International Journal of Industrial Engineering Computations 6 (2015) 81–98

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A generalized multi-depot vehicle routing problem with replenishment based on LocalSolver

Ying Zhang[a,b*], Mingyao Qi[b], Lixin Miao[b] and Guotao Wu[b]

[a]*Department of Industrial Engineering, Tsinghua University, Beijing 100084, China*
[b]*Research Center on Modern Logistics, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China*

C H R O N I C L E

Article history:
Received July 6 2014
Received in Revised Format August 5 2014
Accepted August 27 2014
Available online August 28 2014

Keywords:
Vehicle routing
Multi-depot
Replenishment
Generalized model
Local search

A B S T R A C T

In this paper, we consider the multi depot heterogeneous vehicle routing problem with time windows in which vehicles may be replenished along their trips. Using the modeling technique in a new-generation solver, we construct a novel formulation considering a rich series of constraint conditions and objective functions. Computation results are tested on an example comes from the real-world application and some cases obtained from the benchmark problems. The results show the good performance of local search method in the efficiency of replenishment system and generalization ability. The variants can be used to almost all kinds of vehicle routing problems, without much modification, demonstrating its possibility of practical use.

1. Introduction

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem, which focuses on the optimal arrangement or schedule of a fleet of vehicles while serving scattered customers. The interest on VRP is indeed motivated by its practical relevance and considerable difficulty. Since first proposed in Dantzig (1959), hundreds of papers have considered all the main variants of this problem for which both exact and heuristic approaches are proposed: the capacitated VRP (CVRP), the VRP with time windows (VRPTW), the multi-depot VRP (MDVRP), the VRP with Backhauls (VRPB), the open VRP (OVRP), the pickup and delivery problem (PDPTW) and the site-dependent VRP (SDVRP), just to mention the most important ones. A complete overview of the state-of-the-art on VRP is given in the book by Toth and Vigo (2001), for a comprehensive survey of both construction method and heuristic approaches, see Bräysy and Gendreau (2005a, 2005b).

However, some aspects that arise in real application have not received much attention in the Operations Research literature. For instance, vehicles may perform more than one trip during a given work shift. This may happen when either customer demands are relatively large with respect to vehicle capacity, hence few customers may be served in a single route, or when tight time windows or short duration are imposed. In addition, in many cases the number of available vehicles is supposed to be limited, and

* Corresponding author. Tel: +86-755-26036349 Fax: +86-755-26036005
E-mail: yingzhang8996@gmail.com (Y. Zhang)

doi: 10.5267/j.ijiec.2014.8.005

there may be multi-depots so that vehicles may be replenished along their trips. When the vehicle capacity is small or when the planning period is large, replenishment may be the only practical solution. In urban areas, where travel times are rather small, is often the case that after performing short tours vehicles are reloaded and reused.

In recent years, there has been an increasing interest towards so-called “rich” VRP models. For example, Pisinger and Ropke (2007) demonstrated that all problem variants, including VRPTW, CVRP, MDVRP, SDVRP and OVRP, could be transformed into a rich pickup and delivery model and solved in the adaptive neighborhood search (ALNS) framework and the implementations were discussed by Ropke and Pisinger (2006). Considering the important issues arising in real-world applications, there is not an efficient generalized model dealing with replenishment in the rich VRP cases for the time being.

In this paper, we describe a novel formulation for the generalized multi-depot vehicle routing problem with replenishment and multiple vehicles, and so generate new and interesting families of optimization problems. The next section lists some relevant literatures. The problem definition and mathematical formulation are discussed in section 3, followed by the experimental analysis using a new-generation solver in section 4. Finally, conclusions and future work are considered in section 5.

2. Background

2.1 Relevant Literatures

Some research has focused on the specific and simplified versions of VRP with multiple trips. Azi et al. (2007, 2010, 2012) considered a variant of the VRPTW where each vehicle could perform several routes during its workday. This series of problems are inspired by the home delivery of perishable goods, where routes are of short duration, i.e. the last customer in each route must be served within a given time limit from the route start time. To avoid the high costs associated with the management of a large fleet, a solution is to reuse each vehicle and to allow it to perform multiple delivery routes over the horizon. Azi et al. (2007) considered the identical vehicles and proposed a method based on an elementary shortest path algorithm. Azi et al. (2010) studied a heterogeneous fleet of vehicles. Azi et al. (2012) considered the dynamic case where new customer requests occurred dynamically. They showed the benefits of allowing multiple routes and accounting for future customer requests when deciding the acceptance of a new request.

The vehicle routing problem with multiple uses of vehicles (VRPM) has been addressed through various heuristic means. Fleischmann (1990) proposed a heuristic based on the savings principle for route construction combined with a bin packing procedure for the assignment of routes to vehicles. Taillard et al. (1996) also used the bin packing procedure to assign routes to vehicles and developed an adaptive tabu search heuristic. Other heuristics have also been designed for VRPM, Battarra et al. (2009) proposed an iterative solution approach based on the decomposition method, Olivera and Viera (2007) described a heuristic based on the adaptive memory procedure, Salhi and Petch (2007) addressed a hybrid genetic algorithm using a new non-binary chromosome representation, Lin and Kwok (2006), meanwhile, considered the location of depots and applied metaheuristics of tabu search and simulated annealing, Petch and Salhi (2003) used a multi-phase constructive heuristic, Brandao and Mercer (1998) designed a genetic algorithm, Brandão and Mercer (1997) described a novel tabu search heuristic, Cattaruzza et al. (2012) proposed a hybrid genetic algorithm. All of these problems are solved using heuristic approaches. To the best of our knowledge, only Azi et al. (2010) and Mingozzi et al. (2013) adopted the exact algorithm. Azi et al. (2010) introduced a branch-and-price approach where lower bounds were computed by solving the linear relaxation of a set packing formulation. They were able to routinely solve instances with 25 customers and a few instances with up to 50 customers. Mingozzi et al. (2013) described two set-partitioning-like formulations for the VRPM and studied valid lower bounds based on the linear relaxations. Computational results showed that their proposed exact algorithm could solve instances with up to 120 customers. However, the rich constraints which real-life applications often encountered were not well reflected.

A few literatures also focus on the multi-depot VRPM case. Crevier et al. (2007) studied the Multi-Depot Vehicle Routing Problem with Inter-Depot Routes (MDVRPI) in which vehicles might be replenished at intermediate depots along their route. They proposed a heuristic combining the adaptive memory principle, a tabu search method for the solution of sub problems, and integer programming. Sevilla and de Blas (2003) presented a heuristic algorithm based on neuronal networks and ant colony system. Angelelli and Grazia Speranza (2002) studied the periodic vehicle routing problem with intermediate facilities (PVRP-IF) where the vehicles might renew their capacity at some intermediate facilities. They proposed a tabu search algorithm and extended this method to the waste collection problems, which were the realistic applications.

Collection of waste is part of reverse logistics operations dealing with the flow from the customers to recycling or disposal facilities. The waste collection problem consists of routing vehicles to collect customers waste within given time window while minimizing travel cost. The waste collection vehicle routing problem with time windows (WCVRPTW) concerns with finding cost optimal routes for garbage trucks such that all garbage bins are emptied and the waste is driven to disposal sites while respecting customer time windows. WCVRPTW differs from the traditional VRPTW by that the waste collection vehicles must empty their load at disposal sites and drivers are given the breaks that the law requires. Multiple trips to disposal sites are allowed for the vehicles.

The WCVRPTW has received some attention in recent years. Tung and Pinnoi (2000) considered only one disposal site and formulated the problem into a mixed-integer program, where they modified Solomon's insertion algorithm (Solomon, 1987) and applied it to a waste collection problem in Hanoi, Vietnam. Kim et al. (2006) focused on the commercial waste collection problem with consideration of multiple disposal trips and drivers' lunch breaks. They extended Solomon's well-known insertion approach and a capacitated clustering-based algorithm to improve the route compactness and workload balancing. Ombuki-Berman et al. (2007) studied the same problem using a multi-objective genetic algorithm. Benjamin and Beasley (2010) produced better quality solutions for publicly available waste collection problems using combination of tabu search and variable neighborhood search. Buhrkal et al. (2012) studied the WCVRPTW and gave a linear programming formulation. They proposed an ALNS algorithm and tested it on a set of instances from literature as well as on instances provided by a Danish garbage collection company. Only this paper has a detailed formulation (they didn't solve it), but since each of the disposal sites may be visited more than once, so the decision variable w_{ik} which represents the start time of service at node i by vehicle k may be improper. Moreover, none of an exact algorithm is proposed for this problem, which inspires us to establish a generalized formulation for this category of problem.

2.2 Why we choose LocalSolver

Current integer or constraint programming solvers are mainly based on Tree Search (branch-and-bound, branch-and-cut, branch-cut-price). Tree-search techniques consist in exploring the solution space by recursively instantiating variables composing a solution vector. Running in exponential time, the main drawback is to be limited to small and medium-scale problems. Moreover, if not terminating, tree search offers no more guarantee on the solution quality than any heuristic approach.

In contrast, Local Search consists in applying iteratively some local changes, called moves, to a solution to improve the objective function. LocalSolver is such a math programming solver that primal feasible solutions are computed by pure & direct local-search techniques (Benoist et al., 2011). Relying on local search, LocalSolver is able to scale up to 10 million binary decision variables.

For ultra-large real-life combinatorial problems, especially highly nonlinear 0-1 models, LocalSolver will provide high-quality solutions in very short running times without any tuning. The perfect performance is easily shown on car sequencing, nurse rostering, job shop scheduling and quadratic assignment. Many real-life VRP involves thousands of 0-1 decisions variables, which are out of tree search scope. Considering the considerable complexity of proposed problem, to produce a high quality solution in a short time, LocalSolver is no doubt a better choice, compared with tree search techniques.

Readers can refer to http://www.localsolver.com/ for more information.

3. Formulation

3.1 Problem Description

This problem is inspired from a real-life VRP related to manufacturing enterprise requirements. This is only for daily vehicle routing optimization. There are several warehouses, which can provide multiple product types and several vehicles with different capacity. Each vehicle parks at one warehouse at the beginning of the day, and rest at the specified warehouse (maybe another one) at the end of day. At the beginning of the day, each warehouse updates its available product inventory and the customer orders are collected. Every vehicle needs to refill at the warehouse, then visit customers for unloading. Refill (at warehouses) and unloading (at customers) requires some time, which is equal to refill/unload quantity * refill/unload speed. Some customers require that only selected vehicles can serve. By default, any vehicle can visit any customer. Each customer needs only one type of product, which can be satisfied by several warehouses that store this product. Moreover, vehicles are allowed and encouraged to re-use, i.e., make multiple trips to warehouse and customers. The overall goal is to maximize the number of customers whose demands are satisfied and minimize the traveling cost (weighted distance).

First, we give the generalized graph representation. This problem is defined on a directed graph $G = (N, A)$ where the set of nodes $N = C \cup W$ consists of $|C|$ customers $C = \{1,2, \cdots, |C|\}$ and $|W|$ warehouses $W = \{|C| + 1, |C| + 2, \cdots, |C| + |W|\}$ and the set of arcs is $A = \{(i, j)|i, j \in N, i \neq j\}$. Each node $i \in N$ has an associated time window $[ET_i, LT_i]$, where ET_i and LT_i are the earliest and latest time, respectively, to start the service. Thus, a vehicle has to wait if it arrives at node i before ET_i. Each node $i \in N$ also has a lunch time $[a_i, b_i]$, during which the service (loading or unloading) cannot be proceeded. Each service or dwell time s_i consists of preparation time and corresponding refill/unloading time. With each arc $(i, j) \in A$ is associated a distance d_{ij}. We define q_i as the demand of a customer $i \in C$. We also have a set V of vehicles to deliver goods from the warehouse to customers. It is assumed that each vehicle $k \in V$ has an associated capacity Q_k. The duration of each route is limited by forcing the last customer to be served within t_k time units of the route start time.

The objective is threefold, including maximize the number of customers served, minimize the weighted travel distance and maximize the loading rate, while satisfying the time window of each nodes and loading capacity and time duration of each vehicle.

3.2 Formulation

Some notations, which will be used in the following sections are listed in Table 1. In this Table, "Nodes" is the common properties of "Customers" and "Warehouses", $N_0 = \{0\} \cup N = \{0\} \cup C \cup W = \{0,1,2 \cdots, |C|, |C| + 1, |C| + 2, \cdots, |C| + |W|\}$ is an ordered set, which means that the customers are put before the warehouses and the first number 0 is reserved, the reason of which will be explained later.

We will give a simple example before the model establishment. Distribution with replenishment is very common in daily transport, especially in perishable goods transportation, where the duration of each trip is very short. So the vehicles need to be replenished at the nearest warehouse to continue serving customers during its work shift. A "*trip*" is the path starts and ends at two warehouses (whether they are the same or not). The set of all trips assigned to a vehicle is called a "*route*" whose total duration cannot exceed a preset value t_k. For a system with 2 warehouses and 5 customers, the route of one vehicle may be:

W1(Refill 16)→C2(Unload 3)→C5(Unload 5)→C1(Unload 8)→W2(Refill 15)→C3(Unload 8)→C4(Unload 7)→W1

This vehicle is replenished twice, first at W1 and then at W2.

Table 1
Notations

Vehicles	u_k	Unload Speed
	v_k	Average Travel Speed
	Q_k	Capacity
	w_k^o	Parking Warehouse at the beginning of the day
	w_k^d	Rest Warehouse at the end of the day
	f_k	unit cost per kilometer
	τ_k	Maximal travel time
	t_k	Maximal Duration
	K_k	Setup cost
Customers	q_i	Quantity of demand
	p_i	Priority
Warehouses	C_j	Capacity
	r_j	Refill Speed
Nodes	$[ET_i, LT_i]$	Time windows of service
	$[a_i, b_i]$	Lunch Time
	d_{ij}	Distance time between i and j
	l_i	Preparation Time
Sets	C	All customers
	W	All warehouses
	V	All vehicles available
	$N = C \cup W$	All Nodes
	$N_0 = \{0\} \cup N$	An extended ordered set
	W_i'	Warehouses that stores the product of customer i
	V_i'	Vehicles that can serve customer i

3.2.1 Decision variables

In the basic VRP and VRPTW, we often define the binary variable $x_{ijk} = 1$ for $i, j \in N, k \in V$ iff vehicle k drives directly from node i to node j. However, this definition seems hard to represent our problem, for some nodes (warehouses) may be visited more than once.

The modeling techniques in the basic MDVRP and VRPM seem difficult to formulate this problem. For the former, the depot a vehicle starts from and returns to is fixed and known. By introducing the decision variable x_{ijmk}, which means that vehicle k based at depot m travels from node i to node j, the MDVRP model is easily obtained. While considering our problem from another point of view, as long as a vehicle arrives at a warehouse to be replenished, then it starts a new trip and travels to another warehouse. The trip between any two warehouses can be viewed as the so-called inter-routes. But we don't know exactly the start terminal and end terminal of the vehicle serving this trip. So both the distance and loading time are unknown.

For the latter, we define x_{ijkr}, which means vehicle k travels from node i to node j on its trip r. We don't know exactly how many trips a vehicle can travel per work shift. Meanwhile, the second trip is closely related to its predecessor: the start warehouse and start time both depend on the first one. But it is difficult to represent this relationship by this definition.

Confronted with these difficulties, we try to establish the model with replenishment from a new perspective. We define for each vehicle a route made of a predefined number of *sequences* (can be interpreted as *positions*). Each such sequence is assigned a node number. The sequences with index 0 code for "no visit", with indices from 1 to $|C|$ code for the visit to this customer and with indices from $|C| + 1$ to $|C| + |W|$ code for the visit to a warehouse. Suppose that the maximal number of nodes a vehicle can visit per day is s. The maximal value $s_{max} = 2|C|$, i.e., in extreme cases there is just one customer on each trip. In computation, we often set $s = |N_0| = 1 + |C| + |W|$ for simplicity.

Binary variable $x_{ink} = 1, \forall i \in N_0, \forall k \in V, n = 0,1, \cdots, s$ iff node i is assigned on sequence n of vehicle

k. The vehicle-customer constraint (some customers require that only selected vehicles can serve) and start/end terminal constraint (for each vehicle, start terminal is the warehouse this vehicle starts from at the beginning of the day, end terminal is the warehouse this vehicle returns to at the end of the day) can be both expressed by the definition of x_{ink} easily (Tab. 2).

Table 2
Definition of decision variables

i	*k*	*n*	Value
$i \in C$	$k \in V_i'$	$n = 1,2,\cdots,s-1$	*bool*
	$k \notin V_i'$		*0*
$i \in W$	$\forall k \in V$		*bool*
$\forall i \in N$	$\forall k \in V$	$n = 0$	$i == w_k^o$
$\forall i \in N$	$\forall k \in V$	$n = s$	$i == w_k^d$

This relationship can be simply represented by the "*if-else*" expression. Boolean decision variables are declared using the operator *bool*. "==" defines a boolean expression which takes value of 0 and 1, as in logic algebra. Note that if we relax the last value when $\forall i \in N, \forall k \in V, n = s$ from $i == w_k^d$ to *bool*, then it results as the so called OVRP.

3.2.2 Intermediate variables

Intermediate variables, also called modeling expressions, can be declared using the mathematical operators, such as Decisional, Arithmetic, Logical, Relational and Conditional in LocalSolver. They will help represent constraint conditions and objective functions.

y_{nk} means the node assigned to vehicle *k* on sequence *n*, z_{nk} denotes as the location of vehicle *k* on sequence *n* (because a sequence can be empty so the location is the previous actually assigned). Using the ternary operator ?: as in programming language such as C++, Java, etc., we have

$$y_{nk} = \min\left\{|N|, \sum_{i \in N} ix_{ink}\right\} \quad \forall k \in V, n = 0,1,\cdots,s \tag{1}$$

$$z_{nk} = \sum_{i \in N} x_{ink} == 1 ? y_{nk} : z_{n-1,k} \quad \forall k \in V, n = 1,2,\cdots,s \tag{2}$$

Boolean expression $\sum_{i \in N} x_{ink} == 1$ signifies whether there is a node assigned on sequence *n* of vehicle *k*. In LocalSolver, we can get the value of an array by the index of an expression. Using this feature, if we want to know the time windows or demand of the node assigned to vehicle *k* on sequence *n*, we can simply get the value of corresponding array by index of y_{nk}, for example $q_{y_{nk}}$.

3.2.3 Data Reprocessing

Due to the integration of warehouses and customers as nodes, we need to reprocess the input data before the definition of objective functions and constraint conditions.

From (1), if $i = 0$, then $ix_{ink} = 0$ whether the value of x_{ink} is 1 or not. In other words, the value of ix_{ink} has nothing to do with x_{ink} if current node is 0 ($i = 0$). Thus we reserve 0 in the element of set N_0 and arrange customers and warehouses from index 1. This is the reason why the definition of "sequences" says that index 0 codes for "no visit".

Then we integrate quantity demand, preparation time and time windows. If all the nodes (customers and warehouses) have this attributes, we take the corresponding values; otherwise we fill it with 0. For example, the vector of demand $q = \{0, q_1, q_2, \cdots, q_{|C|}, 0, 0, \cdots, 0\}$ and the vector of preparation time $l = \{0, l_1, l_2, \cdots, l_{|C|}, l_{|C|+1}, l_{|C|+2}, \cdots, l_{|C|+|W|}\}$.

W_i' is the set of warehouses that stores the product that customer i needs. Its value can be obtained simply by comparison in a loop.

3.2.4 Objective Functions

(1) Maximize number of customers served

For a distribution system with limited resources, we want to maximize the number of customers served. As denoted in the table, every customer has a priority p_i which takes integer values. The bigger the value, the more urgent the demand is. So want to arrange the customers with higher value of p_i to be served with greater priority. Thus the first objective function usually is

$$\max \sum_{i\in C} p_i \sum_{k\in V}\sum_{n=0}^{s} x_{ink} \tag{3}$$

If all customers have the same priority, i.e., $p_i = 1, \forall i \in C$, formula (3) is equivalent to maximize the number of customers served.

(2) Minimize total travel distance

Since in LocalSolver we can use an expression as the index of an array, we need to transform the distance matrix $[d_{ij}]$ to a one-dimensional array *D*. The following formula (4) may be an alternative method. Define dis_{nk} as the distance between two adjacent nodes assigned on sequence *n* and on sequence *n*-1 of vehicle *k*.

$$D_{i+|N_0|\cdot j} = D_{j+|N_0|\cdot i} = d_{ij} \tag{4}$$

$$dis_{nk} = D_{z_{nk}+|N_0|\cdot z_{n-1,k}} \tag{5}$$

By the introduction of dis_{nk}, the total travel distance can be expressed easily.

$$\min \sum_{k\in V}\sum_{n=0}^{s} dis_{nk} = \sum_{k\in V}\sum_{n=0}^{s} D_{z_{nk}+|N_0|\cdot z_{n-1,k}} \tag{6}$$

(3) Maximize loading rate

Actually, we should have needed to define a series of decision variables to describe the quantity of product a vehicle refills at a warehouse, which are continuous and highly rely on the subsequent customers this vehicle will serve. In the example introduced in section 3.2, the vehicle needs to pick up 16 at W1, just equal to the sum of demands of C2, C5 and C1. If we don't know which customers this vehicle will serve in advance, the quantity of replenishment is difficult to determine. If these decision variables defined, the optimization maybe time consuming. For simplicity, we assume that as soon as a vehicle arrives at a warehouse, refill to its whole capacity. Such simplification may lead to a suboptimal solution, but it is worth doing because the complexity is lowering down. We just need to adjust the solution (such as change the loading time) a bit in the output.

Back to this example, if the capacity of this vehicle is 20, we suppose that it is replenished to 20 both at W1 and W2. After optimization, we know that when it leaves W1, it serves C2, C5 and C1, so we change the replenishment quantity to 16. Through such simplification, when a vehicle arrives at a warehouse to be replenished, it maybe not empty, which leads to a low loading rate. In other words, to improve the loading rate, we need to minimize the loading before replenishment.

Define $q\_start_{nk}, q\_change_{nk}, q\_end_{nk}$ as the loading of vehicle *k* before arriving at sequence *n* (we ignore if this node is a customer since we only want to minimize the quantity before replenishment), changes at sequence *n* and leaving sequence *n*, respectively. Here "*change*" means the variation of the

loading of the vehicle.

$$q\_change_{nk} = (y_{nk} > |C|) \,?\, Q_k : -q_{y_{nk}} \tag{7}$$

$$q\_end_{nk} = (y_{nk} > |C|) \,?\, Q_k : q\_end_{n-1,k} + q\_change_{nk} \tag{8}$$

$$q\_start_{nk} = (y_{nk} > |C|) \,?\, q\_end_{n-1,k} : 0 \tag{9}$$

Their initial values are ignored. Customers' demands are changed to their opposite numbers. Boolean expression $y_{nk} > |C|$ is to determine whether the node on sequence *n* of vehicle *k* is a warehouse or not. With these intermediate expressions, to maximize the loading rate, we need to minimize

$$\min \sum_{k \in V} \sum_{n=0}^{s} q\_start_{nk} \tag{10}$$

(4) Minimize number of vehicles used

If any customer appears on the sequence (except the first and the last one, which are the terminals), the vehicle is used. We can represent this relationship by the logical OR operator.

$$isUsed_k = \bigvee_{i \in C} \bigvee_{n=1}^{s-1} x_{ink} \tag{11}$$

$$\min \sum_{k \in V} isUsed_k \tag{12}$$

If expression $isUsed_k$ is equal to 1, then vehicle *k* is used. Then Eq. (12) is the objective that minimize the number of vehicles used.

(5) Minimize total cost

For each vehicle, the total cost consists of the fixed cost when this vehicle is used and the weighted travel cost.

$$\min \sum_{k \in V} \left(isUsed_k \cdot K_k + f_k \cdot \sum_{n=0}^{s} dis_{nk} \right) \tag{13}$$

(6) Minimize number of replenishment

Finally, on the premise of maximizing the customers served, we want to minimize the number of replenishment. This is done by simply the minimization of the sum of x.

$$\min \sum_{i \in N} \sum_{k \in V} \sum_{n=1}^{s-1} x_{ink} \tag{14}$$

In LocalSolver, if multiple objectives are defined, they are interpreted as a lexicographic objective function. The lexicographic ordering is induced by the order in which objectives are declared. The objective that is defined earlier has a higher priority.

3.2.5 Relevant Constraints

(1) Internal Constraint

Each customer is served more than once

$$\sum_{k\in V}\sum_{n=1}^{s-1} x_{ink} \le 1 \quad \forall i \in C \tag{15}$$

No more than one node per vehicle per sequence.

$$\sum_{i\in N} x_{ink} \le 1 \quad \forall k \in V, n = 1,2,\cdots,s-1 \tag{16}$$

(2) Warehouse-Customer constraint

Let g_{nk} represent the last visited warehouse of vehicle *k* on sequence *n*.

$$g_{0k} = w_k^o \quad \forall k \in V \tag{17}$$

$$g_{nk} = y_{nk} > |C| \,?\, y_{nk} : g_{n-1,k} \quad \forall k \in V, n = 1,2,\cdots,s \tag{18}$$

As mentioned above, $|C|$ is the number of customers, if $y_{nk} > |C|$ is true, the current node is a warehouse, otherwise a customer. In the following algorithm (Tab. 3), we construct an array *compatible* to transform the warehouse-customer relationship to a one-dimensional array. For the case with 5 customers and 2 warehouses, *compatible* has $2 \times (5 + 2) = 14$ elements. If the values of the first 7 elements are 1, it means that the corresponding node could be served by the first warehouse. So do the second 7 elements. The construction method can be described as follows:

Table 3
Procedures to generate the compatible array

Procedure: Compatible array generation

1. Initial the array $Compatible = \emptyset$.
2. **for** $j = 1:|W|$
3. **for** $i = 1:|C|$
4. **if** $(j \in W_i')$
5. $Compatible = Compatible \cup \{1\}$;
6. else
7. $Compatible = Compatible \cup \{0\}$;
8. end if
9. end for
10. **for** $i = |C| + 1:|W|$
11. **if**$(i - |C| == j)$
12. $Compatible = Compatible \cup \{1\}$;
13. else
14. $Compatible = Compatible \cup \{0\}$;
15. end if
16. end for
17. end for
18. return $Compatible$.

Compatible is an array takes value of 0 and 1. From section 3.1, each customer needs only one type of product, which can be satisfied by several warehouses that store this product. So when a vehicle is reloaded one certain type of product at a warehouse, it can then only serve those customers that need

this product, we rename this constraint as the so-called “warehouse-customer constraint”. To achieve this goal, we constraint that the value of *compatible* with index $z_{nk} - 1 + |N| \times (g_{nk} - |C| - 1)$ is 1.

(3) Vehicle Capacity Constraint

Any time the loading quantity of the vehicle should be greater than 0.

$$q\_end_{nk} \geq 0 \quad \forall k \in V, n = 0,1,\cdots,s \tag{19}$$

(4) Warehouse Capacity Constraint

No stock out is allowed in the warehouses.

$$\sum_{k \in V} \sum_{n=0}^{s-1} x_{ink} q\_change_{nk} \leq C_i \quad \forall i \in W \tag{20}$$

(5) Time Windows Constraint

Each node $i \in N$ is associated a time window $[ET_i, LT_i]$ and lunch time $[a_i, b_i]$. Thus, a vehicle has to wait if it arrives at i before ET_i. And the service must finish before LT_i and cannot span over $[a_i, b_i]$. So if the service is not able to be finished before a_i, the time to start the service is put off to b_i.

For sequence n on vehicle k, $preTime_{nk}$ is the preparation time, $loadTime_{nk}$ is the refill/unload time, $isFinished_{nk}$ denotes as whether the service can be finished before the lunch time or not, $timeStart_{nk}$ and $timeEnd_{nk}$ are the corresponding time to start the service and end the service, respectively.

$$preTime_{nk} = \sum_{i \in N} x_{ink} == 1 \,?\, l_{y_{nk}} : 0 \tag{21}$$

$$loadTime_{nk} = \frac{abs(q\_change_{nk})}{(y_{nk} > |C|) \,?\, r_{y_{nk}} : u_k} \tag{22}$$

$$lunchBegin_{nk} = \max\left\{timeEnd_{n-1,k} + T_{nk}, ET_{y_{nk}}\right\} \tag{23}$$

$$lunchEnd_{nk} = lunchBegin_{nk} + preTime_{nk} + loadTime_{nk} \tag{24}$$

$$isFinish_{nk} = lunchBegin_{nk} < b_{y_{nk}} \;\&\&\; lunchEnd_{nk} > a_{y_{nk}} \tag{25}$$

$$timeStart_{nk} = isFinish_{nk} \;?\; b_{y_{nk}} : lunchBegin_{nk} \tag{26}$$

$$timeEnd_{nk} = timeStart_{nk} + preTime_{nk} + load_{nk} \tag{27}$$

$$timeEnd_{nk} \leq LT_{y_{nk}} \quad \forall n = 0,1,\cdots,s; \forall k \in V \tag{28}$$

The nonlinearity is shown in Eq. (22). Refill speed depends on the warehouses, while unload speed depends on the vehicles. So the ternary operator ?: appears in the denominator. In Eq. (23), T_{nk} is the travel time between the nodes assign to sequence *n* and *n*-1 of vehicle *k*. Its value can be determined by the array *D* which is defined in Eq. (4) and the speed of vehicle *k*. Eqs.(21-27) are series of definition. (28) is the constraint to ensure the service ends before the time window is closed.

(6) Maximal Duration Constraint

The duration of a route is the time interval between the start time at the first node and the end time at the last node.

$$timeEnd_{sk} - timeStart_{0k} \leq t_k \quad \forall k \in V \tag{29}$$

(7) Maximal Driving time Constraint

The driving time of a vehicle is the sum of all traveling time between two adjacent nodes.

$$\sum_{n=0}^{s} T_{nk} \le \tau_k \quad \forall k \in K \tag{30}$$

(8) Additional Acceleration Strategy

We constraint that a vehicle cannot travel directly from a warehouse to another warehouse, providing that stock out never happens. The following two constraints reduce the solution space.

$$Con_{nk} = z_{nk} > |C| \,\&\&\, z_{n+1,k} > |C| \,\&\&\, z_{nk} \neq z_{n+1,k} \quad \forall k \in K; n = 0,1,\cdots,s-1 \tag{31}$$

$$Con_{nk} == 0 \quad \forall k \in V; n = 0,1,\cdots,s-1 \tag{32}$$

4. Computation Results

4.1 Real-life VRP

In this section, we give an example which is inspired from a real-life VRP. This example has 3 warehouses, 4 heterogeneous vehicles and 25 customers. At the beginning of the day, the customers' orders are collected (including ones that were unmet the day before). Each vehicle starts from the specific warehouse, serves customers and returns to the given warehouse when its total duration is reached, or the time window of its end terminal will "close".

Table 4
Data of warehouses

Name	Latitude	Longitude	Capacity(T)	Time Window	Preparation Time(min)	Refill Speed (T/min)	Product Type
W1	29.907	121.679	200	5:00-21:00	20	0.3	a,b,c
W2	30.427	120.768	100	6:00-20:00	20	0.4	b,c
W3	30.251	120.582	100	5:00-19:00	20	0.4	a,b

Table 5
Data of vehicles

Name	Unload Speed (T/Min)	Cost per km	Speed (Km/Hr)	Capacity(T)	Time Window	Maximal travel distance(km)	Duration (h)	Terminal
V1	0.5	6.5	60	32	5:00-22:00	600	15	W1
V2	0.6	6.5	50	25	5:00-22:00	600	15	W2
V3	0.5	6.5	40	32	5:00-22:00	600	15	W3
V4	0.5	6.5	60	30	5:00-22:00	600	15	W1

This example has many constraints, to the best of our knowledge, in the static VRP scope no paper has solved such a complex problem. But in LocalSolver, it is a quite easy issue. Given the unit of time in minutes (min) and distance in kilometers (km), we have

$N_0 = \{0,1,2,3,\cdots,24,25,26,27,28\}, w_1^o = 26, w_1^d = 26, W_1' = \{26,27,28\}, V_1' = \{28\}$
$startTime = \{480, 420,480,\cdots,480,360,300,360,300\}.$

Table 6
Data of customers

Nam	Latitud	Longitud	Demand	Typ	Time	Lunch Time	Preparation	Priorit	Vehicles
C1	29.628	120.832	25	b	08:00-17:00	11:30-13:00	20	1	V3
C2	29.891	121.786	10	b	07:00-17:00		20	1	
C3	29.907	121.818	10	b	08:00-17:00		20	1	
C4	29.876	121.637	10	a	08:00-17:00	11:30-13:00	20	1	V1
C5	29.872	121.636	10	a	06:00-23:59	7:00-9:00	20	1	
C6	28.704	121.571	20	c	08:00-17:00		20	1	
C7	30.186	120.535	11	c	08:00-17:00	11:30-13:00	20	1	
C8	29.951	121.498	13	b	08:00-17:00	11:30-13:00	20	1	V2,V3
C9	29.940	120.351	10	b	08:00-17:00	11:30-13:00	20	1	
C10	29.807	121.662	10	b	08:00-17:00		20	1	
C11	29.957	121.723	10	b	08:00-17:00	11:30-13:00	20	1	
C12	29.836	121.457	6	b	08:00-17:00		20	1	V3,V4
C13	29.835	121.457	4	b	07:00-17:00		20	1	V2
C14	29.931	121.830	4	b	08:00-17:00	11:00-13:00	20	1	
C15	30.308	120.034	10	b	08:00-16:00	11:00-13:00	20	1	
C16	30.240	120.385	8	c	08:00-17:00	11:30-13:00	20	1	
C17	29.894	121.801	6	b	08:00-17:00		20	1	
C18	29.836	121.554	10	c	06:00-23:59	7:00-22:00	20	1	V1,V3
C19	29.769	121.534	15	b	06:00-20:00		20	1	
C20	29.806	121.595	10	b	08:00-17:00	11:00-13:00	20	1	
C21	28.732	121.614	25	c	08:00-17:00		20	1	
C22	28.732	121.614	10	c	08:00-17:00		20	1	
C23	29.699	121.422	15	b	06:00-17:59		20	1	
C24	29.918	121.639	20	a	08:00-17:00		20	1	
C25	29.917	121.867	10	a	06:00-17:59		20	1	V1,V2

Notes: the empty cells in the columns of “Lunch time” refer to “no lunch is required”, while in the columns of “Vehicles only” refer to “all vehicles can serve”.

Take (3), (12), (6), (10) and (14) as the five objective functions, whose priorities decrease in order. Set the time limit of each objective to 10 seconds, the statistical result of the 4 vehicles are:

Table 7
Route information

Vehicle	Terminal	Departure Time	Return Time	Travel Distance(km)
V1	W1	5:00:00	17:51:14	304.1862
V2	W2	6:00:00	18:36:32	267.0346
V3	W3	5:00:00	18:02:21	270.4916
V4	W1	5:00:00	20:39:30	410.9691

Customers C1 and C21 are not served. For C1, we observe that only V3 can serve it. On the last trip, V3 carries loading of 31 (with its capacity of 32) to serve C10, C19 and C12. It is 15:38:04 when the service of C12 is finished. The distance between C12 and the nearest warehouse W1 is 22.84km. So if V3 travels to W1 to be replenished, and then serves C1 and then returns to W3, the total travel time on path $\mathrm{C12} \rightarrow \mathrm{W1} \rightarrow \mathrm{C1} \rightarrow \mathrm{W3}$ is $\frac{(22.84+87.54+73.42)km}{40km/h} = 4.6h$. The total replenishment time and total service duration is $\left(20 + \frac{25}{0.3} + 20 + \frac{25}{0.5}\right) min = 2.89h$. So when V3 arrives at W3, it is 23:07:24. However the last time V3 should return is 19:00.

Actually, the time when V3 arrives at C1 is 20:07, while the time window of C1 is 08:00-17:00. So anyway, the demand of C1 cannot be satisfied.

The advantage of using LocalSolver is that a good solution can be obtained in a very short time, even if the constraint conditions are very complex.

Table 8
Route details

	Node	Quantity change(T)	Distance	Travel Time	Arrive Time	Dwell Time(min)	Leave Time
V1	W1	30	0	0	5:00:00	0	5:00:00
1.1	C18	-10	14.419	14.419	5:14:25	85.581	6:40:00
1.2	C20	-10	5.1905	5.1905	6:45:11	114.8095	8:40:00
1.3	C4	-10	8.788	8.788	8:48:47	40	9:28:47
1.4	W1	30	5.2592	5.2592	9:34:02	126.6667	11:40:42
1.5	C6	-20	134.3588	134.3588	13:55:04	60	14:55:04
1.6	C22	-10	5.2465	5.2465	15:00:19	40	15:40:19
1.7	W1	0	130.9242	130.9242	17:51:14	0	17:51:14
V2	W2	13	0	0	6:00:00	0	6:00:00
2.1	C8	-13	87.9675	105.561	7:45:33	56.10567	8:41:39
2.2	W1	20	18.1478	21.77736	9:03:26	103.3333	10:46:46
2.3	C24	-20	4.0274	4.83288	10:51:36	53.33333	11:44:56
2.4	W1	24	4.0274	4.83288	11:49:46	103.3333	13:33:06
2.5	C14	-4	14.8023	17.76276	13:50:52	26.66667	14:17:32
2.6	C25	-10	3.9081	4.68972	14:22:13	36.66667	14:58:53
2.7	C17	-6	6.8343	8.20116	15:07:05	30	15:37:05
2.8	C13	-4	33.8231	40.58772	16:17:41	26.66667	16:44:21
2.9	W2	0	93.4967	112.19604	18:36:32	0	18:36:32
V3	W3	30	0	0	5:00:00	0	5:00:00
3.1	C11	-10	114.6937	172.04055	7:52:02	47.95945	8:40:00
3.2	C3	-10	10.7318	16.0977	8:56:05	40	9:36:05
3.3	C2	-10	3.6002	5.4003	9:41:29	40	10:21:29
3.4	W1	31	10.4642	15.6963	10:37:11	126.6667	12:43:51
3.5	C10	-10	11.2055	16.80825	13:00:40	40	13:40:40
3.6	C19	-15	13.0705	19.60575	14:00:16	50	14:50:16
3.7	C12	-6	10.5324	15.7986	15:06:04	32	15:38:04
3.8	W3	0	96.1933	144.28995	18:02:21	0	18:02:21
V4	W1	29	0	0	5:00:00	0	5:00:00
4.1	C7	-11	114.4641	114.4641	6:54:27	107.5359	8:42:00
4.2	C16	-8	15.7226	15.7226	8:57:43	36	9:33:43
4.3	C15	-10	34.4924	34.4924	10:08:12	40	10:48:12
4.4	W3	25	53.0195	53.0195	11:41:14	95	13:16:14
4.5	C9	-10	41.1782	41.1782	13:57:24	40	14:37:24
4.6	C23	-15	106.9186	106.9186	16:24:19	50	17:14:19
4.7	W1	10	33.8931	33.8931	17:48:13	120	19:48:13
4.8	C5	-10	5.6403	5.6403	19:53:51	40	20:33:51
4.9	W1	0	5.6403	5.6403	20:39:30	0	20:39:30

4.2 Benchmark Problem

4.2.1 VRPTW with replenishment

A large number of approaches, including exact algorithms and metaheuristics, have been proposed for solving the VRPTW. Most of these methods were applied to the Solomon benchmark problems. This data set contains 56 instances, each of which has 100 customers and a single depot and a homogeneous fleet of vehicles. Most of the proposed algorithms use vehicle minimization as primary objective and travel distance minimization as secondary objective. But to the best of our knowledge, few articles consider the multiple use of vehicles.

To verify the efficiency of our formulation, we halve the vehicles' capacity. Hence the number of customers on a single trip a vehicle can serve is limited. As a consequence, vehicles have to return to the depot to be replenished and continue distribution. We just take the first two instances i.e., C101 and C102, as examples. C101 needs 12 vehicles, while C102 still needs 10 vehicles. The total travel distances are 2017.633 and 1984.97, respectively.

C101
V0: 0 20 33 31 35 37 **0** 28 26 23 22 21 0
V1: 0 90 87 86 94 **0** 38 39 36 34 52 0
V2: 0 67 78 76 71 70 73 **0** 6 4 75 0
V3: 0 24 7 8 15 30 0
V4: 0 57 55 54 44 **0** 16 14 12 0
V5: 0 43 42 **0** 83 82 58 60 59 69 0
V6: 0 13 17 27 29 11 9 **0** 88 89 91 0
V7: 0 5 3 18 19 84 77 79 80 0
V8: 0 98 96 95 10 **0** 46 45 48 51 50 49 47 0
V9: 0 32 41 40 **0** 74 72 61 64 68 66 0
V10: 0 81 63 62 **0** 92 93 97 99 0
V11: 0 65 25 53 56 **0** 85 100 2 1 0

C102
V0: 0 20 24 8 10 **0** 46 61 64 66 69 0
V1: 0 26 17 18 19 15 **0** 45 48 51 50 59 72 0
V2: 0 78 76 71 70 84 **0** 88 95 96 12 0
V3: 0 81 63 62 **0** 29 38 39 36 52 49 47 0
V4: 0 13 25 27 **0** 56 58 60 68 **0** 31 37 34 0
V5: 0 32 33 **0** 94 92 93 97 **0** 89 85 91 2 0
V6: 0 57 55 **0** 40 44 73 77 79 80 82 83 0
V7: 0 90 87 86 74 **0** 16 14 23 22 21 0
V8: 0 43 42 41 35 **0** 11 9 100 99 98 1 75 0
V9: 0 67 65 54 53 **0** 30 28 6 4 7 3 5 0

The number “0” in the route is marked as bold to represent the replenishment. As an example, we find that “V5” of C102 is replenished two times, the start time to service each node is listed in the brackets to show the feasibility.

V5: 0(0) → 32(31.62) → 33(123.62) → 0(247.15) → 94(287.76) → 92(381.36) → 93(475) → 97(570) → 0(700.31) → 89(737) → 85(832.39) → 91(930.39) → 2(1038.41) → 0(1149.03)

4.2.2 MDVRP with replenishment

MDVRPTW considers cases where there are multiple depots. It aims at designing a set of minimum cost routes for a vehicle fleet serving many customers with known demands and predefined time windows. Each vehicle departs from a depot to visit customers, follow its route and finally returns to the depot where it starts. The cost of a solution is defined as the total distance traveled by the vehicles.

Lots of literatures studied variants for MDVRP. Xu et al. (2012) studied the multi depot heterogeneous vehicle routing problem with time windows, using a modified variable neighborhood search (VNS) algorithm. Kuo and Wang (2012) proposed a VNS heuristic for the MDVRP with loading cost. Gulczynski et al. (2011) developed an integer programming-based heuristic for the multi-depot split delivery vehicle routing problem. Wu et al. (2002) combined the location-allocation problem, where several unrealistic assumptions, such as homogeneous fleet type and unlimited number of available vehicles were relaxed. Cordeau et al. (1997) proposed a tabu search heuristic capable of solving periodic and MDVRP.

In these studies, each customer is visited by a vehicle based at one of these depots. To the best of our knowledge, few papers consider the cases that vehicles can perform multiple trips, let alone be replenished in other depots. Jordan et al. (1984, 1987) assumed that customer demands were all equal to vehicles’ capacity and that vehicles might travel back-and-forth between two depots. Angelelli and Grazia Speranza (2002) and Crevier et al. (2007) studied the MDVRP with intermediate facilities and inter-depot routes, respectively, as already introduced in section 2. But they ignored the time windows constraints.

To test the performance of our formulation in the multiple depot case, we construct two instances based on Cordeau et al. (1997) available on website http://www.bernabe.dorronsoro.es/vrp/.

The capacity of each vehicle are divided by 2.5 to make a trip much “shorter”. The results are shown below.

Pr01
V0: 49 35 44 31 41 7 22 37 49
V1: 50 34 10 **50** 45 6 27 3 48 11 50
V2: 51 13 33 4 19 14 28 1 51
V3: 51 20 29 8 5 **51** 26 25 17 18 16 51
V4: 52 9 42 46 39 **52** 2 15 23 36 32 43 52
V5: 52 47 24 **52** 30 12 21 38 40 52

Pr02
V0: 97 41 86 20 19 **97** 73 16 64 17 97
V1: 97 8 43 63 77 90 45 70 59 84 97
V2: 97 81 62 37 69 **98** 78 88 33 9 97
V3: 98 96 **99** 55 92 68 27 74 44 94 98
V4: 98 65 60 25 **97** 72 87 32 1 98
V5: 98 48 51 76 3 12 66 56 22 47 98
V6: 99 10 6 24 14 18 99
V7: 99 15 67 50 80 2 **99** 42 85 36 53 83 71 99
V8: 99 93 38 39 7 5 99
V9: 100 79 75 40 34 4 13 61 100
V10: 100 21 57 54 11 **100** 89 31 49 82 35 100
V11: 100 29 95 46 30 23 26 **100** 52 58 28 91 100

Table 9
Comparison result

Instance	Original Solution		Modified Solution (Vehicle capacity is divided by 2.5)	
	Number of vehicles	Total travel distance	Number of vehicles	Total travel distance
Pr01	6	1083.98	6	1239.12
Pr02	12	1763.07	12	2471.03

The characteristic of MDVRP is that each customer is visited by a vehicle based at one depot and each route starts and ends at the same depot. If we allow a vehicle to be reused during its work shift, then replenished at its "own" depot maybe suboptimal. Take V4 of Pr02 as an example. This vehicle starts from W_{98}, but is also replenished at W_{97}.

The comparison with the solution of the original data is listed in Tab. 9. Since the vehicle capacities are not the same, there is little comparability in fact. The results just show that we can also ensure the feasibility even though vehicles carry much less per trip.

4.3 Optimality Test

LocalSolver searches a better solution with heuristic moves, to test the optimality, we examine a set of benchmark instances for VRP with replenishment. Crevier et al. (2007) studied the MDVRPI where the route of a vehicle could be composed of multiple stops at intermediate depots in order for the vehicle to be replenished. They developed a heuristic and designed a set of benchmark instances for this problem. This set contains 12 randomly generated instances. Each instance has multiple depots, a fleet of homogeneous vehicles and several customers whose demands must be satisfied. The coordinates of the central depot, the one each vehicle starts from and ends at, are set equal to the average coordinates of the other depots. Furthermore, the refill time at the depot and the unload time at customers are proportional to the corresponding quantity of goods. The duration to serve each node is the sum of preparation time and refill/unload time. Each vehicle has an associated capacity and maximum duration. They assumed that each customer can be visited by any vehicle and none of the nodes have time windows constraints and lunch time constraints, which are much easier than ours. In our test, the time limit for each objective function is 15 seconds. The results obtained by LocalSolver as well as those by Crevier et al. (2007) are listed in Table 10.

Table 10
Comparison results

Instance	*r*	*n*	*m*	*D*	*Q*	Crevier et al.	LocalSolver
a1	3	48	6	550	60	1179.79	1224.99
b1	3	96	4	1200	210	1217.07	1319.24
c1	3	192	5	1850	360	1886.15	2408.28
d1	4	48	5	600	80	1059.43	1085.87
e1	4	96	5	1300	230	1309.12	1501.53
f1	4	192	4	2000	380	1576.33	1894.25
g1	5	72	5	750	80	1181.13	1264.58
h1	5	144	4	1550	230	1547.25	1812.09
i1	5	216	4	2350	380	1927.99	2484.82
j1	6	72	4	800	100	1120.65	1158.71
k1	6	144	4	1650	250	1586.92	1780.15
l1	6	216	4	2500	400	1884.92	2370.77
Average						1456.40	1692.10

r: number of depots; *n*: number of customer; *m*: number of vehicles;
D: maximum duration; *Q*: capacity of a vehicle

It seems that our solution is somewhat poorer in solution quality. The reason maybe that the algorithm in Crevier (2007) is problem-characteristic, while ours is just a generalized method. LocalSolver can get a not bad solution in nearly one second for all these instances, which make it more suitable to put into practical use. No matter how the input changes, we needn't make much modifications, for almost all variants of VRP.

5. Conclusions

There are several variant types of VRP. The open multi-depot heterogeneous vehicle routing problem with time windows in which vehicles may be replenished along their trips, which combines the MDVRP, VRPTW, OVRP and VRPM has not been addressed in the literature. In this paper, using the modeling features in LocalSolver, we construct a novel formulation considering a rich series of constraint conditions and objective functions. Computation results show that the mathematical model performed effectively in real-world applications.

Further, the formulation can be applied successfully without much modification to other variant VRPs with replenishment, such as VRPTW and MDVRPTW, while those problems imposing replenishment are mostly solved in metaheuristic methodology in literature. Comparative results demonstrate that the proposed formulation can also get a good solution in a very short time.

In future work, we could focus on developing efficient heuristic for solving the generalized VRP with replenishment and multiple trips, and integrate it in decision support system.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (71272030), as well as Science & Technology Foundation of Shenzhen City (CXZZ20130321145336439). The authors would like to thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the final version of the paper. They would also like to thank the Editors for their generous comments and support during the review process.

References

Angelelli, E., & Grazia Speranza, M. (2002). The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2), 233-247.

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

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

Azi, N., Gendreau, M., & Potvin, J. Y. (2007). An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European journal of operational research*, 178(3), 755-766.

Battarra, M., Monaci, M., & Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36(11), 3041-3050.

Benjamin, A. M., & Beasley, J. E. (2010). Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, 37(12), 2270-2280.

Benoist, T., Estellon, B., Gardi, F., Megel, R., & Nouioua, K. (2011). Localsolver 1. x: a black-box local-search solver for 0-1 programming. *4OR*, 9(3), 299-316.

Brandao, J. C. S., & Mercer, A. (1998). The multi-trip vehicle routing problem. *Journal of the Operational research society*, 799-805.

Brandão, J., & Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European journal of operational research*, 100(1), 180-191.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation science*, 39(1), 104-118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, Part II:

Metaheuristics. *Transportation science*, 39(1), 119-139.
Buhrkal, K., Larsen, A., & Ropke, S. (2012). The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia-Social and Behavioral Sciences*, 39, 241-254.
Cattaruzza, D., Absi, N., Feillet, D., & Vidal, T. (2014). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236(3), 833-848.
Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2), 105-119.
Crevier, B., Cordeau, J. F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2), 756-773.
Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1), 80-91.
Fleischmann, B. (1990). The vehicle routing problem with multiple use of the vehicles. Working Paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany.
Guerrero, W. J., Prodhon, C., Velasco, N., & Amaya, C. A. (2013). Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics*, 146(1), 359-370.
Gulczynski, D., Golden, B., & Wasil, E. (2011). The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering*, 61(3), 794-804.
Jordan, W. C. (1987). Truck backhauling on networks with many terminals. *Transportation Research Part B: Methodological*, 21(3), 183-193.
Jordan, W. C., & Burns, L. D. (1984). Truck backhauling on two terminal networks. *Transportation Research Part B: Methodological*, 18(6), 487-503.
Kim, B. I., Kim, S., & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12), 3624-3642.
Kuo, Y., & Wang, C. C. (2012). A variable neighborhood search for the multi-depot vehicle routing problem with loading cost. *Expert Systems with Applications*, 39(8), 6949-6954.
Lin, C. K. Y., & Kwok, R. C. W. (2006). Multi-objective metaheuristics for a location-routing problem with multiple use of vehicles on real data and simulated data. *European Journal of Operational Research*, 175(3), 1833-1849.
Mingozzi, A., Roberti, R., & Toth, P. (2013). An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing*, 25(2), 193-207.
Olivera, A., & Viera, O. (2007). Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1), 28-47.
Ombuki-Berman, B. M., Runka, A., & Hanshar, F. (2007, May). Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms. *In Proceedings of the Third IASTED International Conference on Computational Intelligence* (pp. 91-97). ACTA Press.
Petch, R. J., & Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133(1), 69-92.
Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8), 2403-2435.
Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4), 455-472.
Salhi, S., & Petch, R. J. (2007). A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, 6(4), 591-613.
Sevilla, F. C., & de Blas, C. S. (2003). Vehicle routing problem with time windows and intermediate facilities. *SEIO Õ03 Edicions de la Universitat de Lleida*, 3088-3096.
Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2), 254-265.
Taillard, E. D., Laporte, G., & Gendreau, M. (1996). *Vehicle routeing with multiple use of vehicles*. Journal of the Operational research society, 1065-1070.
Toth, P., & Vigo, D. (Eds.). (2001). The vehicle routing problem. *Siam*.

Tung, D. V., & Pinnoi, A. (2000). Vehicle routing–scheduling for waste collection in Hanoi. *European Journal of Operational Research*, 125(3), 449-468.

Wu, T. H., Low, C., & Bai, J. W. (2002). Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29(10), 1393-1415.

Xu, Y., Wang, L., & Yang, Y. (2012). A New Variable Neighborhood Search Algorithm for the Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows. *Electronic Notes in Discrete Mathematics*, 39, 289-296.