, K \tag{2.8}$$

$$y_i - y_j + (M+N)x_{ijk} \leq N + M - 1 \quad For\ 1 \leq i \neq j \leq N\ and\ 1 \leq k \leq K \tag{2.9}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i, j, k \tag{2.10}$$

In this formulation, *i* and *j* denote nodes, *k* means the index of vehicle. *N* is the number of customers, *M* is the number of depots and *K* is the number of vehicles. Constraints 2.2 and 2.3 restricts that each customer has to be visited once by one vehicle. Constraint 2.4 ensures that for each depot, incoming number of vehicles is equal to the departure number of vehicles. Constraints 2.5 and 2.6 are the vehicle capacity constraint and route duration constraint, respectively. Constraints 2.7 and 2.8 ensure that vehicles depart and return to the same depot. The following constraint 2.9 eliminates the sub-tour and constraint 2.10 restricts that all variables are binary variables.

2.4. Ant Colony System with Variable Neighborhood Search (ACSVNS).

In this chapter, Ant Colony System with Variable Neighborhood Search (ACSVNS) is proposed to solve MDVRP. ACSVNS is an extension of the classic Ant Colony System (ACS)

algorithm. ACS is one of the famous algorithm based on swarm intelligence. Ants always can find the shortest route between their nest and the food (Dorigo, Maniezzo, & Colorni, 1996) because they can leave pheromone when they are walking and the shortest route usually can accumulate the highest level of pheromone. By simulating the food-seeking behaviors of ant colonies in nature, the Ant Colony System (ACS) algorithm was developed by Dorigo et al. (1996). During the last 20 years, the ACS algorithm and its derived algorithms have been successfully applied to solve the VRP and its variants (e.g. Bell and McMullen (2004), Dorigo et al. (1996), Gambardella, Montemanni, Rizzoil and Lucibello (2007), Gajpal and Abad (2009), Lin et al. (2014), Montoya-Torres et al. (2015) , Yu et al. (2011) and Ting and Chen (2013), etc.).

The proposed ACSVNS algorithm uses two types of ants, *depot-ants* and *route-ants*. The *depot-ants* are used to assign customers to depot and *route-ants* are used to generate MDVRP routes. The main component of ant colony is the definition of trail intensity. Therefore τ_{ik}^{d} and τ_{ij}^{r} are denoted as the trail intensity related to *depot-ants* and *route-ants*, respectively. Specifically, the term τ_{ik}^{d} represents the trail intensity of *depot-ants* for assigning customers i to depot k, and τ_{ij}^{r} represents the trail intensity of *route-ants* for traveling to node j from node i (node *i* and *j* can be either a customer or a depot). After ants assign customer into different depots and generate MDVRP routes, Perturbation scheme, relocate local search and revised swap local search is applied one by one to improve the solution quality.

ACSVNS Algorithm:

1. **Procedure** ACSVNS()
2. **Begin**
3. *Set_Parameters*;
4. *Initialize_trail_intensity_matrix*; /* *Initialize trail intensity* τ_{ik}^{d} *and* τ_{ij}^{r} */
5. $V^{*} \leftarrow \infty$; /* V^{*} *represents the globle best distance value found so far* */
6. **for** *i* ← *1 to z* **do** /* *z is the number of iterations* */
7. **for** *j* ← *1 to y* **do** /* *y is the number of ants* */
8. $CD_k \leftarrow AssignCustomerToEachDepot(\tau_{ik}^{d})$; /* *Depot ants assign customer to each depots* */
9. $S \leftarrow GenerateMDVRPRoutes(CD_k)$; /* *Route ants generate MDVRP routes* */
10. $S^{L} \leftarrow VNS(S, ln_1)$; /* *Apply VNS local search to ant solution S,* ln_1 *is the loop number for VNS* */
11. **if** $Distance(S^{L}) < V^{*}$ **then** /* *If find a new global best solution, then update it* */
12. $S^{*} \leftarrow S^{L}$;
13. $V^{*} \leftarrow Distance(S^{L})$;
14. **end if**
15. **end for**
16. $S \leftarrow S^{*}$; /* *At the end of each interation, apply VNS local search one more time based on* S^{*} */
17. $S^{L} \leftarrow VNS(S, ln_2)$; /* *Apply VNS local search to ant solution S* */
18. **if** $Distance(S^{L}) < V^{*}$ **then** /* *If find a new global best solution, then update it* */
19. $S^{*} \leftarrow S^{L}$;
20. $V^{*} \leftarrow Distance(S^{L})$;
21. **end if**
22. $UpdateTrailIntensity(S^{*}, \tau_{ik}^{d}, \tau_{ij}^{r})$; /* *Update trail intensity based on the global best solution* */
23. **end for**
24. **Return** S^{*}, V^{*};
25. **End**

Fig. 2. Pseudo code of ACSVNS algorithm

Figure 2 illustrates the pseudo code of ACSVNS algorithm. At the beginning of ACSVNS, all necessary parameters and trail intensity matrices (τ_{ik}^{d}) and (τ_{ij}^{r}) are initialized (line 3 and 4). The main loop (from line 6 to 23) is the iteration loop for ACSVNS which contains an inner loop (from 7 to 15). This inner loop is the ant loop. In each ant loop, an ant solution is generated and this solution is improved using VNS local search. In the main loop, after the inner ant loop, VNS local search is applied one more time based on the best solution found so far to find a new best solution (line 16 to 21). This step is very important for ACSVNS because the best solution found so far already has a high level of solution quality. Compared with a normal ant

solution, it has a higher probability to have a better neighborhood solution. Finally, at the end of each main loop, the trail intensity τ_{ik}^{d} and τ_{ij}^{r} are updated based on the best solution found so far (line 22) which guide the ants to generate solution in the next iteration.

In the following section, all key steps of ACSVNS are provided in detail.

2.4.1. Initial Trail Intensity Matrixes

Ants generate solutions according to the trail intensity. Therefore, the first step in ACSVNS is to initialize trail intensity of *depot-ants* τ_{ik}^{d} and trail intensity of *route-ants* τ_{ij}^{r}. In MDVRP, there is a route duration constraint which means the distance for route *k* cannot exceed the route duration T_k. Therefore, some customers can only be assigned to a few or even only one depot because of the limited route duration. The customer i can be feasibly assigned to depot k if following condition is satisfied:

$$2 \cdot d_{ik} \leq T_k \qquad 2.11$$

The term d_{ik} represents the distance between customer i and depot k. Based on this condition, trail intensity τ_{ik}^{d} can be initialized as follows.

$$\tau_{ik}^{d} = \begin{cases} 0.01, & if\ customer\ i\ can\ be\ assigned\ to\ depot\ k \\ 0, & otherwise \end{cases}, \quad \forall i \in C, k \in D$$

Term C denotes the customer set and D denotes the depot set. V represents the union of C and D. For all the nodes in V, the trail intensity for *route-ants* τ_{ij}^{r} is initialized as $\tau_{ij}^{r} = 0.01$ $(\forall i, j \in V)$. Numerical experiment indicates that the absolute values of initial τ_{ij}^{d} or τ_{ij}^{r} do not affect the solution. We varied the values of τ_{ij}^{d} and τ_{ij}^{r} from 10 to 0.00001 but we could not find significant difference on the final solution. Therefore, we randomly set the initial values of τ_{ij}^{d} and τ_{ij}^{r} as 0.01.

2.4.2. Generate Ant Solution

In ACSVNS, each artificial ant generates one feasible solution in every iteration. For MDVRP, a feasible solution is a set of VRP routes for all depots (MDVRP routes). The MDVRP solution for each artificial ant is generated in two steps as described below.

Step 1. Assign Customers to Depots

The first step is to assign customers to different depots using *depot-ant*. The probability of assigning customer i to depot k is as follows:

$$P_{ik}^{d} = \frac{\tau_{ik}^{d}}{\sum_{l \in D} \tau_{il}^{d}}, \qquad \forall i \in C, k \in D \tag{2.12}$$

Assignment of customers to depot starts with first customer and then other customers are iteratively assigned to different depots to form set CD_k $(\forall k \in D)$. The term CD_k $(\forall k \in D)$ represents the set of customers assigned to depot k. At the end of this step, a set CD_k is obtained for each depot k $(\forall k \in D)$.

Step 2. Generate VRP Routes

After assigning customers to depots, *routes-ant* is used to generate VRP routes for each depot based on the trail intensity of *route-ants* τ_{ij}^{r}. Let VC_k $(\forall k \in D)$ stands for the set of unvisited customer for depot k. At the beginning, VC_k is initialized as $VC_k = CD_k$. For depot k, the VRP route starts from depot k and visits the customer one by one from set VC_k to form VRP route. The probability of visiting customer j from the current customer i is as follows:

$$P_{ij} = \frac{\tau_{ij}^{r}}{\sum_{l \in VC_k} \tau_{il}^{r}}, \qquad \forall j \in VC_k \tag{2.13}$$

The selected customer is then removed from the set VC_k. The procedure continues till set

VC_k becomes an empty set. During the route generation process, if the route cannot satisfy the demand constraint (constraint 2.5) or route duration constraint (constraint 2.6), a new route need to be arranged for the current depot. Specifically, demand constraint means that for each vehicle, the remaining cargo load level must be greater or equal than the demand of the next node (customer of depot) on route. Duration constraint means that for each vehicle, the total route length cannot exceed a certain limit.

2.4.3. Variable Neighborhood Scheme (VNS)

VNS is a set of local search techniques to improve the quality of ant solutions. As shown in Figure 2, in ACSVNS, VNS is applied in each ant loop (line 10) for the generated ant solution and at the end of each main loop (line 17) for the best solution found so far.

VNS consists of three neighborhood solution search techniques, specifically, perturbation scheme, relocate local search and revised swap local search. Figure 3 illustrates the pseudo code of VNS. The main structure of VNS is a loop (from line 4 to 12). In each loop, the ant solution generated in the previous loop (S') is applied perturbation scheme, relocate local search and revised swap one by one. Finally, return the best neighborhood solution found in VSN (S^*). In ACSVNS, all neighborhood techniques are applied based on Remove Cost (*RC*) matrix and Insertion Cost (*IC*) matrix. As can be seen in Figure 3, *RC* and *IC* only calculate once in perturbation scheme and the following relocate and revised swap local search can reuse these matrices, which dramatically improve the efficiency of algorithm. This is also a distinct feature of ACSVNS. The details of *RC*&*IC* matrix definition, calculation, perturbation scheme, relocate local search and revised swap local search are discussed in the following sections

VNS Local Search:

```
1.  Procedure VNS(S, ln) /* S is an ant solution, a set of MDVRP routes. ln is the loop number of VNS */
2.  Begin
3.    Distance(S^L) ← ∞; /* S^L represents the best solution found in VNS */
4.      for i ← 1 to ln do
5.        S' ← PerturbationScheme(S); /* Apply the PerturbationScheme for the ant solution S */
6.        S' ← RelocateLocalSearch(S', RC, IC); /* Apply the RelocateLocalSearch for S'. RC is remove cost matrix and
    IC is insertion cost matrix which are initialized and calculated in Perturbation Scheme*/
7.        S' ← RevisedSwapLocalSearch(S', RC, IC); /* Apply the RevisedSwapLocalSearch for S' */
8.        if Distance(S') < Distance(S^L) then /* If find a new best VNS solution, then update it */
9.          S^L ← S';
10.       end if
11.       S ← S'; /* The next round of VNS is based on S' instead of S^L */
12.     end for
13.   Return S^L;
14. End
```

Fig. 3. Pseudo code of VNS local search

2.4.3.1. Perturbation Scheme

Perturbation scheme is an effective method to diversify solutions and escape from the local optimal. Therefore, sometimes, perturbation scheme cannot improve solution immediately, but it can help the follow-up local search to find a better solution.

```
Perturbation Scheme:
1.  Procedure PerturbationScheme (S) /* S is an ant solution, a set of MDVRP routes. */
2.  Begin
3.    r ← Random(); /* Generate a random number between 20% to 40 % of total customer number */
4.    S′ ← RemoveCustomersFromRoutes(S, r, NC); /* Remove customer from solution S, NC is the unselected cust set */
5.    RC ← CalculateRemoveCost(S′); /* RC store the remove cost for each customer */
6.    IC ← CalculateInsertionCost(S′); /* IC store the minimum insertion cost for each customer */
7.    for i ← 1 to R do /* R is the element number of NC */
8.      BestCust ← 0; /* Variable BestCust records which customer is inserted into routes in this round */
9.      BestCust ← SelectBestCust(NC, S′, RC, IC); /* Select the best customer who has the minimum distance increase*/
11.     if BestCust ≠ 0 then
12.       InsertBestCust(BestCust, S′); /* Insert the best customer */
13.     else /* If there is no customer can be inserted into the existing routes, then start a new route */
14.       BestCust ←SelectBestCustRandomly(NC); /* Randomly choose one customer from NC as BestCust */
15.       StartNewRouteForBestCust(BestCust, S′);
16.     end if
17.     UpdateRemoveCost(CC, S′, RC); /* Update remove cost for influenced customers */
18.     UpdateInsertCost(CC, S′, IC); /* Update insert cost for influenced customers */
19.     RemoveBestCustFromNC(BestCust); /* Remove the best customer from NC */
20.   end for
21.   Return S′;
22. End
```

Fig. 4. Pseudo code of Perturbation Scheme

Figure 4 demonstrates the structure of perturbation scheme. It can be seen that the main steps of perturbation scheme are removed (line 4) and reinserted (line 7 to 20). During the removal process, 20% to 40% of total number of customers are randomly removed from the original solution *S* and store them into the unvisited customer set *NC*. If more customers are removed then the neighborhood solution will be completely new. On the other hand, if less customers are removed, the solution will not be diversified. Therefore, finally, we determined to remove 20% to 40% of customers in each round.

After removing process, remove cost (*RC*) matrix, insertion cost (*IC*) are generated (line 5 and 6). The objective of MDVRP is to minimize the total travel distance of all vehicles. The principle of all local search techniques is to change the positions of customers to build neighborhood solutions. By calculating the total distance of neighborhood solution, it is known

that whether this neighborhood solution is better than original solution. When changing a customer to a new positions, the total changed distance is always the sum of the changed distance because of removing this customer from his previous position (Remove Cost) and the changed distance because of inserting this customer to new position (Insertion Cost). Therefore, by storing all these values into *RC* and *IC* matrices, the values need to be loaded from the matrices and do not need to be recalculated all the times. Based on *RC* and *IC*, the net cost matrix (A^{Net}) are calculated. The insertion part of perturbation scheme, the following relocate local search and revised swap local search are executed based on *RC, IC* and A^{Net} matrices. The proposed *RC, IC* and A^{Net} *matrix* uses following notations:

Notation	Short Definition
δ_{jr}^{+}	Change in distance of route *r* due to movement of a customer *j* to the best position in route *r*
δ_{j}^{-}	Change in distance due to removal of a customer from its current route
δ_{jr}^{Net}	Net change in distance of a solution due to movement of a customer *j* to route *r*
A^{Net}	Net cost matrix for increase in distance of solution

Fig. 5. Notations for *RC* and *IC* matrix

The term δ_{j}^{-} represents the decrease in the total distance if customer *j* is removed from its existing route. The term δ_{jr}^{+} represents the increase in distance of route *r* if customer *j* is removed from its original route and is reinserted in route *r* at its best position. The term δ_{jr}^{Net} represents change in distance of solution if customer *j* is removed from its original route and reinserted in route *r* at best position. The value of δ_{jr}^{Net} can be calculated by adding δ_{jr}^{+} and δ_{j}^{-} i.e. $\delta_{jr}^{Net} = \delta_{jr}^{+} + \delta_{j}^{-}$. In all these definitions, change is defined as difference between new

value and old value (i.e. new value – old value). The three matrices, *RC*, *IC* and A^{Net} , can be constructed to represent the increase or decrease of distance of all the customers in all routes (Figure 6). After generating *RC* and *IC* matrix, A^{Net} can be calculated. The cost of all insertion is stored in net cost matrix A^{Net}. For example cost of inserting the unvisited customer 4 to route 1 can be found at δ_{41}^{net} cell of matrix A^{Net}. Hence the net cost matrix A^{Net} can be scanned to find the best neighborhood insertion of unvisited customers.

$$IC = \begin{bmatrix} \delta_{11}^{+} & \delta_{12}^{+} & \cdots & \delta_{1r}^{+} \\ \delta_{21}^{+} & \delta_{22}^{+} & \cdots & \delta_{2r}^{+} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1}^{+} & \delta_{n2}^{+} & \cdots & \delta_{nr}^{+} \end{bmatrix} \qquad RC = \begin{bmatrix} \delta_{1}^{-} \\ \delta_{2}^{-} \\ \vdots \\ \delta_{n}^{-} \end{bmatrix}$$

$$A^{Net} = \begin{bmatrix} \delta_{11}^{net} & \delta_{12}^{net} & \cdots & \delta_{1r}^{net} \\ \delta_{21}^{net} & \delta_{22}^{net} & \cdots & \delta_{2r}^{net} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1}^{net} & \delta_{n2}^{net} & \cdots & \delta_{nr}^{net} \end{bmatrix}$$

Fig. 6. Development of net cost matrix

In the loop from line 7 to 20, all removed customers are inserted into routes based on A^{Net}. For the customers who cannot be inserted into any positions in the routes because of demand or route duration constraints, a new route is started (line 13 to 15). After inserting a removed customer, the route structure changes. Therefore, *RC* and *IC* of some remaining removed customers also changes. At the end of insertion loop, *RC* and *IC* of all influenced customers need to be updated (line 17 and 18). Consider inserting unselected customer 6 and 9, before insertion, original solution S_0:

$$S_0 = \begin{cases} D_1 - c_2 - c_4 - D_1 \\ D_1 - c_7 - D_1 \\ D_2 - c_5 - c_{10} - D_2 \\ D_2 - c_3 - c_1 - c_8 - D_2 \end{cases} \qquad NC = \{\boldsymbol{c_6}, \boldsymbol{c_9}\}$$

In S_0, there are 4 routes for depots D_1 and D_2. Though scanning net cost matrix $A^{Net,}$, δ_{61}^{net} is the least net cost within all net cost related to customer 6 and 9, which means inserting customer 6 into route 1 is the best insertion. Hence S_1 is generated:

$$S_1 = \begin{cases} D_1 - c_2 - \boldsymbol{c_6} - c_4 - D_1 \\ D_1 - c_7 - D_1 \\ D_2 - c_5 - c_{10} - D_2 \\ D_2 - c_3 - c_1 - c_8 - D_2 \end{cases} \qquad NC = \{\boldsymbol{c_9}\}$$

Inserting customer 6 into route 1 only influences the structure of route 1. So we need to update only relevant columns of factories in *IC* matrix. Similarly, in *RC* matrix, only the remove cost of customer 2, 4 and customer 6 need to updated (illustrate in Figure 7). Finally, net cost matrix (A^{Net}) is updated as per new values of *IC* and *RC* matrices.

$$IC = \begin{bmatrix} \delta_{11}^{+} & \delta_{12}^{+} & \delta_{13}^{+} & \delta_{14}^{+} \\ \delta_{21}^{+} & \delta_{22}^{+} & \delta_{23}^{+} & \delta_{24}^{+} \\ \vdots & & & \\ \vdots & & & \end{bmatrix} \qquad RC = \begin{bmatrix} \delta_1^{-} \\ \delta_2^{-} \\ \delta_3^{-} \\ \delta_4^{-} \\ \delta_5^{-} \\ \delta_6^{-} \\ \delta_7^{-} \\ \delta_8^{-} \end{bmatrix}$$

Fig. 7. Update *RC* and *IC* for influenced customers

Therefore, *RC* and *IC* matrices are only calculated once at the beginning and only some values are recalculated and updated during the insertion process. Besides, the calculated *RC* and *IC* matrix are further passed to and reused by the relocate local search and revised swap local search in the following steps, which dramatically improves the efficiency of whole algorithm. This reuse technique is also first time proposed in this thesis.

In perturbation scheme, after inserting all unselected customers, relocate local search is applied in the next step.

2.4.3.2. Relocate Local search

```
Relocate Local Search:
1.  Procedure RelocateLocalSearch (S, RC, IC) /* S is a perturbated solution, RC is remove cost and IC is insertion cost.*/
2.  Begin
3.    flag ← 0;
4.    While flag = 0
5.      NC ← ResetUnVisitedCustSet();/*Put all customers into NC */
6.      for i ← 1 to n do /* n is the element number of NC */
7.        CC ← Random(NC); /* Randomly choose one customer from NC as the current customer, CC */
8.        position ← 0; /* position records the best relocate position for CC which has the maximum distance decrease */
9.        position ← SelectTheBestRelocatePositionForCurrentCust(S, RC, IC, CC);
10.       if position ≠ 0 then /* A better position for CC is found */
11.         S′ ← ApplyTheRelocate(CC, position, S);
12.         UpdateRemoveCost(CC, S′, RC); /* Update remove cost for influenced customers */
13.         UpdateInsertCost(CC, S′, IC); /* Update insert cost for influenced customers */
14.       end if
15.       RemoveTheCurrentCustFromNC(CC, NC);
16.     end for
17.     if Distance(S) = Distance(S′) then
18.       flag = 1; /* if no better solution found, then stop the revised relocate local search */
19.     end if
20.   end While
21.   Return S′;
22. End
```

Fig. 8. Pseudo code of Relocate Local Search

In relocate local search, each customer will be relocated to the best positions based on the maximum total routes distance decrease value. Figure 8 illustrates the pseudo code of relocate local search. The main structure of relocate local search is similar as the insertion part in the perturbation scheme but there are also some differences. In relocate local search:

1) All customers instead of a random number of customers are put into the unvisited customer set *NC* (line 5).

2) When choosing the current customer *CC* from *NC* to relocate, *CC* is randomly selected instead of selecting and relocating the customer who has the overall maximum negative

net cost value (line 7).

3) Relocate local search may be applied more than once until the solution keeps on improving (line 18, the variable flag controls whether stop the relocate local search).

The solution obtained after the relocate local search goes through revised swap local search.

2.4.3.3. Revised swap Local search

```
Revised Swap Local Search:
1.  Procedure RevisedSwapLocalSearch (S, RC, IC) /* S is a relocated solution, a set of MDVRP routes. */
2.  Begin
3.    flag ← 0;
4.    While flag = 0
5.      NC ← ResetUnVisitedCustSet();
6.      for i ← 1 to n do /* n is the element number of NC */
7.        CC ← Random(NC); /* Randomly choose one customer from NC as the current customer, CC */
8.        SC ← 0; /* Variable SC is the selected swap customer */
9.        SC ← SelectTheSwapCust(S, RC, IC, CC); /* Select SC based on the maximum total distance decrease */
10.       if SC ≠ 0 then /* A feasible swap customer is found */
11.         S′ ← ApplyTheSwap(CC, SC, S); /* Swap the CC and SC */
12.         UpdateRemoveCost(CC, SC, S′, RC); /* Update Remove Cost for influenced customers */
13.         UpdateInsertCost(CC, SC, S′, IC); /* Update insert Cost for influenced customers */
14.       end if
15.       RemoveTheCurrentCustFromNC(CC, NC);
16.     end for
17.     if Distance(S) = Distance(S′) then
18.       flag = 1; /* if no feasible swap found, then stop the revised swap local search */
19.     end if
20.   end While
21.   Return S′;
22. End
```

Fig. 9. Pseudo code of Revised Swap Local Search

Revised swap local search is a new local search technique proposed in this thesis. In the classic swap local search, two selected customers exchanges their positions. However, in the revised swap local search, two selected customers are exchanged their routes and they will be inserted into the best positions in the new routes. Based on the current customer (*CC*), the Swap

Customer (*SC*) is selected based on the *RC*, *IC* and A^{Net} matrix.

Figure 9 shows the pseudo code of revised swap local search. Similar as the insertion part of perturbation scheme and relocate local search, in revised swap local search, after swap *CC* and *SC*, *RC* and *IC* of influenced customers are need to be updated (line 12 and 13). Revised swap local search is also applied until no better solution can be found.

Figure 10 demonstrates the difference between the classic swap local search and revised swap local search. In this example, c_{10} and c_9 are selected to apply swap local search. In classic swap local search, these two customers will be exchanged their positions and the total distance reduces to 2770 from 2890. However, in the revised swap local search, these two customers only exchange their routes, and they are inserted into the best positions on the new routes. Finally, the total distance reduces to 2650.

Classic Swap Local Search

$$S_0 = \begin{cases} D_1 - c_2 - c_4 - D_1 \\ D_1 - c_7 - \boldsymbol{c_{10}} - D_1 \\ D_2 - c_5 - D_2 \\ D_2 - c_3 - c_1 - c_8 - D_2 \\ D_3 - c_6 - \boldsymbol{c_9} - D_3 \end{cases}$$

Total distance: 2890

$$S_1 = \begin{cases} D_1 - c_2 - c_4 - D_1 \\ D_1 - c_7 - \boldsymbol{c_9} - D_1 \\ D_2 - c_5 - D_2 \\ D_2 - c_3 - c_1 - c_8 - D_2 \\ D_3 - c_6 - \boldsymbol{c_{10}} - D_3 \end{cases}$$

Total distance: 2770

Revised Swap Local Search

$$S_0 = \begin{cases} D_1 - c_2 - c_4 - D_1 \\ D_1 - c_7 - \boldsymbol{c_{10}} - D_1 \\ D_2 - c_5 - D_2 \\ D_2 - c_3 - c_1 - c_8 - D_2 \\ D_3 - c_6 - \boldsymbol{c_9} - D_3 \end{cases}$$

Total distance: 2890

$$S_1 = \begin{cases} D_1 - c_2 - c_4 - D_1 \\ D_1 - c_7 - \boldsymbol{c_9} - D_1 \\ D_2 - c_5 - D_2 \\ D_2 - c_3 - c_1 - c_8 - D_2 \\ D_3 - \boldsymbol{c_{10}} - c_6 - D_3 \end{cases}$$

Total distance: 2650

Fig. 10. Comparison between Classic Swap Local Search and Revised Swap Local Search

2.4.4. Update the Trail Intensity.

At the end of each iteration, trail intensities of *depot-ants* τ_{ik}^{a} and trail intensities of *route-ants* τ_{ij}^{t} are updated by the best solution found so far. The trail intensities of *depot-ants* are updated as follows:

$$\tau_{ik}^{a^{new}} = \alpha * \tau_{ik}^{a^{old}} + \Delta\tau_{ik}, \quad \forall i \in C, k \in D \tag{2.14}$$

Where,

$$\Delta\tau_{ik} = \begin{cases} \frac{1}{l_{ik}}, if\ customer\ i\ \ is\ assigned\ to\ depot\ k\ in\ the\ best\ solution. \\ 0, otherwise. \end{cases}$$

Here, α is the trail persistence level and l_{ik} is the distance between customer i and depot k. Similarly, route trail intensities are updated as follows:

$$\tau_{ij}^{t^{new}} = \beta * \tau_{ij}^{t^{old}} + \Delta\tau_{ij}, \quad \forall i, j \in V, i \neq j \tag{2.15}$$

Where,

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{l_{ij}}, if\ node\ i\ \ and\ j\ are\ connected\ in\ the\ best\ solution. \\ \tau_{ij}^{t^{new}}, otherwise. \end{cases}$$

In this way, the trail intensities are updated at the end of each iteration, which guides the artificial ants to find better solutions in the next iteration.

2.5. Numerical Experiments and Analysis.

There are 23 benchmark instances for MDVRP extracted from previous literature. All these 23 benchmark instance and best-known solutions are available at http://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/. The characteristics of these 23

instances are listed in Table 1. In the table, n, m, k, R, Q represent the number of customers, number of depots, number of vehicles for each depot, route duration and load capacity for each depot.

Table 1. MDVRP Instance Characteristics

Instance	n	m	k	R	Q
1	50	4	4	∞	80
2	50	4	2	∞	160
3	75	5	3	∞	140
4	100	2	8	∞	100
5	100	2	5	∞	200
6	100	3	6	∞	100
7	100	4	4	∞	100
8	249	2	14	310	500
9	249	3	12	310	500
10	249	4	8	310	500
11	249	5	6	310	500
12	80	2	5	∞	60
13	80	2	5	200	60
14	80	2	5	180	60
15	160	4	5	∞	60
16	160	4	5	200	60
17	160	4	5	180	60
18	240	6	5	∞	60
19	240	6	5	200	60
20	240	6	5	180	60
21	360	9	5	∞	60
22	360	9	5	200	60
23	360	9	5	180	60

For ACSVNS, we set the number of iteration A as 1000, number of artificial ants y as 20, trail persistence for *depot-ants* α, trail persistence for *route-ants* β as 0.95, number of perturbation local search for ant solution nl_1 is 15 and number of perturbation local search for the best solution so far nl_2 is 10. For numerical experiments, the ACSVNS algorithms were coded on C++ and implemented them on AMD Opteron 2.3 GHz with 16GB of RAM.

The proposed ACSVNS is compared with some algorithms which have an excellent performance in solving MDVRP in literature. They are FIND algorithm (Renaud et al, 1996), CGL method (Cordeau et al, 1997), a standard ACO (Dorigo et al, 1996), ACO with the ant-weight strategy and the mutation operation (ACO-WM) (Yu et al, 2009), PIACO method (Yu et al, 2011), HGSADC algorithm (Vidal et al., 2012) and ELTG (Escobar et al., 2014).

Relative Percentage Deviation (RPD) is used to evaluate the performance of ACSVNS and other algorithms. For j^{th} instance and i^{th} algorithm, the RPD_{ij} is calculated based on the following equation:

$$RPD_{ij} = \frac{(H_j^i - B_j)}{B_j} \times 100\% \qquad 2.16$$

In this formula, H_j^i stands for the solution obtained by the i^{th} algorithm for the j^{th} instance and B_j represents the best solution found so far in literature for the j^{th} problem instance. The experiment results and related RPD of different algorithms are provided in table 2. Additionally, as mentioned in the paper of Yu et al. (2011), the run time is not only dependent on the CPU of the computers, but also on the operation system, compiler, programming language and the precision used during the execution of the run. Therefore the comparison of CPU time is not included.

From Table 2, it can be seen that HGSADC has the best performance on average RPD and it is followed by ELTG, PIACO, CGL, ACSVNS, FIND, ACO-WM and ACO. However, ACSVNS can match the existing best known solution for 16 instances, which is the third best in all algorithms. Additionally, ACSVNS found new best solutions for the instance 5, 6 and 7. The experiment results show that ACSVNS has the best performance on solving small and medium-sized MDVRP instances but when solving large-sized instances, the performance is

unstable. For the instance 23, the largest instance, which considers 360 customers and 9 depots, only ACSVNS and HGSADC found the best solution in literature. However, for the instance 10, the RPD of ACSVNS is the largest in these 8 algorithms.

2.6. Conclusion.

In this chapter, Ant Colony System with Variable Neighborhood Search algorithm (ACSVNS) is proposed to solve MDVRP. In ACSVNS, two types of ants, depot-ants and route-ants are used to generate solution. Besides, perturbation scheme, relocate and revised swap local search are applied to improve solution quality. All these neighborhood search techniques are based on Remove Cost (*RC*) matrix and Insertion Cost (*IC*) matrix. In ACSVNS, for each ant, *RC* and *IC* matrices are only calculated once and they are passed and reused during the whole neighborhood search process. Except this reuse technique, the revised swap local search is also first time proposed in the VRP literature. Compared with the classic swap local search, this revised swap local search is effective and efficient because in the revised swap local search, two selected customers exchange their routes instead of positions. After testing the 23 MDVRP benchmark instances, ACSVNS can get the best solution for 16 instances and also get a new best solution for 3 instances, which proves that ACSVNS is an excellent algorithm to solve MDVRP.

Table 2. The results of ACSVNS compared with other heuristic methods

Ins	Best-known	FIND		CGL		ACO		ACO-WM		PIACO		HGSADC		ELTG		ACSVNS	
	solutions	Best	RPD	Best	RPD	Best	RPD	Best	RPD	Best	RPD	Best	RPD	Best	RPD	Best	RPD
1	**576.86**	**576.86**	0.00	**576.86**	0.00	**576.86**	0.00	**576.86**	0.00	**576.86**	0.00	**576.86**	0.00	**576.86**	0.00	**576.86**	0.00
2	**473.53**	**473.53**	0.00	**473.53**	0.00	484.28	2.27	**473.53**	0.00	**473.53**	0.00	**473.53**	0.00	**473.53**	0.00	**473.53**	0.00
3	**641.18**	**641.18**	0.00	645.15	0.62	645.15	0.62	**641.18**	0.00	**641.18**	0.00	**641.18**	0.00	**641.18**	0.00	**641.18**	0.00
4	**1001.04**	1003.86	0.28	1006.66	0.56	1020.52	1.95	1001.49	0.04	1001.49	0.04	**1001.04**	0.00	**1001.04**	0.00	1002.06	0.10
5	750.03	750.26	0.03	753.4	0.45	750.26	0.03	750.26	0.03	750.26	0.03	750.03	0.00	750.03	0.00	**744.826**	-0.69
6	876.5	876.5	0.00	877.84	0.15	878.34	0.21	876.5	0.00	876.5	0.00	876.5	0.00	876.5	0.00	**876.395**	-0.01
7	881.97	892.58	1.20	891.95	1.13	898.8	1.91	887.11	0.58	885.69	0.42	881.97	0.00	884.66	0.30	**879.785**	-0.25
8	**4371.66**	4485.08	2.59	4482.44	2.53	4508.14	3.12	4500.15	2.94	4482.38	2.53	4372.78	0.03	**4371.66**	0.00	4469.48	2.24
9	**3858.66**	3937.81	2.05	3920.85	1.61	4083.44	5.83	3913	1.41	3912.23	1.39	**3858.66**	0.00	3880.55	0.57	3963.29	2.71
10	**3629.6**	3669.38	1.10	3714.65	2.34	3747.62	3.25	3693.4	1.76	**3663**	0.92	3631.11	0.04	**3629.6**	0.00	3759.88	3.59
11	**3545.48**	3648.94	2.92	3580.84	1.00	3599.93	1.54	3564.74	0.54	3554.08	0.24	3546.06	0.02	**3545.48**	0.00	3669.96	3.51
12	**1318.95**	**1318.95**	0.00	**1318.95**	0.00	1327	0.61	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00
13	**1318.95**	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00	**1318.95**	0.00
14	**1360.12**	1365.68	0.41	**1360.12**	0.00	1375.22	1.11	1373.18	0.96	1365.68	0.41	**1360.12**	0.00	**1360.12**	0.00	**1360.12**	0.00
15	**2505.29**	2551.45	1.84	2534.13	1.15	2588.22	3.31	2565.67	2.41	2551.45	1.84	**2505.29**	0.00	**2505.29**	0.00	**2505.29**	0.00
16	**2572.23**	**2572.23**	0.00	**2572.23**	0.00	2604.9	1.27	**2572.23**	0.00	**2572.23**	0.00	**2572.23**	0.00	**2572.23**	0.00	**2572.23**	0.00
17	**2708.99**	2731.37	0.83	2720.23	0.41	2776.99	2.51	**2708.99**	0.00	**2708.99**	0.00	**2708.99**	0.00	**2708.99**	0.00	**2708.99**	0.00
18	**3702.75**	3781.03	2.11	3710.49	0.21	3907.88	5.54	3846.05	3.87	3781.03	2.11	**3702.75**	0.00	**3702.75**	0.00	3765.49	1.69
19	**3827.06**	**3827.06**	0.00	**3827.06**	0.00	3863.03	0.94	**3827.06**	0.00	**3827.06**	0.00	**3827.06**	0.00	**3827.06**	0.00	**3827.06**	0.00
20	**4058**	4097.06	0.96	**4058**	0.00	4231.28	4.27	4142	2.07	4097.06	0.96	**4058**	0.00	**4058**	0.00	**4058**	0.00
21	**5474.74**	5656.46	3.32	5535.99	1.12	5579.86	1.92	5495.54	0.38	**5474.74**	0.00	**5474.74**	0.00	**5474.74**	0.00	5590.44	2.11
22	**5702.06**	5718	0.28	5716.01	0.24	5897.64	3.43	5832.07	2.28	5772.23	1.23	**5702.06**	0.00	**5702.06**	0.00	**5702.06**	0.00
23	**6078.75**	6145.58	1.10	6139.73	1.00	6341.61	4.32	6183.13	1.72	6125.58	0.77	**6078.75**	0.00	6095.46	0.27	**6078.75**	0.00
Average			0.91		0.63		2.17		0.91		0.56		0.00		0.05		0.65

CHAPTER 3

ANT COLONY ALGORITHM FOR A MULTI-DEPOT GREEN VEHICLE ROUTING PROBLEM

A Multi-depot Green Vehicle Routing Problem (MDGVRP) is considered in this chapter. A Two-stage Ant Colony System (TSACS) algorithm is proposed to find solutions for this problem. Same as the ACSVNS in chapter 2, TSACS also uses two types of ants, *depot ant* and *route ant* to generate solution. The solution for MDGVRP is useful for companies which employ the Alternative Fuel-Powered Vehicles (AFVs) to deal with the obstacles brought by the limited number of the Alternative Fuel Stations (AFSs). This chapter adds fuel tank capacity constraints to make it more meaningful and closer to real-world case. The numerical experiment is performed on randomly generated problem instances to evaluate the performance of proposed algorithms.

3.1. Introduction

The green logistics is attracting people's attention because of the growing environmental concern worldwide. The issue of green logistics arises because the current practice on production, processing and distribution system have triggered various environmental problems.

Under this background, Green Vehicle Routing problem (GVRP) has attracted the attention of researchers (Erdoğan & Miller-Hooks, 2012; Lin et al., 2014; Schneider et al., 2014). The classical Vehicle Routing Problem (VRP) focuses on minimizing the total transportation cost generated in the process of distribution services, while the GVRP emphasizes on addressing environmentally sustainable issues in the delivery distribution of

supply chains along with the optimal economic cost of delivery (Lin et al., 2014). According to the research of Erdoğan and Miller-Hooks (2012), the design of GVRP requires the use of the Alternative Fuel-powered Vehicles (AFV), which relies on greener fuel source such as electricity, natural gas, hydrogen, etc. However, Erdoğan and Miller-Hooks (2012) pointed out two main obstacles encountered while replacing the conventional vehicles with the AFVs: 1) the limited capacity of fuel tank or batteries of AFVs, and 2) the scarcity of Alternative Fuel Stations (AFSs). These obstacles make the problem formulation and algorithm design of GVRP more complex than those of VRP.

The Multi-Depot Vehicle Routing Problem (MDVRP) is another branch of VRP which has also attracted tons of attention of researchers and practitioners (Montoya-Torres et al., 2015; Vidal et al., 2012; Yu et al., 2011). In the MDVRP, the fleet of vehicles serve customers from several depots and returns to the same depot. Research about the MDVRP is meaningful for companies that have a wide range of business establishments and have more than one depot.

In recent years, many large-scale multinational companies such as UPS, Coca-Cola and GM have especially paid attention to their environmentally sustainable performances (Farneti & Guthrie, 2009; Morhardt, Baird, & Freeman, 2002). They are exhausting their ability to keep balance between economic performance and environment protection. For these companies, the design of transportation routes based on GVRP or MDVRP may not provide the desired optimal solution. Therefore, in this thesis, a new variety of problem called the Multi-Depot Green Vehicle Routing Problem (MDGVRP), is addressed. In MDGVRP, the AFVs start from different depots, serve customers, and at the end return to the original depots. Due to the limited capacity of fuel tank of AFVs and the scarcity of AFSs, each AFV is required to visit AFS for

refueling. The objective of MDGVRP is to minimize the route distance of the AFV fleets. Thus, compared with MDVRP or GVRP, MDGVRP has more constraints and subsequently, is different to formulate and solve. An example of MDGVRP is as follows: it is assumed that there is one logistic company using AFVs to deliver goods to its five customers $(C_1, C_2, C_3, C_4, C_5)$. The company assigns its AFVs from two depots (D_1, D_2) and these AFVs can refuel at depots or AFSs (F_1, F_2). The tank capacity $T\ (gallons)$ of an AFV is 60 gallons and it is assumed that there is a linear relationship between transportation distance and fuel consumption with a consumption rate $r\ (gallons/mile)$ equaling to 0.2 (Fraer, Dinh, Chandler, & Buchholz, 2005). The objective of this company is to design the minimum-distanced distribution routes of all AFV fleet. During the traveling process, if the remaining fuel is not enough to reach the next node, the AFV has to visit AFS to refuel. After the whole visiting process, all customers have to be visited once and only once. The following Figure 11 illustrates the coordinates of these customers, depots and the AFSs and Table 3 and 4 provide the distance and fuel consumption between these vertices, respectively.

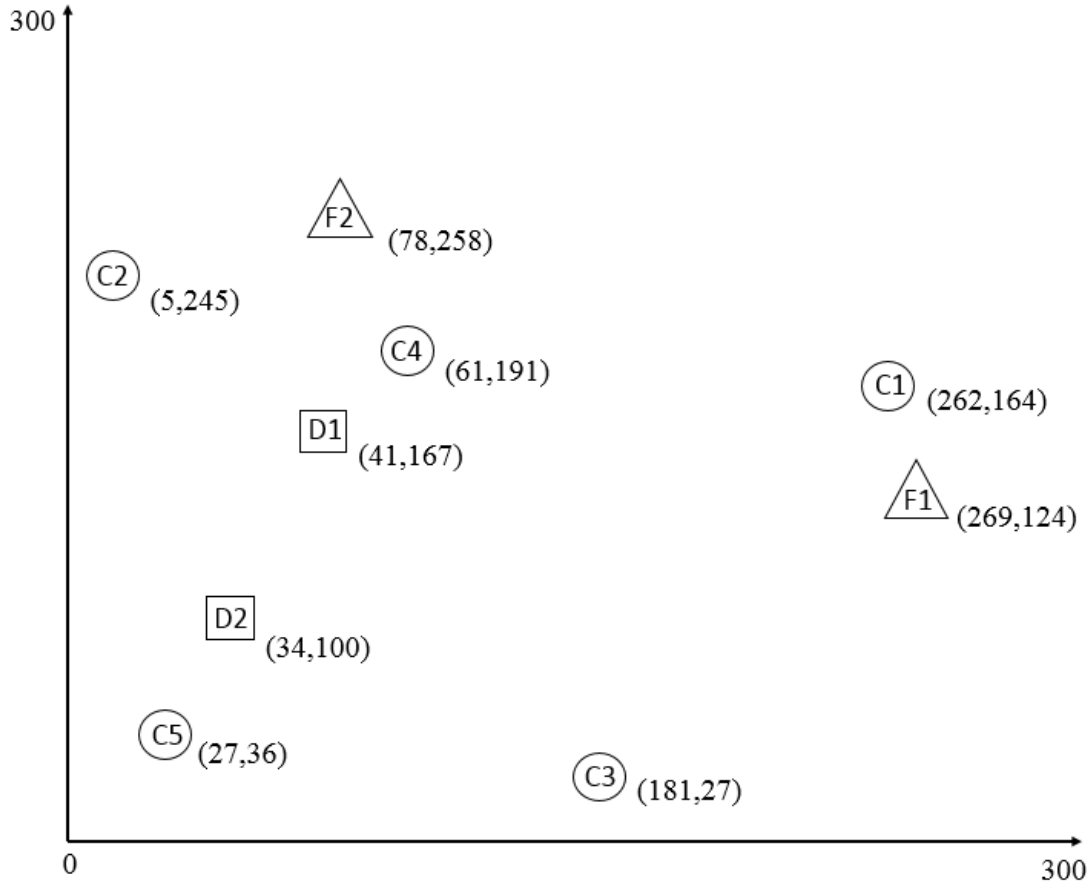

Fig.11. Positions of the vertices in MDGVRP

Table 3

Distance between vertices.

	D1	D2	C1	C2	C3	C4	C5	F1	F2
D1	0.00	67.36	221.02	85.90	197.99	31.24	131.75	232.02	98.23
D2	-	0.00	236.80	147.87	164.13	94.92	64.38	236.22	164.01
C1	-	-	0.00	269.46	159.15	202.81	267.59	40.61	206.62
C2	-	-	-	0.00	280.18	77.79	210.16	290.41	74.15
C3	-	-	-	-	0.00	203.21	154.26	130.97	252.92
C4	-	-	-	-	-	0.00	158.69	218.53	69.12
C5	-	-	-	-	-	-	0.00	257.50	227.78
F1	-	-	-	-	-	-	-	0.00	233.32
F2	-	-	-	-	-	-	-	-	0.00

Table 4

Fuel consumption between vertices.

	D1	D2	C1	C2	C3	C4	C5	F1	F2
D1	0.00	13.47	44.20	17.18	39.60	6.25	26.35	46.40	19.65
D2	-	0.00	47.36	29.57	32.83	18.98	12.88	47.24	32.80
C1	-	-	0.00	53.89	31.83	40.56	53.52	8.12	41.32
C2	-	-	-	0.00	56.04	15.56	42.03	58.08	14.83

C3	-	-	-	-	0.00	40.64	30.85	26.19	50.58
C4	-	-	-	-	-	0.00	31.74	43.71	13.82
C5	-	-	-	-	-	-	0.00	51.50	45.56
F1	-	-	-	-	-	-	-	0.00	46.66
F2	-	-	-	-	-	-	-	-	0.00

The following Figure 12 illustrates a feasible solution for this MDGVRP. In this solution, this company needs to assign one AFV from D_1 to deliver goods for C_4 and C_2. Besides, there are also 2 AFVs needed to be assigned from D_2 to serves C_5, C_1 and C_3. It is needed to notice that for the AFV which visits C_1 and C_3, it has to refuel in the F_1 after serving customers to get C_1 and go back to its original depot D_2.

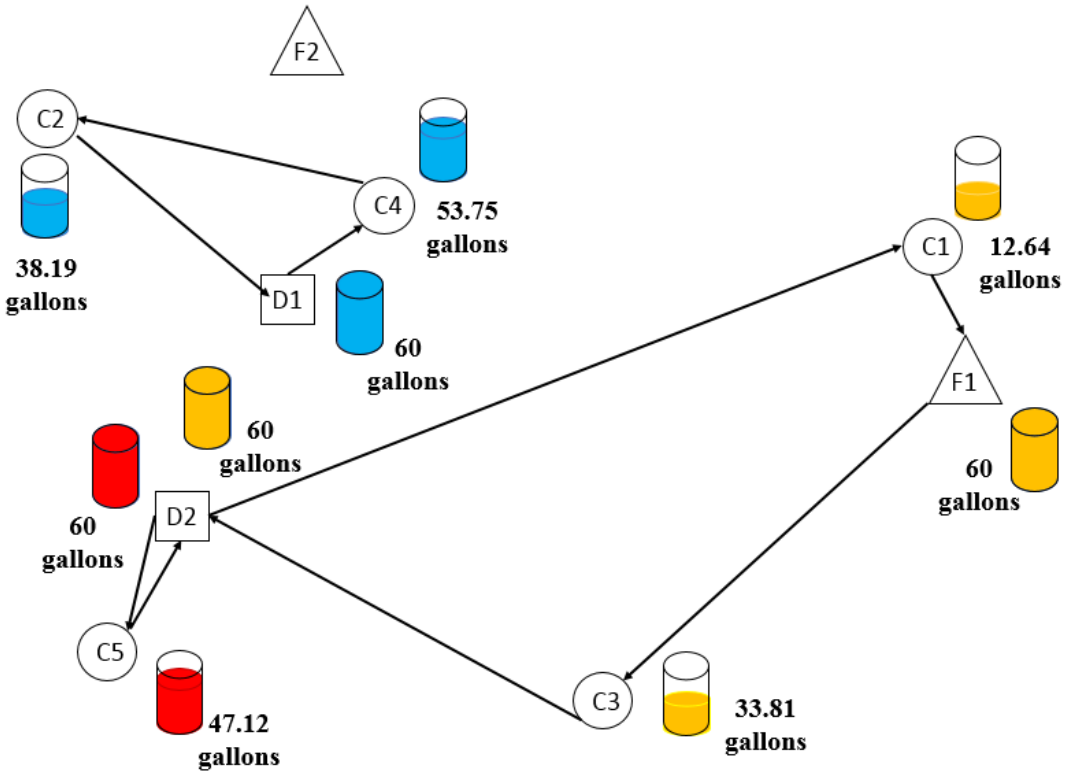

Fig.12. A feasible solution to the MDGVRP.

Therefore, the distribution routes for these three AFVs are:

For AFV1: $D_1 - C_4 - C_2 - D_1$

For AFV2: $D_2 - C_5 - D_2$

For AFV3: $D_2 - C_1 - F_1 - C_3 - D_2$

Total transportation distance: 896.223 miles.

MDGVRP is an NP-hard problem because it turns to MDVRP when fuel tank capacity is set to be unlimited. For NP-hard problem, finding the optimal solution is hard. Therefore, in this thesis, revised Partition Based Algorithm (PBA) and Two-stage Ant Colony System (TSACS) algorithm are proposed to solve MDGVRP. Similar to the ACSVNS proposed in chapter 2, TSACS also use of two types of ants for two different purposes. The first type of ant is used to assign customers to depots while the second type of ant is used to find the routes.

The structure of the rest of this chapter is organized as follows. Chapter 3.2 reviews the related literature. Chapter 3.3 introduces the definition and formulation of MDGVRP. Chapter 3.4 presents the algorithms specially designed for solving MDGVRP. Numerical experiments are presented in chapter 3.5 and are followed by the conclusion in chapter 3.6.

3.2. Literature Review

MDGVRP is a new variant of VRP problem, therefore, there is no literature focusing on this problem. The MDGVRP is based on the GVRP and MDVRP. Literature on MDVRP is reviewed in chapter 2. Therefore, some important previous studies in GVRP are reviewed in the following section.

Although the research on GVRP just began about 10 years ago, this problem has received extensive attention from researchers because of the awareness of people on the importance of environment protection. According to the comprehensive literature survey on the GVRP of Lin et al (2014), there are two categories of GVRP: Pollution Routing Problem (PRP) and Green-

VRP. Although these categories focus on economic cost and environment cost simultaneously, the PRP reduces environment cost by minimizing the fuel consumption or minimizing the Green House Gas (GHG) emissions, while the Green-VRP alleviates the environment damage by using AFVs instead of conventional vehicles. Erdoğan and Miller-Hooks (2012) first addressed the AFVs based GVRP. They proposed a model to optimize the transportation routes in order to overcome the obstacles caused by the limited fuel tank capacity of the AFVs. Based on their work, Felipe, Ortuño, Righini and Tirado (2014) proposed a heuristic approach to solve GVRP with multiple technologies and partial recharges, Montoya, Gueret, Mendoza and Villegas (2016) developed a multi-space sampling heuristic for GVRP, and Koç and Karaoglan (2016) solved GVRP with an exact algorithm based on branch and bound. Schneider et al. (2014) broadened the boundary of GVRP by adding the customer time window constraints to the VRP for electric vehicles.

Although the work of Kaabachi et al. (2017), Jabir et al.(2017) also focus on MDGVRP, their MDGVRP model is developed from Pollution Routing Problem (PRP) for minimizing the total Green House Gas (GHG) emission. They did not consider tank capacity of vehicles. Our model is developed from Green-VRP (G-VRP) which minimizes the total travelling distance and considers tank capacity of AFVs. The MDGVRP considered in this thesis is based on the Green-VRP addressed by Erdoğan and Miller-Hooks (2012). Compared with their model, our model considers the demands of customers and can be used to solve the multi depots problem instead of a single depot problem.

Based on the idea of Nagy and Salhi (2005), a revised Partition Based Algorithm (PBA) is proposed in this thesis. We also proposed Two-stage Ant Colony System (TSACS) algorithm

to obtain high-quality solutions.

3.3. Problem Description and Formulation

A standard MDGVRP can be described as the problem of designing least distance routes from more than one depot to a set of customers. Each customer is associated with a non-negative demand to be delivered. Each customer is visited by the vehicle fleet once and the demand of customer is satisfied after this visit. A vehicle starts from a depot, serves customers one by one, and finally returns to its original depot. During the service process, when the remaining cargo of a vehicle cannot satisfy the demand of the next customer, the vehicle returns to depots for reloading cargo. If it is necessary to refuel during the service process, the vehicles visit the AFSs to refuel. Therefore, a particular AFS can be visited by more than once. The objective of the problem is to minimize the total distance travelled by all vehicles.

Let $G = (V, E)$ be a complete and directed graph, in which V is a set of vertices and E is a set of edges between different vertices. The vertex set V has three subsets: customer set $C = \{c_{1,}\, c_{2,} \ldots\, c_N\}$ for N customers, depot set $D = \{c_{N+1,}\, c_{N+2,} \ldots\, c_{N+M}\}$ for M depots and AFS set $S = \{c_{N+M+1,}\, c_{N+M+2,} \ldots\, c_{N+M+NS}\}$ for NS AFSs. Therefore, the number of customers, depots and AFSs are N, M and NS, respectively. The edge set $E = \{(c_i, c_j) : c_i, c_j \in V, i \neq j\}$ represents the edges connecting different vertices of V and every element of E is associated with the distance between two vertices d_{ij} and fuel consumption f_{ij}. In MDGVRP, we assume that the relationship between travel distance and fuel consumption is linear. The fuel consumption rate r is assumed to be a fixed value. Therefore, the fuel consumption in arc (c_i, c_j) is calculated as $f_{ij} = r \cdot d_{ij}$.

All other necessary notations which are used to formulate MDGVRP are defined as follows:

K The total number of vehicles

r Fuel consumption rate (gallons per mile)

T Tank capacity of vehicle (gallons)

Q Load capacity of vehicle

q_i The demand of vertex i

d_{ij} The distance between vertices i and j

Decision Variables

x_{ijk} is a binary variable taking value 1 if vehicle travels directly from vertex i to j by the k^{th} vehicle. Otherwise, x_{ijk} equals 0.

f_i The remaining fuel level after visiting vertex i

l_i The remaining cargo level after visiting vertex i

For every AFS, n_f $(f = 1, 2, \ldots NS)$ is the number of dummy vertices. In this way, the number of visiting times for every AFS can be recorded by n_f. In the research of Bard, Huang, Dror, and Jaillet (1998), there are more details and techniques about dummy vertex.

C The set of customers

D The set of depots

S The set of AFSs

$DS = D \cup S$ The set contains all depots and AFSs.

$CS = C \cup S$ The set contains all customers and AFSs.

$V = C \cup D \cup S$ The set contains all customers, depots and AFSs vertices

The mathematical formulation for MDGVRP is as follows;

$$\min \sum_{i,j \in V} \sum_{k \in K} d_{ij} x_{ijk} \tag{3.1}$$

Subject to

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} = 1, \quad \forall\, i \in C \tag{3.2}$$

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} = 1, \quad \forall\, j \in C \tag{3.3}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0, \quad \forall\, h \in V,\, k \in K \tag{3.4}$$

$$\sum_{i \in D} \sum_{j \in V} x_{ijk} \leq 1, \quad \forall\, k \in K \tag{3.5}$$

$$l_j \leq l_i - q_j \cdot \sum_{k \in K} x_{ijk} + Q \cdot \left(1 - \sum_{k \in K} x_{ijk}\right), \quad \forall\, i \in CS, j \in CS \tag{3.6}$$

$$0 \leq l_i \leq Q - q_i, \quad \forall\, i \in V \tag{3.7}$$

$$f_j \leq f_i - r \cdot d_{ij} \cdot \sum_{k \in K} x_{ijk} + T \cdot \left(1 - \sum_{k \in K} x_{ijk}\right), \quad \forall\, i, j \in C \tag{3.8}$$

$$f_j \leq T - r \cdot d_{ij} \cdot \sum_{k \in K} x_{ijk}, \quad \forall\, i \in DS, j \in C \tag{3.9}$$

$$f_i \geq r \cdot d_{ij} \cdot \sum_{k \in K} x_{ijk}, \quad \forall\, i, j \in V \tag{3.10}$$

$$x_{ijk} \in \{0,1\}, \quad \forall\, i, j \in V, k \in K \tag{3.11}$$

In this formulation, equation 3.1 is the objective function which minimizes the total route distance of all vehicles. Constraints 3.2 and 3.3 restrict that each customer can only be visited by one vehicle and can only be visited once. Constraint 3.4 ensures that for each vehicle, the number of outgoing arcs from one vertex is equal to the incoming arcs to this vertex. Constraint 3.5 means that for any vehicle k, it can only leave a depot not more than once. Constraints 3.4 and 3.5 combine together to ensure that the starting and ending depot of a vehicle is the same. Equations 3.6 and 3.7 ensure that the vehicle capacity constraint is preserved. Constraints 3.8, 3.9 and 3.10 guarantee that vehicle tank capacity constraint T is satisfied. In this formulation, the constraint 3.8, 3.9 and 3.10 distinguish the MDGVRP from MDVRP. These constraints are completely different from other formulation because of the specific property of MDGVRP. Constraint 3.8 calculates fuel consumption if vehicle travels between two customers. Constraint 3.9 calculates fuel consumption from depot or fuel station to customer. The constraint 3.10 ensures that the remaining fuel is enough to go to its next destination. Finally, the constraint 3.11 limits x_{ijk} as a binary variable.

3.4. Solution of MDGVRP

In this section, two methods are designed to solve MDGVRP. Inspired by the PBA proposed by Nagy and Salhi (2005) which is used to solve MDVRP problem, a revised Partition Based Algorithm (PBA) is proposed to solve MDGVRP. In addition, a Two-stage Ant Colony System (TSACS) algorithm is also proposed for MDGVRP. The details of these two algorithms are provided in the following sections.

3.4.1. Partition Based Algorithm (PBA)

Based on the idea of dividing customers into borderline and non-borderline customers, the Partition Based Algorithm (PBA) is proposed by Nagy and Salhi (2005) to solve MDVRP. In this thesis, we extend PBA for solving MDGVRP. A borderline customer is a customer which is situated approximately halfway between two depots. Consider customer i with nearest depot p and second nearest depot q. Let d_{ip} and d_{iq} be the distance between customer i and depot p, and the distance between customer i and depot q, respectively. Customer i is identified as a borderline customer if $\frac{d_{iq}}{d_{ip}} \geq r_a$ (r_a is a parameter between 0.5 and 1). Otherwise, customer i is considered as a non-borderline customer. The process of applying PBA to solve MDGVRP is as follows:

Step 1. Divide all customers into borderline and non-borderline customers.

Step 2. Assign all non-borderline customers to their nearest depots.

Step 3. Generate GVRP route for each depot and associated non-borderline customers are as follows.

Step3.1. Generate TSP routes based on nearest neighbor criteria NNC (Gutin, Yeo, &

Zverovich, 2002) for non-borderline customers associated with depot.

Step 3.2. Generate GVRP routes from the TSP routes.

Step 4. Insert the borderline customers into one of GVRP routes based on cheapest insertion criteria. While inserting borderline customers, it should be ensured that the new GVRP route should satisfy all constraints for MDGVRP. It is possible that some of the borderline customers cannot be inserted into any GVRP routes. In this case, create another trip for customers starting from their nearest depot and returning to the same depot either directly or through the visit of AFSs.

The step 3 of the algorithm is the most difficult part of PBA. Firstly, in step 3.1, TSP routes are generated for each depot based on the NNC. In NNC, the nearest node is selected as the next node on route for visit until all nodes have been selected. In step 3.2, GVRP route from TSP is obtained in two phases. In the first phase VRP route is generated from the TSP route and in the second phase GVRP route is generated from the VRP route. Zhang, Gajpal and Appadoo (2017) generate GVRP routes from TSP routes in two separated steps. This thesis extends the work of Zhang et al. (2017) to include more feasible GVRP solutions. Specifically, our GVRP generating process not only insert fuel stations, but also insert depots. The details to generate GVRP route from TSP route are given below.

Phase 1: Generate VRP route from TSP route.

VRP routes are generated before GVRP routes. VRP routes are generated by inserting depots into the TSP routes based on customer demand. The VRP routes start from depots with full cargo load and visits the customers in the same order as they appear in TSP routes. However, when the remaining cargo of the vehicle cannot satisfy the demand of next customer (NC_k), a

depot is inserted in the TSP route. In this way, VRP routes for all depots can be generated.

Phase 2: Generate GVRP route from VRP route.

Generating GVRP routes is the most challenging part for generating the solutions for MDGVRP. Generating GVRP routes need to insert AFSs or depot into the VRP routes based on the fuel consumption between the nodes in the VRP routes. The process of generating the GVRP route based on the VRP route for one depot k is shown in the flow chart Figure 13.

In GVRP route for depot k, when the remaining fuel level (f) of the vehicle is not enough to get the next node, this vehicle is required to visit an AFS or go back to depot k to refuel. Therefore, in this situation, an AFS or a depot is inserted into the VRP route for depot k. Although Figure 13 illustrates the whole process of inserting AFS or depot in different situations, some important techniques need to be emphasized.

The AFS is inserted only when all of the following conditions are satisfied:

1. The remaining fuel level of the vehicle is sufficient to reach the AFS.
2. The fuel consumption between the AFS and depot k is less than tank capacity T.
3. The distance travelled to visit the next customer (NC_k) through the AFS is shorter than the distance travelled to visit the next customer (NC_k) through the depot k.

The depot k is inserted only when either condition 1 and 2 or condition 3 is satisfied:

Either:

1. The remaining fuel level of the vehicle is sufficient to reach the depot k.
2. The distance travelled to visit the next customer (NC_k) through the depot k is shorter than the distance travelled to visit the next customer (NC_k) through the AFS.

Or:

3. The remaining fuel level of the vehicle is not sufficient to reach an AFS or the depot k. In this situation, we try to insert depot k in previous node. The process is repeated till a place is found for inserting depot k.

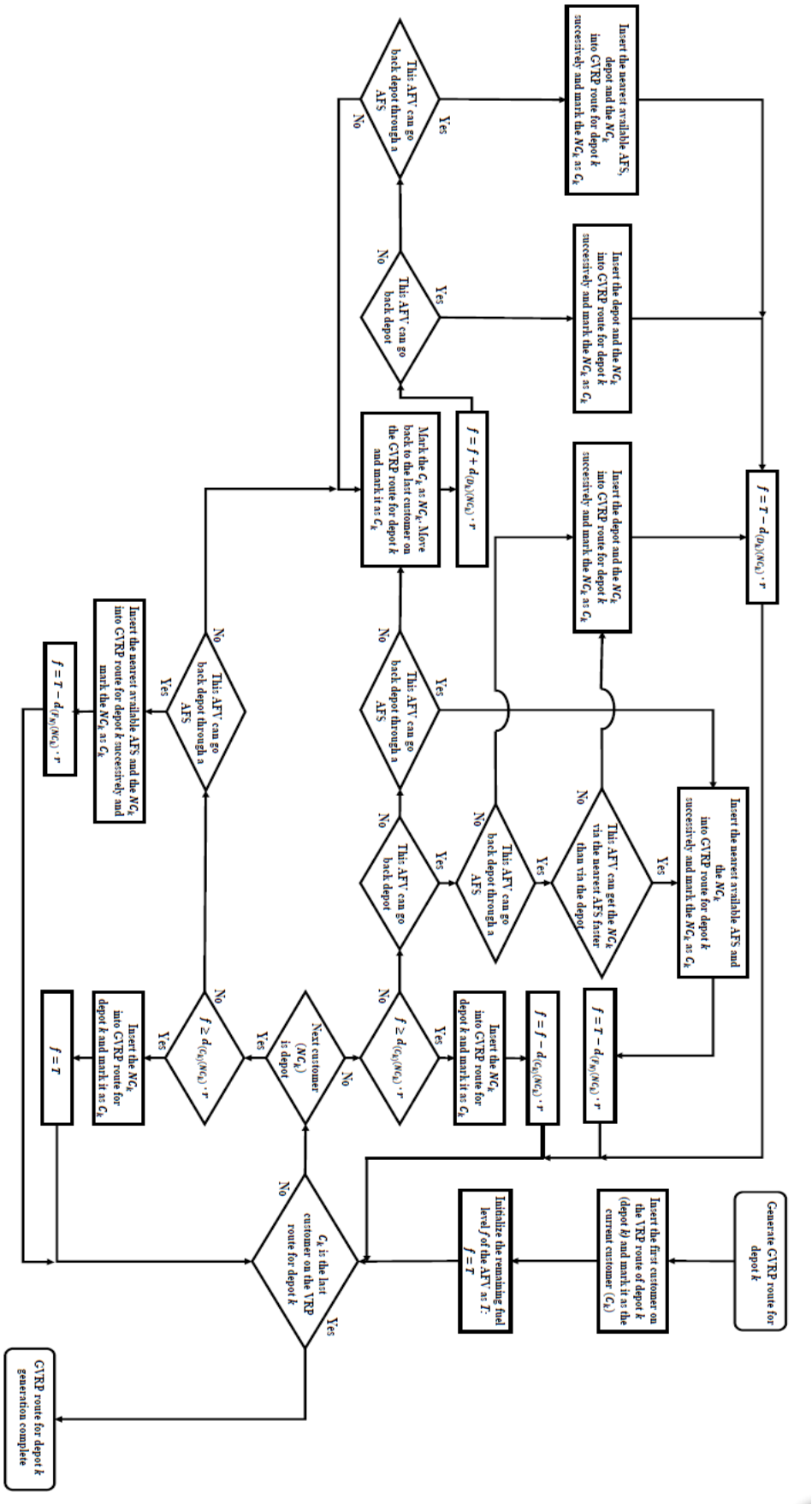

Fig.13. Process of generating GVRP routes for depot *k*

3.4.2. Two-stages Ant Colony System (TSACS) Algorithm

It was observed that ants usually walk through the shortest route between their nest and the food (Dorigo et al., 1996) because of the different levels of pheromone density in different routes. By simulating the food-seeking behaviors of ant colonies in nature, the Ant Colony System (ACS) algorithm was developed by Dorigo et al. (1996). In the last 20 years, the ACS algorithm has been successfully applied to solve the VRP and its variants (e.g. Bell and McMullen (2004), Dorigo et al. (1996), Gambardella, Montemanni, Rizzoil and Lucibello (2007), Gajpal and Abad (2009), Lin et al. (2014), Montoya-Torres et al. (2015) , Yu et al. (2011) and Ting and Chen (2013), etc.).

In this thesis, we proposed a Two-stage Ant Colony System (TSACS) algorithm specially designed for MDGVRP. Same as the ACSNVS proposed to solve MDVRP in chapter 2, the proposed TSACS algorithm also uses two types of ants, *depot-ants* to assign customers to depot and *route-ants* to generate routes, respectively. The trail intensity denoted as τ_{ik}^{d} is related to *depot-ants* and the trail intensity denoted as τ_{ij}^{r} is related to *route-ants*. More introduction of these two types of ants is described in chapter 2.4. The main steps of the TSACS are as follows:

Step 1. Initialize two trail intensity matrixes (τ_{ik}^{d}) and (τ_{ij}^{r}). Create A number of *depot-ants* and *route-ants*, which means that there are A number of solutions generated in each iteration. Set $iteration = 1$.

Step 2. While $iteration \leq 100$ do the following:

Step 2.1. Generate an ant-solution for each ant using the trail intensities.

Step 2.2. Improve each ant-solution by using two types of local search in the following order:

- Redundancy removal local search.

- Insertion local search.
- Redundancy removal local search.

Step 2.3. Update the best solution found so far.

Step 2.4. Update trail intensities for *depot-ants* and *route-ants* according to the best solution found so far.

Step 3. Return the best solution found so far.

In the following section, all steps of applying this TSACS are provided in detail.

3.4.2.1. Initial Trail Intensity Matrices

At the beginning of the whole algorithm, trail intensity of *depot-ants* τ_{ik}^{d} and trail intensity of *route-ants* τ_{ij}^{r} are initialized. As mentioned in previous sections, there is a significant difference between MDVRP and MDGVRP. In MDVRP, customers can be assigned to any depot. However, in MDGVRP, some customers can only be assigned to a few or even only one depot because of the limited tank capacity of vehicles. The customer i can be feasibly assigned to depot k if at least one of the following condition is satisfied:

Condition 1. A vehicle can depart from depot k to visit customer i and come back depot k without refueling.

Condition 2. A vehicle can depart from depot k to visit customer i by refueling at an AFS l, and then the vehicle can go back to depot k from customer i directly.

Initially all trail intensities are kept same. Thus the trail intensity τ_{ik}^{d} is initialized as follows.

$$\tau_{ik}^{d} = \begin{cases} 0.01, & if\ customer\ i\ can\ be\ assigned\ to\ depot\ l \\ 0, & otherwise \end{cases}, \qquad \forall i \in C, k \in D$$

The trail intensity for *route-ants* τ_{ij}^{r} is initialized as $\tau_{ij}^{r} = 0.01\ (\forall i,j \in V)$.

3.4.2.2. Generate Ant Solution

As mentioned in chapter 3.3.2, ACSVNS and TSACS are designed to solve MDVRP and MDGVRP, respectively, they have similar solving strategy, firstly assigning customers to different depots and secondly generate routes for each depots. However, due to the difference between MDVRP and MDGVRP, trail intensity initialization process and routes generation process of ACSVNS and TSACS are different.

In TSACS, each artificial ant generates one feasible solution in every iteration. For MDGVRP, a feasible solution is a set of GVRP routes for all depots (MDGVRP routes). The MDGVRP solution for each artificial ant is generated in two steps described below.

Step 1. Assign Customers to Depots

The first step is to assign customers to different depots using *depot-ant* and the probability of assigning customer i to depot k is as follows:

$$P_{ik}^{d} = \frac{\tau_{ik}^{d}}{\sum_{l \in D} \tau_{il}^{d}}, \qquad \forall i \in C, k \in D \tag{3.12}$$

Assignment of customers to depot starts with first customer and then other customers are iteratively assigned to different depots to form set $CD_k\ (\forall k \in D)$. The term $CD_k\ (\forall k \in D)$ represents the set of customers assigned to depot k. At the end of this step, a set CD_k is obtained for each depot $k\ (\forall k \in D)$.

Step 2. Generate GVRP Routes

After assigning customers to depots, *routes-ant* is used to generate TSP routes for each

depot based on the trail intensity of *route-ants* τ_{ij}^{r}. Let $VC_k\ (\forall k \in D)$ stands for the set of unvisited customer for depot k. In the beginning, set VC_k is initialized as $VC_k = CD_k$. For depot k, the TSP route starts from depot k and visits the customer one by one from set VC_k to form TSP route. The probability of visiting customer j from the current customer i is as follows:

$$P_{ij} = \frac{\tau_{ij}^{r}}{\sum_{l \in VC_k} \tau_{il}^{r}}, \qquad \forall j \in VC_k \tag{3.13}$$

The selected customer is removed from set VC_k. The procedure continues till set VC_k becomes empty. After generating TSP routes, GVRP routes are generated based on the procedure described in chapter 3.3.1.

3.4.2.3. Local Search

After generating an ant-solution for MDGVRP, we apply local search on the solution of each artificial ant. In TSACS, we adopt two types of local search, redundancy removal (RR) local search and insertion local search.

Local Search 1: Redundancy Removal (RR) Local Search

The ant-solution contains depots and AFSs as redundant nodes which can be removed to improve the solution quality. The ant solution generates the redundant nodes as well as the insertion local search. Therefore, this local search is applied twice. First time it is applied after generating ant solution and the second time it is applied after using insertion local search. The results of numerical experiments show that the redundancy removal local search can greatly improve the quality of solutions and reduce solving time at the same time. This will be discussed in the section 5.

Local Search 2: Insertion Local Search

Insertion local search finds the neighbor solutions by inserting the customer to all other feasible positions. An example of local search for MDGVRP is as follows: consider a problem with six customers denoted as c_1, c_2, c_3, c_4, c_5 and c_6; three depots denoted as D_1, D_2 and D_3; and a two AFSs denoted as F_1 and F_2. Consider a following starting solution S_0:

$$S_0 = \begin{cases} D_1 - c_2 - F_1 - c_4 - F_1 - D_1 \\ D_2 - c_3 - D_2 - c_1 - F_2 - c_5 - D_2 \\ D_3 - c_6 - F_1 - D_3 \end{cases}$$

Step 1. Choose the first customer (c_2) from S_0 and insert it into all **feasible positions** and record the best neighbor solution.

$$Neighbor\ Solution\ S_1 = \begin{cases} D_1 - F_1 - \boldsymbol{c_2} - c_4 - F_1 - D_1 \\ D_2 - c_3 - D_2 - c_1 - F_2 - c_5 - D_2 \\ D_3 - c_6 - F_1 - D_3 \end{cases}$$

$$Neighbor\ Solution\ S_2 = \begin{cases} D_1 - F_1 - c_4 - \boldsymbol{c_2} - F_1 - D_1 \\ D_2 - c_3 - D_2 - c_1 - F_2 - c_5 - D_2 \\ D_3 - c_6 - F_1 - D_3 \end{cases}$$

$$Neighbor\ Solution\ S_3 = \begin{cases} D_1 - F_1 - c_4 - F_1 - \boldsymbol{c_2} - D_1 \\ D_2 - c_3 - D_2 - c_1 - F_2 - c_5 - D_2 \\ D_3 - c_6 - F_1 - D_3 \end{cases}$$

... ...

$$Neighbor\ Solution\ S_8 = \begin{cases} D_1 - F_1 - c_4 - F_1 - D_1 \\ D_2 - c_3 - D_2 - c_1 - F_2 - c_5 - D_2 \\ D_3 - c_6 - F_1 - \boldsymbol{c_2} - D_3 \end{cases}$$

Eight neighbor solutions can be found in this round. Assume that neighbor solution S_6 is the best neighbor solution. This solution is considered for further evaluation.

$$Best\ Neighbor\ Solution = \begin{cases} D_1 - F_1 - c_4 - F_1 - D_1 \\ D_2 - c_3 - D_2 - c_1 - \boldsymbol{c_2} - F_2 - c_5 - D_2 \\ D_3 - c_6 - F_1 - D_3 \end{cases}$$

Step 2. Choose the next customer from the solution found in step 1 repeat the procedures in step 1.

Step 3. Accept the solution found in step 2 if it is better than the initial solution S_0.

3.4.2.4. Update the Trail Intensity.

At the end of each iteration, trail intensities of *depot-ants* τ_{ik}^{a} and trail intensities of *route-ants* τ_{ij}^{t} are required to update. The update steps and description are the same as ACSVNS which introduced in chapter 2.4.4.

3.5. Numerical Experiments and Analysis.

The MDGVRP is first time introduced in this thesis. Therefore new problem instances are generated to test the performance of the two algorithms designed for solving MDGVRP. These instances can be used as a benchmark problem instances for future research.

We generated 17 small problem instances and 60 large problem instances randomly. The positions of depots, AFSs and customers are randomly generated on 300 by 300 mile grid, while generating the customer location it is ensured that a customer can be assigned to at least one depot. The tank capacity of vehicles T is 60 (liters) and fuel consumption rate r is 0.2 (liters/mile). The demand of each customer q_i $(\forall i \in C)$ is generated between 15 and 25 and the capacity of vehicle Q is 300.

These 18 small instances (labeled as MDGVRPT 1 to MDGVRPT 18) consider number of customer (N) as 5, 7, 9, 11, 13 and 15; the number of depots (M) as 2, 3 and 4; and the number of AFSs (NS) as 2, respectively. Thus a total of 18 small instances is generated. The large instances (labeled as MDGVRP 1 to MDGVRP 60), considers the situations where the number of customers is 25, 50, 75, 100, 150 and 200; the number of depots is 4, 6 and 8; and the number of AFSs is 2, 4, 6 and 8, respectively. Thus a total of 60 large instances is generated. The small problem instances are used to obtain the optimal solution by solving mathematical

formulation provided in section 3. The large problem instances are generated to evaluate the relative performance of proposed algorithms. Before testing the instances, all necessary parameters are set as follows,

For TSACS, we set the number of iteration A as 100, number of artificial ants y as 100, trail persistence for *depot-ants* α and trail persistence for *route-ants* β as 0.95. For PBA, the rate distinguishing between borderline and non-borderline customer r_a is set as 0.7.

3.5.1. Experiment Results and Analysis.

For numerical experiments, we programed the TSACS and PBA algorithms on C++ and implemented them on AMD Opteron 2.3 GHz with 16GB of RAM. The mathematical formulation is solved using AMPL on a desktop with Intel Core i5-4590 3.3GHz with 8GB of RAM. The computing time of mathematical formulation is affected by number of dummy vertices. Therefore, we set the number of dummy vertices for each AFS to 2 (3 nodes in total for each AFS) and the maximum number of vehicles is three times the number of depots. All experiment results are provided in the following sections.

3.5.2. Experiments on Small-size Instances.

The purpose of small-size instances is to evaluate the absolute performance of proposed algorithm. We use formulation described in section 3 to obtain optimal solution. We report the MDGVRP route length (miles), CPU time (seconds) used to solve the instance and Percentage Gap (PG) from the optimal solution for proposed TSACS and PBA algorithm. The results are shown in the Table 5.

Table 5. Experiment results of small-sized instance

Instance	*N*	*NS*	*M*	CPLEX		TSACS			PBA		
				Optimal Solution (miles)	CPU Time (Seconds)	Route Length (miles)	CPU Time (Seconds)	PG (%)	Route Length (miles)	CPU Time (Seconds)	PG (%)
MDGVRPT1	5	2	2	896.22	0.22	896.22	1.00	0.00	1348.67	<1	50.48
MDGVRPT2	7	2	2	955.42	0.52	955.42	2.42	0.00	1432.71	<1	49.96
MDGVRPT3	9	2	2	1028.53	19.86	1028.53	3.20	0.00	1947.98	<1	89.39
MDGVRPT4	11	2	2	1074.67	190.38	1074.67	4.70	0.00	1702.21	<1	58.39
MDGVRPT5	13	2	2	1176.29	863.11	1176.29	8.67	0.00	2249.32	<1	91.22
MDGVRPT6	15	2	2	1324.98	6012.00	1334.13	12.68	0.69	3436.14	<1	159.34
MDGVRPT7	5	2	3	819.35	0.35	819.35	1.01	0.00	1182.58	<1	44.33
MDGVRPT8	7	2	3	867.28	0.31	867.28	1.60	0.00	1230.50	<1	41.88
MDGVRPT9	9	2	3	990.66	0.74	990.66	2.48	0.00	990.66	<1	0.00
MDGVRPT10	11	2	3	1027.07	3.62	1027.07	3.72	0.00	1027.07	<1	0.00
MDGVRPT11	13	2	3	1078.13	4.69	1078.13	5.35	0.00	1095.19	<1	1.58
MDGVRPT12	15	2	3	1305.94	12907.60	1305.94	9.00	0.00	1343.83	<1	2.90
MDGVRPT13	5	2	4	687.24	0.28	687.24	1.35	0.00	764.00	<1	11.17
MDGVRPT14	7	2	4	854.54	0.37	854.54	2.12	0.00	854.54	<1	0.00
MDGVRPT15	9	2	4	891.09	0.90	891.09	3.24	0.00	891.09	<1	0.00
MDGVRPT16	11	2	4	934.67	1.47	934.67	4.72	0.00	978.21	<1	4.66
MDGVRPT17	13	2	4	1162.48	31.22	1162.48	7.43	0.00	1209.08	<1	4.01
Average				1004.39	1178.68	1004.92	4.39	0.04	1393.16	<1	35.84

From Table 5, it can be seen that in 16 out of 17 small-size instances, TSACS can find the optimal solution in 4.82 seconds on average, which is much less than the 1178.68 seconds spent by CPLEX. It can also be observed that when the problem size increases, the CPU time of CPLEX rises dramatically. We could not solve the instance for more than 15 customers.

The PBA got the optimal solution for 4 instances, but the average PG between solutions of PBA and the optimal solution is 35.84%. The PG for MDGVRPT6 is 159.34%, which shows that the solution quality of PBA is not as good as that of TSACS for small-size instances.

3.5.3. Experiments on Large-size Instances.

The optimal solution cannot be obtained for bigger problem instances. Therefore, Relative

Percentage Deviation (RPD) is used to evaluate the performance of TSACS and PBA. For j^{th} instance and i^{th} algorithm, the RPD_{ij} is calculated based on the following equation:

$$RPD_{ij} = \frac{(H_j^i - B_j)}{B_j} \times 100\% \quad 3.14$$

In this formula, H_j^i stands for the solution obtained by the i^{th} algorithm for the j^{th} instance and B_j represents the best solution from all evaluated algorithms for the j^{th} problem instance. The results of large instances are shown in Table 6.

Table 6. Experiment results of large-size instance

Instance	*N*	*NS*	*M*	TSACS			PBA		
				Route Length (Miles)	CPU Time (Seconds)	RPD (%)	Route Length (Miles)	CPU Time (Seconds)	RPD (%)
MDGVRP1	25	2	4	1635.71	44.46	0.00	2538.25	<1	55.18
MDGVRP2	50	2	4	2565.44	324.42	0.00	6303.19	<1	145.70
MDGVRP3	100	2	4	4002.66	2333.56	0.00	9290.51	<1	132.11
MDGVRP4	150	2	4	5901.93	7940.30	0.00	12503.50	<1	111.85
MDGVRP5	200	2	4	6861.84	18243.80	0.00	12838.80	<1	87.10
MDGVRP6	25	4	4	1535.80	49.20	0.00	2616.82	<1	70.39
MDGVRP7	50	4	4	2232.17	332.71	0.00	6616.30	<1	196.41
MDGVRP8	100	4	4	3901.54	2622.65	0.00	8073.64	<1	106.93
MDGVRP9	150	4	4	5599.71	9250.13	0.00	9945.53	<1	77.61
MDGVRP10	200	4	4	6709.83	23223.10	0.00	14123.40	<1	110.49
MDGVRP11	25	6	4	1425.99	49.24	0.00	2361.22	<1	65.58
MDGVRP12	50	6	4	2024.93	343.22	0.00	5322.98	<1	162.87
MDGVRP13	100	6	4	3658.60	2830.63	0.00	7330.17	<1	100.35
MDGVRP14	150	6	4	5423.17	10208.10	0.00	10579.60	<1	95.08
MDGVRP15	200	6	4	6662.95	24216.20	0.00	12257.00	<1	83.96
MDGVRP16	25	8	4	1447.71	57.36	0.00	2333.55	<1	61.19
MDGVRP17	50	8	4	2064.51	411.41	0.00	4785.81	<1	131.81
MDGVRP18	100	8	4	3566.53	3288.45	0.00	9059.93	<1	154.03
MDGVRP19	150	8	4	5668.90	11839.80	0.00	10727.20	<1	89.23
MDGVRP20	200	8	4	6895.52	29353.40	0.00	11931.50	<1	73.03
MDGVRP21	25	2	6	1315.10	44.41	0.00	1934.56	<1	47.10
MDGVRP22	50	2	6	1935.35	326.44	0.00	5689.22	<1	193.96
MDGVRP23	100	2	6	3300.23	2526.33	0.00	11171.70	<1	238.51
MDGVRP24	150	2	6	4622.66	8417.02	0.00	10925.20	<1	136.34

MDGVRP25	200	2	6	6593.05	21910.30	0.00	16779.70	<1	154.51
MDGVRP26	25	4	6	1414.12	58.04	0.00	1965.17	<1	38.97
MDGVRP27	50	4	6	2093.92	373.73	0.00	6370.88	<1	204.26
MDGVRP28	100	4	6	4043.84	2905.73	0.00	9992.36	<1	147.10
MDGVRP29	150	4	6	5556.45	10743.60	0.00	8457.54	<1	52.21
MDGVRP30	200	4	6	6588.35	25550.10	0.00	14173.10	<1	115.12
MDGVRP31	25	6	6	1398.51	60.34	0.00	2150.05	<1	53.74
MDGVRP32	50	6	6	1988.82	416.05	0.00	5031.00	<1	152.96
MDGVRP33	100	6	6	3469.46	3689.33	0.00	6969.81	<1	100.89
MDGVRP34	150	6	6	5211.99	12764.20	0.00	9271.03	<1	77.88
MDGVRP35	200	6	6	6774.36	28925.00	0.00	17161.90	<1	153.34
MDGVRP36	25	8	6	1635.42	73.42	0.00	2273.43	<1	39.01
MDGVRP37	50	8	6	2129.04	477.60	0.00	5722.04	<1	168.76
MDGVRP38	100	8	6	3477.16	3721.45	0.00	8051.29	<1	131.55
MDGVRP39	150	8	6	5500.62	14103.00	0.00	14675.00	<1	166.79
MDGVRP40	200	8	6	6562.86	30803.60	0.00	15832.60	<1	141.25
MDGVRP41	25	2	8	1452.60	60.12	0.00	1862.59	<1	28.22
MDGVRP42	50	2	8	2296.16	399.02	0.00	3319.38	<1	44.56
MDGVRP43	100	2	8	3985.42	3397.45	0.00	5646.05	<1	41.67
MDGVRP44	150	2	8	5900.79	11138.60	0.00	8460.22	<1	43.37
MDGVRP45	200	2	8	8035.46	26622.30	0.00	8732.05	<1	8.67
MDGVRP46	25	4	8	1431.83	64.40	0.00	1738.87	<1	21.44
MDGVRP47	50	4	8	2102.15	483.91	0.00	3476.42	<1	65.37
MDGVRP48	100	4	8	3458.22	3561.82	0.00	5159.57	<1	49.20
MDGVRP49	150	4	8	6462.17	13122.60	0.00	8765.33	<1	35.64
MDGVRP50	200	4	8	6504.99	30181.90	0.00	9677.55	<1	48.77
MDGVRP51	25	6	8	1548.83	75.99	0.00	1886.90	<1	21.83
MDGVRP52	50	6	8	2062.96	491.73	0.00	3471.80	<1	68.29
MDGVRP53	100	6	8	3623.67	4068.03	0.00	4817.82	<1	32.95
MDGVRP54	150	6	8	4360.03	13215.50	0.00	11329.00	<1	159.84
MDGVRP55	200	6	8	6258.14	32654.90	0.00	10337.00	<1	65.18
MDGVRP56	25	8	8	1605.07	87.27	0.00	1952.87	<1	21.67
MDGVRP57	50	8	8	1938.14	496.52	0.00	3499.92	<1	80.58
MDGVRP58	100	8	8	3470.67	4064.62	0.00	5316.21	<1	53.18
MDGVRP59	150	8	8	4616.38	13920.50	0.00	8608.19	<1	86.47
MDGVRP60	200	8	8	5643.40	31956.90	0.00	9471.00	<1	67.82
Average				3869.23	8415.27	0.00	7561.09	<1	94.50

In Table 6, it is obvious that the quality of solutions generated by TSACS is much better than that of PBA. The RPD of TSACS is 0 for all problem instances which means that TSACS can obtain better solution for all problem instances. The average RPD of PBA is 94.50%, which

means that the solution of PBA is 94.50% worse than that of TSACS on average. In addition, the solution quality of PBA is very "unstable". For some instances, such as MDGVRP45 and MDGVRP51, PBA can get relatively high quality solutions. However, for most of other instances, the solution quality of PBA is unsatisfactory. However, PBA has a very good performance in solving time. The PBA can solve all instances within one second. For TSACS, its solving time increases with the size of instance. Table 3 shows the average solving time of TSACS with the different number M.

Table 7. Average Solving Time of TSACS

M	Average TSACS Solving Time （Seconds）
4	7348.11
6	8394.49
8	9503.21

From Table 7, we can see that for the instance which consider 8 depots, TSACS needs 9503.21 seconds on average to solve the problem. In fact, for TSACS, solution quality and CPU time are correlated. The CPU time can be shortened by decreasing the number of iterations or reducing the number of artificial ants. However, decreasing the CPU time decreases the solution quality. Therefore a trade-off analysis is performed to fix the number of iterations and number of ants.

3.5.4. Effect of Redundancy Removal (RR) Local Search

In TSACS, the use of RR local search greatly affects the solution quality. Table 6 reveals the results obtained by the TSACS with the use of RR local search and without the use of RR

local search.

Table 8. Experiments For The Effect of RR Local Search

Instance	*N*	*NS*	*M*	TSACS (Without)		TSACS (with RR Local Search)			
				Route Length (miles)	CPU Time (Seconds)	Route Length (miles)	CPU Time (Seconds)	Routes Length Decrease （%）	Solving Time Decrease （%）
MDGVRP1	25	2	4	1700.36	52.58	1635.71	44.46	3.95	18.26
MDGVRP2	50	2	4	2541.66	404.70	2565.44	324.42	-0.93	24.75
MDGVRP3	100	2	4	4922.67	3340.21	4002.66	2333.56	22.98	43.14
MDGVRP4	150	2	4	6775.71	10826.40	5901.93	7940.30	14.80	36.35
MDGVRP5	200	2	4	9479.03	26287.60	6861.84	18243.80	38.14	44.09
MDGVRP6	25	4	4	1621.55	57.62	1535.80	49.20	5.58	17.11
MDGVRP7	50	4	4	2531.20	462.25	2232.17	332.71	13.40	38.93
MDGVRP8	100	4	4	4335.00	3412.64	3901.54	2622.65	11.11	30.12
MDGVRP9	150	4	4	6015.45	12107.20	5599.71	9250.13	7.42	30.89
MDGVRP10	200	4	4	7906.80	28925.80	6709.83	23223.10	17.84	24.56
MDGVRP11	25	6	4	1441.88	68.22	1425.99	49.24	1.11	38.55
MDGVRP12	50	6	4	2118.59	457.76	2024.93	343.22	4.63	33.37
MDGVRP13	100	6	4	4087.03	3758.44	3658.60	2830.63	11.71	32.78
MDGVRP14	150	6	4	6802.10	13503.80	5423.17	10208.10	25.43	32.29
MDGVRP15	200	6	4	10599.60	32577.20	6662.95	24216.20	59.08	34.53
MDGVRP16	25	8	4	1447.71	61.57	1447.71	57.36	0.00	7.34
MDGVRP17	50	8	4	2331.17	507.22	2064.51	411.41	12.92	23.29
MDGVRP18	100	8	4	4014.63	4027.47	3566.53	3288.45	12.56	22.47
MDGVRP19	150	8	4	6302.71	14275.30	5668.90	11839.80	11.18	20.57
MDGVRP20	200	8	4	7867.94	33152.90	6895.52	29353.40	14.10	12.94
MDGVRP21	25	2	6	1320.14	61.35	1315.10	44.41	0.38	38.14
MDGVRP22	50	2	6	1944.22	455.40	1935.35	326.44	0.46	39.50
MDGVRP23	100	2	6	3733.76	3590.66	3300.23	2526.33	13.14	42.13
MDGVRP24	150	2	6	4992.16	12929.90	4622.66	8417.02	7.99	53.62
MDGVRP25	200	2	6	7002.04	30505.40	6593.05	21910.30	6.20	39.23
MDGVRP26	25	4	6	1393.94	84.69	1414.12	58.04	-1.43	45.92
MDGVRP27	50	4	6	2291.56	577.56	2093.92	373.73	9.44	54.54
MDGVRP28	100	4	6	4890.12	4799.14	4043.84	2905.73	20.93	65.16
MDGVRP29	150	4	6	6301.28	15268.00	5556.45	10743.60	13.40	42.11
MDGVRP30	200	4	6	10396.20	38493.50	6588.35	25550.10	57.80	50.66
MDGVRP31	25	6	6	1481.84	89.51	1398.51	60.34	5.96	48.34
MDGVRP32	50	6	6	2145.35	556.68	1988.82	416.05	7.87	33.80
MDGVRP33	100	6	6	3731.32	4435.06	3469.46	3689.33	7.55	20.21
MDGVRP34	150	6	6	6548.96	16479.50	5211.99	12764.20	25.65	29.11
MDGVRP35	200	6	6	8182.66	38447.40	6774.36	28925.00	20.79	32.92

MDGVRP36	25	8	6	1591.82	90.67	1635.42	73.42	-2.67	23.49
MDGVRP37	50	8	6	2114.69	618.10	2129.04	477.60	-0.67	29.42
MDGVRP38	100	8	6	3623.30	4797.56	3477.16	3721.45	4.20	28.92
MDGVRP39	150	8	6	5811.25	17611.30	5500.62	14103.00	5.65	24.88
MDGVRP40	200	8	6	8234.16	41633.00	6562.86	30803.60	25.47	35.16
MDGVRP41	25	2	8	1408.11	78.71	1452.60	60.12	-3.06	30.92
MDGVRP42	50	2	8	2405.94	600.29	2296.16	399.02	4.78	50.44
MDGVRP43	100	2	8	4736.35	4604.43	3985.42	3397.45	18.84	35.53
MDGVRP44	150	2	8	7259.98	16023.80	5900.79	11138.60	23.03	43.86
MDGVRP45	200	2	8	11676.50	39624.70	8035.46	26622.30	45.31	48.84
MDGVRP46	25	4	8	1476.28	83.13	1431.83	64.40	3.10	29.08
MDGVRP47	50	4	8	2210.03	605.38	2102.15	483.91	5.13	25.10
MDGVRP48	100	4	8	3427.44	4778.76	3458.22	3561.82	-0.89	34.17
MDGVRP49	150	4	8	5372.35	15905.10	6462.17	13122.60	-16.86	21.20
MDGVRP50	200	4	8	8878.25	39558.60	6504.99	30181.90	36.48	31.07
MDGVRP51	25	6	8	1529.85	97.12	1548.83	75.99	-1.23	27.81
MDGVRP52	50	6	8	2093.30	632.58	2062.96	491.73	1.47	28.64
MDGVRP53	100	6	8	3399.43	5024.50	3623.67	4068.03	-6.19	23.51
MDGVRP54	150	6	8	5580.63	17643.50	4360.03	13215.50	28.00	33.51
MDGVRP55	200	6	8	12358.70	45814.30	6258.14	32654.90	97.48	40.30
MDGVRP56	25	8	8	1460.64	97.64	1605.07	87.27	-9.00	11.88
MDGVRP57	50	8	8	2083.61	669.05	1938.14	496.52	7.51	34.75
MDGVRP58	100	8	8	3067.47	5026.62	3470.67	4064.62	-11.62	23.67
MDGVRP59	150	8	8	5025.72	17680.70	4616.38	13920.50	8.87	27.01
MDGVRP60	200	8	8	6127.35	41408.70	5643.40	31956.90	8.58	29.58
Average				4569.19	11262.48	3869.23	8415.27	18.09	33.83

From Table 8, it can be seen that for the 60 large instances, the average route length obtained by TSACS is 3869.23 miles when RR local search is used. This value is 18.09% less that the average length generated by the TSACS when RR local search is not used. At the same time, the average solving time of TSACS is 11262.48 seconds when RR local search is not used. This number is 33.83% more than the 8415.27 seconds when RR local search is used. These results indicate that the use of RR local search not only increases the solution quality but also decreases the CPU time. TSACS with RR local search can find high-quality solutions faster because many redundant nodes are removed before local search and the number of low-

quality neighbor solutions are reduced. Lower number of neighbor solutions reduces the solving time of TSACS.

Therefore, RR local search is crucial for TSACS to solve MDGVRP to improve the solution quality and to reduce solving time. We also apply this technique to PBA, however, it is not very effective in PBA. In PBA, all border-line customers are inserted into a proper place in the MDGVRP routes first, therefore the number of redundant nodes are quite low. Because of this reason, the solution quality improvement after RR local search in PBA is not very significant.

3.6. Conclusion.

Recent years, more and more multi-national transportation and retail corporations are not only considering their economic sustainability but also their environmental sustainability performance. Under this background, in this chapter, Multi-Depot Green Vehicle Routing Problem (MDGVRP) is addressed. In MDGVRP, the Alternative Fuel-powered Vehicles (AFVs) are used to deliver goods to customers. These AFVs departure from different depots, serve customers, and at the end returned to their original depots. In the service process, AFVs need to consider the remaining fuel level and remaining cargo level. The MDGVRP is formulated as MILP formulation to find the optimal solution.

We proposed two algorithms, Two-stage Ant Colony System (TSACS) Algorithm and Partition Based Algorithm (PBA) to solve MDGVRP. We generated 18 small-size instances and 60 large-size instances to test the performance of TSACS and PBA. The experiment results reveal that although the PBA can solve each instance within 1 second, the solution quality is

unsatisfactory. However, TSACS has a very good overall performance on solution quality. It can search the best solution for all 17 small-size instances in 4.82 seconds and the solution is only 0.04% worse than the optimal solution on average. These results indicate that the swarm intelligence based on ant colony algorithm can effectively solve MDGVRP.

With the development of the alternative fuel-powered vehicles and the growing concern for environmental protection of people, we believe the research on MDGVRP will bear more fruits in the future.

CHAPTER 4

CONCLUSION

In recent years, global warming has become the most serious environmental problem. The main reason of global warming is the greenhouse gases (GHG) emissions. In addition, carbon dioxide (CO_2) is the major GHG. According to the report of International Energy Agency (IEA), 23% of global carbon dioxide (CO_2) was generated from transportation sector. Besides, nearly 75% of the emissions in transportation sector were generated from the road sector in 2013 (IEA 2014). Under this context, due to the growing environmental concern worldwide, GVRP has been becoming a very high-profile research topic in VRP field. In addition, the research on MDGVRP can guide companies to build more efficient routes for their EVs, which can extend battery life of EVs and reduce the environmental impacts of recycling batteries.

In chapter 2, an Ant Colony System with Variable Neighborhood Search algorithm (ACSVNS) is proposed to solve Multi-depot Vehicle Routing Problem (MDVRP). In ACSVNS, perturbation scheme, relocate local search and revised swap local search are applied to improve the solution quality. The reuse of Remove Cost (*RC*) and Insertion Cost (*IC*) matrix during the whole neighborhood search process is the distinct feature of ACSVNS and this reuse technique is first proposed in this thesis. Besides, the revised swap local search which applied in ACSVNS is also first proposed in the VRP literature. Compared with the classic swap local search, this revised swap local search is more efficient and effective. Based on the experiment results of the 23 benchmark instances of MDVRP, ACSVNS can match the existing best known solution for 16 instances, which is the third best in all algorithms in MDVRP literature. Additionally, ACSVNS found new best solutions for the instance 5, 6 and 7. These results

prove that ACSVNS has a very good performance in solving MDVRP. The work on MDVRP in this thesis lays a solid foundation for the work on MDGVRP in chapter 3.

Based on the MDVRP and GVRP literature, in this thesis, a new variant vehicle routing problem, Multi-depot Green Vehicle Routing Problem (MDGVRP), is formulated. In MDGVRP, the AFVs start from different depots, serve customers, and at the end return to the original depots. Due to the limited capacity of fuel tank of AFVs and the scarcity of AFSs, each AFV is required to visit AFS for refueling. A revised Partition Based Algorithm (PBA) and a Two-stage Ant Colony System (TSACS) algorithm are proposed to find solutions for this problem. 17 small-sized instances and 60 large-sized instances are generate the performance of those two algorithms. The experiment results show that while PBA has a shorter solving time, TSACS have a better overall performance on both solving time and solution quality.

Although TSACS can solve MDGVRP much faster than CPLEX, the time complexity of some functions in TSACS, such as Redundancy Removal Local Search, is very high, which means that with the problem size increasing, the solving time will increase dramatically. Therefore, designing more efficient algorithm for MDGVRP or design better local search techniques for TSACS is a very good direction of future research. This thesis is a milestone of the research on MDGVRP. With the importance of protecting environment and economic globalization, I believe the research on MDGVRP will be more important in the future.

Reference:

Bard, J. F., Huang, L. I. U., Dror, M., & Jaillet, P. (1998). A branch and cut algorithm for the VRP with satellite facilities A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, *30*(9), 821–834.

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

Cassidy, P. J., & Bennett, H. S. (1972). TRAMP -- A Multi-Depot Vehicle Scheduling System. *Operation Research Quarterly*, *23*(2), 151–163.

Cordeau JF, GendreauMandLaporteG(1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30:105–119.

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, *6*(1), 80–91.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a Colony of Cooperating Agents, *26*(1), 1–13.

Erdoğan, S., & Miller-Hooks, E. (2012). A Green Vehicle Routing Problem. *Transportation Research Part E: Logistics and Transportation Review*, *48*(1), 100–114.

Escobar, J. W., Linfati, R., Toth, P., & Baldoquin, M. G. (2014). A hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem. *Journal of Heuristics*, *20*(5), 483–509.

Farneti, F., & Guthrie, J. (2009). Sustainability reporting by Australian public sector organisations : Why they report. *Accounting Forum*, *33*(2), 89–98.

Felipe, Á., Ortuño, M. T., Righini, G., & Tirado, G. (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, *71*, 111–128.

Fraer, R., Dinh, H., Chandler, K., & Buchholz, B. (2005). Operating experience and teardown analysis for engines operated on biodiesel blends(B20). *SAE Technical Paper*, *2005*(01–3641).

Gajpal, Y., & Abad, P. (2009a). An ant colony system (ACS) for vehicle routing problem

with simultaneous delivery and pickup. *Computers and Operations Research*, *36*(12), 3215–3223.

Gajpal, Y., & Abad, P. L. (2009b). Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, *196*(1), 102–117.

Gambardella, L. M., Montemanni, R., Rizzoil, A. E., & Lucibello, E. (2007). Ant colony optimization for real-world vehicle routing From theory to applications. *Swarm Intelligence*, *1*(2), 135–151.

Gutin, G., Yeo, A., & Zverovich, A. (2002). Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*, *117*(1–3), 81–86.

IEA (2014). CO2 Emissions From Fuel Combustion Highlights 2014. International Energy Agency, Paris, 10.

IEA (2015). CO2 Emissions From Fuel Combustion Highlights 2015. International Energy Agency, Paris, 7-11Kek, A. G., Cheu, R. L., & Meng, Q. (2008). Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots. Mathematical and Computer Modelling, 47(1), 140-152.

Jabir, E., Panicker, V. V., & Sridharan, R. (2017). Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem. *Transportation Research Part D: Transport and Environment*, *57*, 422–457.

Koç, Ç., & Karaoglan, I. (2016). The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing Journal*, *39*, 154–164.

Kulkarni, R.V., & Bhave, P. R. (1985). Integer programming for mulations of vehicle Routing problems. *European Journal of Operational Research*, *20*(1), 58–67.

Laporte, G., & Osman, I. H. (1995). Routing problems: A bibliography. *Annals of Operations Research*, *61*(1), 227–262.

Lin, C., Choy, K. L., Ho, G. T. S., Chung, S. H., & Lam, H. Y. (2014). Survey of Green Vehicle Routing Problem: Past and future trends. *Expert Systems with Applications*, *41*(4 PART 1), 1118–1138.

Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple

depots. *Computers & Industrial Engineering*, *79*, 115–129.

Montoya, A., Gueret, C., Mendoza, J. E., & Villegas, J. G. (2016). A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, *70*, 113–128.

Nagy, G., & Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, *162*(1), 126–141.

RenaudJ, Laporte G & Boctor FF (1996). A tabu search heuristic for the multi-depot vehicle routing problem. ComputOpnsRes 23:229–235.

Salhi, S., & Nagy, G. (1999). A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling. *The Journal of the Operational Research Society*, *50*(10), 1034–1042.

Schneider, M., Stenger, A., & Goeke, D. (2014). The Electric Vehicle Routing Problem with Time Windows and Recharging Stations. *Transportation Science*, *48*(4), 500–520.

Ting, C., & Chen, C. (2013). A multiple ant colony optimization algorithm for the capacitated location routing problem. *Intern. Journal of Production Economics*, *141*(1), 34–44.

Vidal, T., Battarra, M., Subramanian, A., & Erdoŀan, G. (2015). Hybrid metaheuristics for the Clustered Vehicle Routing Problem. *Computers and Operations Research*, *58*, 87–99.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, *60*(3), 611–624.

YuB,Yang ZZ & Yao BZ (2009). An improved ant colony optimization for vehicle routing problem. EurJOplRes 196:171–176.

Yu, B., Yang, Z.-Z., & Xie, J.-X. (2011). A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, *62*(1), 183–188.

Zhang, S., Gajpal, Y., & Appadoo, S. S. (2017). A meta-heuristic for capacitated green vehicle routing problem. *Annals of Operations Research*, 1–19.