Expert Systems With Applications 56 (2016) 335–348

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

The customer-centric, multi-commodity vehicle routing problem with split delivery

Mohammad Moshref-Javadi*, Seokcheon Lee[1]

School of Industrial Engineering, Purdue University, 315 N. Grant St, West Lafayette, IN 47907, USA

ARTICLE INFO

Article history:
Received 25 September 2015
Revised 16 March 2016
Accepted 17 March 2016
Available online 19 March 2016

Keywords:
Minimum latency problem
Traveling repairman problem
Split delivery
Routing
Metaheuristic

ABSTRACT

In this paper, we present the Customer-centric, Multi-commodity Vehicle Routing Problem with Split Delivery (CMVRPSD) whose objective is to minimize total waiting time of customers in distributing multiple types of commodities by multiple capacitated vehicles. It is assumed that a customer's demand can be fulfilled by more than one vehicle. Two classes of decisions are involved in this problem: routing vehicles to customers and quantifying commodities to load and unload. The CMVRPSD can be applied to distributing commodities in customer-oriented distribution problems for both peacetime and disaster situations. The problem is formulated in two Mixed-Integer Linear Programming (MILP) models, and a heuristic method is proposed by adapting and synthesizing Simulated Annealing (SA) and Variable Neighborhood Search (VNS) for large-scale problems. Experimental results show that the proposed hybrid algorithm outperforms other applicable algorithms such as SA, VNS, and Nearest Neighborhood heuristic.

1. Introduction

In today's competitive environment, customer satisfaction plays an important role in the success of service and manufacturing companies. One important aspect that increases customer's satisfaction in both service and manufacturing systems is the quickness in the delivery of services or products to customers. Research on Minimum Latency Problems (MLPs) has been conducted to achieve this goal from the transportation and logistics perspective. MLPs, also known as Cumulative Vehicle Routing Problems (Chen, Dong, & Niu, 2012) or Customer-Centric Vehicle Routing Problems (Martínez-Salazar, Angel-Bello, & Alvarez, 2014) are a class of routing problems whose objective is to minimize the waiting time of customers. The Traveling Repairman Problem (TRP) Bjelić, Vidović, and Popović (2013), also known as the Deliveryman Problem (Mladenović, Urošević, & Hanafi, 2013), is the basic problem in this class and is defined as follows: given a vehicle in a depot (origin), a set of demand locations, and travel times among demand locations, the problem is to find a path for the vehicle to visit every demand location exactly once and the goal is to minimize the sum of arrival times (latencies). While server-oriented Vehicle Routing Problems (VRPs) aim to minimize the total travel distance of the vehicles, MLPs are customer-oriented by minimizing the latency from the customers' viewpoint. To avoid confusion, we use the "TRP" as the basic problem in the class of Minimum Latency Problems, while "MLPs" refers to general problems in this class. MLPs have many real-world applications, such as humanitarian logistics for delivering food, water, and medical supplies to affected areas, dispatching ambulances to patients' locations (Campbell, Vandenbussche, & Hermann, 2008), repair and fix to minimize total downtime (Ribeiro, Laporte, & Mauri, 2012), and disk head scheduling to minimize total time of data retrieval on a disk (Blum et al., 1994).

Customer satisfaction and competitive business environment on one hand, and the increase in the number, scale, and severity of disasters on the other hand, are the key reasons why MLPs have recently received considerable attention from researchers. Different variants of the problem have been developed from 1986 to 2015: TRP on a straight line (Afrati, Cosmadakis, Papadimitriou, Papageorgiou, & Papakostantinou, 1986), TRP with non-zero service time (Tsitsiklis, 1992), the Dynamic-TRP (Bertsimas & Van Ryzin, 1991; Lee, 2011), capacitated MLP (Angel-Bello, Alvarez, & García, 2013; Ke & Feng, 2013; Lysgaard & Wøhlk, 2014; Mattos Ribeiro & Laporte, 2012; Ngueveu, Prins, & Wolfler Calvo, 2010), capacitated MLP with time window (Bjelić et al., 2013; Tsitsiklis 1992; Heilporn, Cordeau, & Laporte, 2010), and directed MLP (Nagarajan & Ravi, 2008). Some authors also presented different latency-based objective functions: minimize maximum latency (Campbell et al., 2008; Psaraftis, Solomon, Magnanti, & Kim, 1990) and maximize latency-based

* Corresponding author: Tel.: +1 (213) 448-3241.
E-mail addresses: moshref@purdue.edu (M. Moshref-Javadi), Lee46@purdue.edu (S. Lee).
[1] Tel. +1 (765) 494-5419.

http://dx.doi.org/10.1016/j.eswa.2016.03.030
0957-4174/

profit (Coene & Spieksma, 2008; Dewilde, Cattrysse, Coene, Spieksma, & Vansteenwegen, 2013).

Various formulations and algorithms have been proposed for MLPs. Bianco, Mingozzi, and Ricciardelli (1993) formulated the TRP based on the network flow problem in the context of time-dependent traveling salesman problem, and presented two exact algorithms. Heilporn et al. (2010) developed two formulations of the TRP with time window. The first formulation uses the arc flow problem and the second formulation utilizes the sequential assignment model. Authors proposed an exact algorithm using polyhedral analysis. Ngueveu et al. (2010) presented a mixed integer formulation of the capacitated MLP, and designed a Memetic algorithm to deal with the problem. Angel-Bello et al. (2013) proposed two formulations of the directed MLP by using permutation-based decision variables and utilized the mathematical models to solve small-size problems. Bjelić et al. (2013) formulated the heterogeneous capacitated MLP with time window using the arc flow network model, and developed a metaheuristic approach based on the Variable Neighborhood Search to solve the problem. Lysgaard and Wøhlk (2014) also formulated the capacitated MLP using the set partitioning formulation and proposed an exact algorithm.

Several heuristic and metaheuristic algorithms have been recently used to tackle MLPs. The first metaheuristic was proposed for the TRP by Salehipour, Sörensen, Goos, and Bräysy (2011) based on GRASP+VNS/VND. Silva, Subramanian, Vidal, and Ochi (2012) proposed a metaheuristic approach for the TRP, named GILS-RVND which uses Greedy Randomized Adaptive Search Procedure (GRASP) to generate solutions, and Variable Neighborhood Descent (VNDS) and Iterated Local Search (ILS) to improve the solutions. The algorithm is able to solve large-scale TRPs up to 1000 nodes efficiently and up to 50 nodes optimally in less than a second. Ke and Feng (2013) presented a two-phase metaheuristic which utilizes an exchange-based or cross-based operator in the first phase. Then, 3-opt and 4-opt based operators are used followed by a 2-opt search. Other metaheuristic algorithms used to solve MLPs include Genetic Algorithm Ban, Nguyen, Cuong Ngo, and Nguyen (2013), Camci (2014), Particle Swarm Optimization Camci (2014), Memetic algorithm (Ngueveu et al., 2010), VNS/VND (Mladenović et al., 2013; Mattos Ribeiro, & Laporte, 2012), and Tabu Search (Dewilde et al., 2013). The reader is referred to Moshref-Javadi and Lee (2013) for a taxonomy and review of MLPs. The taxonomy includes MLP characteristics, objective functions, and solution approaches.

The review of the literature on MLPs indicates that two cases have been mostly researched as follows:

- Traveling Repairman Problem: assumes an incapacitated vehicle distributing a single type of commodity to customers.
- Capacitated Multi-vehicle MLP: assumes multiple capacitated vehicles distributing a single type of commodity to customers.

The primary research in the MLP literature has been on the single commodity. However, the distribution of multiple types of commodities is common and they have to be considered together because of their interdependency rooted from the differences in size, type, and importance. In disaster relief, for example, water could have higher priority for distribution compared to cloths. Considering the heterogeneity of commodities, load and unload decisions need to be made in conjunction with and in addition to the routing decisions. Also, in real world, a customer's demand can be fulfilled by more than one vehicle and this assumption is called 'split' delivery. Fulfilling a demand by only one vehicle is not always feasible due to limited vehicle capacity and heterogeneity of vehicles. Researchers have shown that split delivery can contribute to cost savings in routing problems (Ambrosino & Sciomachen, 2007; Ho & Haugland, 2004; Sierksma & Tijssen, 1998; Song, Lee, & Kim, 2002). While the VRP with split delivery has been researched in the literature, the MLP with split delivery has gained no attention. In the VRP context, various split delivery problems have been formulated and authors have proposed several exact algorithms (Archetti, Bianchessi, & Speranza, 2014; Archetti, Bianchessi, & Speranza, 2015; Archetti, Bouchard, & Desaulniers, 2011; Silva, Subramanian, & Ochi, 2015; Stålhane, Andersson, Christiansen, Cordeau, & Desaulniers, 2012) and heuristics (Archetti, Speranza, & Hertz, 2006; Berbotto, García, & Nogales, 2014; Nowak, Ergun, & White Iii, 2008; Wang, Du, & Ma, 2014). Chen, Golden, and Wasil (2007) and (Archetti and Speranza (2012) reviewed the applications and algorithms for the split delivery VRPs.

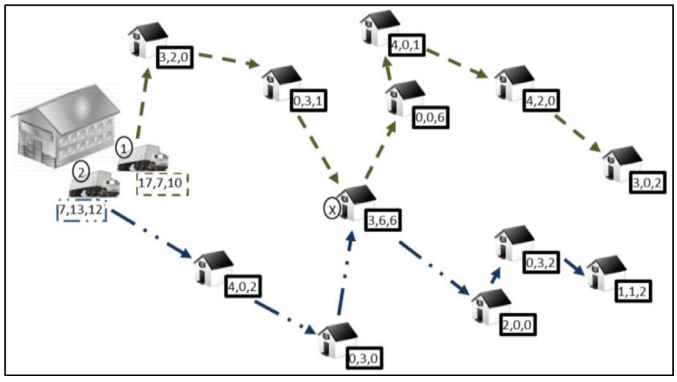

Fig. 1. The CMVRPSD schematic description.

MLP research on multi-commodity and split delivery is missing in the literature despite its practical importance. This paper introduces the *Customer-Centric, Multi-commodity Vehicle Routing Problem with Split Delivery (CMVRPSD)* and proposes two different mathematical formulations. Several optimality properties are investigated and utilized to design a heuristic algorithm for large-scale problems that synthesizes Simulated Annealing (SA) and Variable Neighborhood Search (VNS).

The remainder of the paper is organized as follows: Section 2 presents problem description. Two mathematical models based on network flow and assignment models are presented in Section 3. Section 4 describes the proposed hybrid algorithm and computational results are presented in Section 5. Finally, Section 6 concludes the paper.

2. Problem description

The CMVRPSD includes two classes of decisions: the first decision determines the vehicles' routes while the second decision determines the quantities of commodities to load and unload on the routes of vehicles. The goal is to minimize the total waiting time of customers. We assume that multiple vehicles distribute multiple types of commodities from a single depot. Customers can request any type of commodities and this demand can be fulfilled by more than one vehicle. Fig. 1 illustrates the problem schematically in which there is one depot with two vehicles distributing three types of commodities. The numbers in the boxes of customers indicate the requested commodities of each type while the boxes of vehicles show the quantities of commodities loaded on each vehicle. For example, the delivery to customer X is split between both vehicles; commodity 1 is delivered by vehicle 1 and commodities 2 and 3 are delivered by vehicle 2. More assumptions of the problem are as follows:

- Every vehicle is able to transport all types of commodities.
- Commodities differ with respect to their size and importance.
- Vehicles are heterogeneous in terms of carrying capacity.

- Customers are heterogeneous in demand in terms of type and quantity.
- Vehicles are allowed to load commodities at depot only in the beginning and refilling is not allowed.

3. Mathematical formulation

In this section, we present two mathematical models of the problem. To formulate the CMVRPSD, suppose $G(V,E)$ is an undirected graph where V is the set of $N+1$ vertices $(v_0,\ldots, v_N)$ comprised of customer nodes $(v_1,\ldots, v_N)$ and a depot node (v_0), and E is the set of arcs $\{(v_i, v_j)\colon v_i, v_j \in V, v_i \neq v_j\}$. Associated with each arc, c_{ij} represents the traveling time between nodes v_i and v_j, and the graph is assumed to be symmetric $(c_{ij}=c_{ji})$.

3.1. Mathematical model 1

This model is based on the formulations of the network flow problems. In addition to the defined graph, we add a dummy node v_{N+1} representing the depot together with v_0 in order to facilitate the formulation. The following notations are defined to formulate the problem

Indices	
i,j	Represent customers and depot
k	Represents vehicle
c	Represents commodity
Sets	
K	Set of vehicles
C	Set of commodities
V	Set of customers and depot
V'	Set of customers
Parameters	
q_c	Unit volume of commodity c
Q_k	Capacity of vehicle k
c_{ij}	Travel time between customers i and j
M	Large positive constant
D_{ic}	Demand quantity of customer i for commodity c
Variables	
t_i^k	Arrival time of vehicle k at customer i
x_{ij}^k	Binary variable, 1 if vehicle k traverses arc (i, j) from customer i to customer j, 0 otherwise
z_{ic}^k	The quantity of commodity c transported to customer i by vehicle k.

The Model 1 extends the formulation of the capacitated multi-vehicle MLP developed by Ngueveu et al. (2010) to the multi-commodity case with split delivery as follows:

$$\text{Minimize : Latency} \tag{1}$$

$$\sum_{i\in V} x_{ij}^k = \sum_{i\in V} x_{ji}^k \quad \forall j \in V', k \in K \tag{2}$$

$$\sum_{c\in C}\sum_{i\in V'} z_{ic}^k q_c \le Q_k \quad \forall k \in K \tag{3}$$

$$\sum_{j\in V'} x_{0j}^k = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{i\in V'} x_{i,N+1}^k = 1 \quad \forall k \in K \tag{5}$$

$$M \sum_{\substack{j>0\\ j\neq i}} x_{ij}^k \ge z_{ic}^k \quad \forall k \in K, i \in V', c \in C \tag{6}$$

$$\sum_{k\in K} z_{ic}^k \ge D_{ic} \quad \forall i \in V', c \in C \tag{7}$$

$$t_i^k + c_{ij} - (1 - x_{ij}^k)M \le t_j^k \quad \forall i \in V\backslash\{N+1\}, j \in V\backslash\{0\}, i \neq j, k \in K \tag{8}$$

$$\begin{aligned} &t_i^k \ge 0 \quad \forall i \in V, k \in K,\\ &z_{ic}^k \ge 0 \text{ and integer} \quad \forall i \in V, k \in K, c = 1, \ldots, C\\ &x_{ij}^k \in \{0,1\} \quad \forall i, j \in V, k \in K \end{aligned} \tag{9}$$

In this model, the objective function is to minimize the total latency. Note that, due to split delivery, formulating the objective function is not straightforward and it has to be modified. Therefore, we propose two objective functions in Section 3.3 for the CMVRPSD. Constraints (2) represent the flow continuity in tours. Constraints (3) guarantee that the total quantity delivered by every vehicle does not exceed the vehicle's capacity. Constraints (4, 5) indicate that all tours must start and end at the depot. Constraints (6) relate the variables of the problem, and constraints (7) ensure that all of the demands are satisfied. Latency at each customer is calculated using constraints (8). If vehicle k arrives at customer i at time t_i^k and traverses arc i to j, then the arrival time of the vehicle at customer j is equal to the sum of t_i^k and travel time on arc (i, j). For any customers that a vehicle does not visit, the associated t_j^k is zero according to the minimization objective. To determine the value of M, we sort all of the arcs connected to each customer node and select the longest one for every node (totally N arcs). We also select $|K|-1$ longest arcs connected to the depot. The value of M must be at least the sum of the lengths of the $N+|K|-1$ selected arcs. Constraints (8) eliminate subtours, and finally, constraints (9) specify the ranges and types of the variables.

If the service (unloading) time at each customer is also considered, constraint (8) changes to

$$\begin{aligned} &t_i^k + c_{ij} + r_i^k - (1 - x_{ij}^k)M \le t_j^k,\\ &\forall i \in V\backslash\{N+1\}, \forall j \in V\backslash\{0\}, \forall i \neq j, \forall k \in K \end{aligned} \tag{10}$$

where r_i^k is the service time of vehicle k at customer i.

3.2. Mathematical model 2

This model uses the concept of Assignment Problem by assigning customers to the sequential positions in the route of each vehicle. We define $N+2$ positions in each tour that can be assigned to depot node, N customer nodes, and a dummy node. Let h be the index of the position, H the set of all positions, {0, 1, 2, …, N, $N+1$}, and H' the set of positions except positions 0 and $N+1$, {1, 2, …, N}. The binary variable x_{hk}^i is 1 if vehicle k visits customer i at position h in its tour (i.e., i is h^{th} visit of k); otherwise, it is 0. The problem is formulated as follows:

$$\text{Minimize : Latency} \tag{11}$$

$$x_{0k}^0 = 1 \quad \forall k \in K \tag{12}$$

$$x_{N+1,k}^{N+1} = 1 \quad \forall k \in K \tag{13}$$

$$\sum_{i\in V} x_{hk}^i \le 1 \quad \forall k \in K, h \in H \tag{14}$$

$$\sum_{i\in V} x_{hk}^i \ge \sum_{i\in V} x_{h+1,k}^i \quad \forall k \in K, h \in H\backslash\{N, N+1\} \tag{15}$$

$$\sum_{c\in C}\sum_{i\in V'} z_{ic}^k q_c \le Q_k \quad \forall k \in K \tag{16}$$

$$M \sum_{h\in H'} x_{hk}^i \ge z_{ic}^k \quad \forall k \in K, i \in V', c \in C \tag{17}$$

$$\sum_{k\in K} z_{ic}^k \ge D_{ic} \quad \forall i \in V', c \in C \tag{18}$$

$$t_i^k + c_{ij} - (1 - x_{hk}^i x_{h+1,k}^j)M \le t_j^k$$
$$\forall i \in V, j \in V, i \ne j, k \in K, h \in H \backslash \{N+1\} \quad (19)$$

$$t_i^k \ge 0 \quad \forall i \in V, k \in K$$
$$z_{ic}^k \ge 0 \text{ and integer} \quad \forall i \in V, k \in K, c = 1, ..., C \quad (20)$$
$$x_{hk}^k \in \{0, 1\} \quad \forall i, j \in V, k \in K$$

The objective function minimizes the total latency and will be revisited in detail in Section 3.3. Constraints (12) and (13) ensure that each tour starts and ends at the depot. Constraints (14) guarantee that at most one customer is assigned to any position in a route. Constraints (15) ensure that, in every route, position 0 is filled by depot, position $N+1$ is filled by dummy node, and customers can occupy positions 1, 2, ..., N, starting from 1 and without any void (unoccupied position) in between. So, positions that could remain unassigned are those between the last assigned customer in the route and position $N+1$. Constraints (16) represent the capacity constraint of vehicles. Constraints (17) relate the variables, and constraints (18) ensure that all demands are satisfied. Constraints (19) calculate waiting time of customers. For two consecutive visits to customers i and j in route k, $x_{hk}^i x_{h+1,k}^j = 1$. Thus, the latency at customer j is equal to the sum of the latency at customer i (t_i^k) and travel time on arc (i, j). For any nonconsecutive visits $t_j^k = 0$ and the latency will be zero as well. Finally, constraints (20) represent the types and ranges of variables.

The non-linear term $x_{hk}^i x_{h+1,k}^j$ in constraints (19) can be linearized using a proposition driven by Bulgak and Bektas (2009). We define binary variables U_{hk}^{ij} which combines two variables $x_{hk}^i$and $x_{h+1,k}^j$, and it is 1 only if both variables $x_{hk}^i$and $x_{h+1,k}^j$are 1; otherwise, it is 0. The following constraints are added to the model:

$$x_{hk}^i + x_{h+1,k}^j - U_{hk}^{ij} \le 1 \qquad \forall i \in V, j \in V, i \ne j, k \in K, h \in H \quad (21)$$

$$x_{hk}^i + x_{h+1,k}^j - 2U_{hk}^{ij} \ge 0 \qquad \forall i \in V, j \in V, i \ne j, k \in K, h \in H \quad (22)$$

3.3. Objective functions

Our problem allows split delivery and it leads to multiple deliveries to a customer, i.e., multiple latencies associated with a customer. There could be various ways of defining the objective function in such a situation, and this section presents two rational objective functions and discuss their characteristics.

3.3.1. Maximum arrival time latency

The objective function assumes that the latency at a customer is determined by the last delivery time, i.e., the time taken to fulfill all demands of the customer. Therefore, the latency at customer i is mathematically defined as follows:

$$\text{Latency at customer } i: \ \underset{k \in K}{\text{Max}} \left\{ t_i^k \right\} \quad (23)$$

Thus, the objective function is defined as

$$\text{Min} \sum_{i \in V'} \underset{k}{\text{Max}} \left\{ t_i^k \right\} \quad (24)$$

This objective function can be linearized as in (25) by defining variable u_i and adding constraints (26) as follows:

$$\text{Min} \sum_{i \in V'} u_i \quad (25)$$

$$u_i \ge t_i^k \qquad i \in V', k \in K \quad (26)$$

An example situation where this objective function is appropriate is when companies (can) start production activities with all required materials available. We present three properties of the CMVRPSD with this objective function, when vehicles have equal capacity and distances satisfy the triangle inequality. Properties 2 and 3 have been proved for the VRPs with Split Delivery (Dror, Laporte, & Trudeau, 1994) and we provide their proofs for the MLP context. These properties will contribute to designing an efficient algorithm in Section 4.

Property 1. *If a problem has an optimal solution with s split deliveries, any solution with s′ (s′>s) splits does not have a better objective value.*

Proof. Consider an optimal solution with s splits and assume that customer $i1$'s demand is fulfilled by vehicle $k1$ at time t_{i1}^{k1}. In a new solution, suppose $i1$'s demand is split between vehicles $k1$ and $k2$ at times $\bar{t}_{i1}^{k1}$ and $\bar{t}_{i1}^{k2}$. If $\bar{t}_{i1}^{k1} < \bar{t}_{i1}^{k2}$, the objective value is worse compared to the initial solution. If $\bar{t}_{i1}^{k1} > \bar{t}_{i1}^{k2}$, in the best case the objective function value does not change. □

Property 2. *The number of split deliveries in an optimal solution can be reduced to at most one split delivery between each pair of routes.*

Proof. Suppose two customers $i1$ and $i2$ whose demands are split between two vehicles $k1$ and $k2$. Assumeϕ_{i1}^{k1} and $\phi_{i1}^{k2}$are the quantities of products delivered to customer $i1$ by vehicles $k1$ and $k2$, and ϕ_{i2}^{k1} and $\phi_{i2}^{k2}$are those delivered to customer $i2$ by vehicles $k1$ and $k2$. If the capacity of the vehicles is Q, it is obvious that $\phi_{i1}^{k1} + \phi_{i2}^{k1} \le Q$ and $\phi_{i1}^{k2} + \phi_{i2}^{k2} \le Q$. Without loss of generality, assume a new solution in which $\bar{\phi}_{i1}^{k1} = \min\{\bar{\phi}_{i1}^{k1}, \bar{\phi}_{i1}^{k2}, \bar{\phi}_{i2}^{k1}, \bar{\phi}_{i2}^{k2}\} = 0$. Thus, in the new solution we have, $\bar{\bar{\phi}}_{i1}^{k1} = 0$, $\bar{\phi}_{i1}^{k2} = \phi_{i1}^{k1} + \phi_{i1}^{k2}$, $\bar{\phi}_{i2}^{k2} = Q - \bar{\phi}_{i1}^{k2} = Q - \phi_{i1}^{k1} - \phi_{i1}^{k2}$, and $\bar{\phi}_{i2}^{k1} = \phi_{i2}^{k1} + \phi_{i2}^{k2} - \bar{\phi}_{i2}^{k2} = \phi_{i2}^{k1} + \phi_{i2}^{k2} - Q + \phi_{i1}^{k1} + \phi_{i1}^{k2}$. We need to check the demand fulfilment constraints

$$\bar{\bar{\phi}}_{i1}^{k1} + \bar{\bar{\phi}}_{i1}^{k2} = 0 + \phi_{i1}^{k1} + \phi_{i1}^{k2} \quad (27)$$

$$\bar{\phi}_{i2}^{k1} + \bar{\phi}_{i2}^{k2} = Q - \phi_{i1}^{k1} - \phi_{i1}^{k2} + \phi_{i2}^{k1} + \phi_{i2}^{k2} - Q + \phi_{i1}^{k1} + \phi_{i1}^{k2} = \phi_{i2}^{k1} + \phi_{i2}^{k2} \quad (28)$$

Also, the capacity constraint for $k1$ is:

$$\bar{\bar{\phi}}_{i1}^{k1} + \bar{\bar{\phi}}_{i2}^{k1} = 0 + \phi_{i2}^{k1} + \phi_{i2}^{k2} - Q + \phi_{i1}^{k1} + \phi_{i1}^{k2} \quad (29)$$

As $\phi_{i2}^{k1} + \phi_{i1}^{k1} \le Q$ and $\phi_{i2}^{k2} + \phi_{i1}^{k2} \le Q$, thus, $\bar{\bar{\phi}}_{i1}^{k1} + \bar{\bar{\phi}}_{i2}^{k1} \le Q$. Similarly, the capacity constraint for $k2$ is:

$$\bar{\phi}_{i2}^{k1} + \bar{\phi}_{i2}^{k2} = \phi_{i1}^{k1} + \phi_{i1}^{k2} + Q - \phi_{i1}^{k1} - \phi_{i1}^{k2} = Q \quad (30)$$

Therefore, all of the constraints are satisfied. In the new solution, $\bar{\bar{\phi}}_{i1}^{k1} = 0$ indicating that vehicle $k1$ does not visit customer $i1$. Assume that the arrival time of vehicle $k1$ at customer $i2$ in the new solution is $\bar{t}_{i2}^{k1}$, two cases are possible: if $t_{i2}^{k1} > t_{i2}^{k2}$ in the initial solution, since $k1$ does not visit $i1$ in the new solution, it may arrive earlier ($t_{i2}^{k1} > \bar{t}_{i2}^{k1}$) due to triangle inequality or in the worst case it will arrive at the same time ($t_{i2}^{k1} = \bar{t}_{i2}^{k1}$). So, in this case, the objective value may improve or remain unchanged. In case two if $t_{i2}^{k1} < t_{i2}^{k2}$, the objective value does not change anyway. Therefore, in both cases, an unnecessary split delivery is eliminated. □

Property 3. *At most one customer has split delivery on each route.*

Proof. Suppose that the demands of two customers $i1$ and $i2$ are split in delivery. Portions of $i1$'s and $i2$'s demands are fulfilled by vehicle $k1$ with quantities ϕ_{i1}^{k1} and ϕ_{i2}^{k1}, respectively. The rest of the demand at customer $i1$ is satisfied by vehicle $k2$ with quantityϕ_{i1}^{k2}. Similarly, the rest of the demand at customer $i2$ is fulfilled by vehicle $k3$ with quantityϕ_{i2}^{k3}. Based on the assumption, we have $\phi_{i1}^{k1} + \phi_{i1}^{k2} \leq Q$ and$\phi_{i2}^{k1} + \phi_{i2}^{k3} \leq Q$, and considering the vehicle capacity constraint we have $\phi_{i1}^{k1} + \phi_{i2}^{k1} \leq Q$. Now, if in a new solution we swap ϕ_{i1}^{k2} and ϕ_{i2}^{k1} for $i1$ and $i2$, that is, we let vehicle $k1$ deliver the rest of the customer $i1$'s demand that was delivered by vehicle $k2$ before. Similarly, the portion of customer $i2$'s demand that was delivered by vehicle $k1$ is now fulfilled by vehicle $k2$. Thus, in the new solution, $\bar{\phi}_{i1}^{k1} \leftarrow \phi_{i1}^{k1} + \phi_{i1}^{k2}$ and $\bar{\phi}_{i2}^{k2} \leftarrow \phi_{i2}^{k2} + \phi_{i2}^{k1}$. The new solution is valid because $\phi_{i1}^{k2} \leq Q - \phi_{i1}^{k1}$and $\phi_{i2}^{k1} \leq Q - \phi_{i1}^{k1}$. Thus, in the new solution, customer $i1$'s demand is fulfilled in one visit by vehicle $k1$ and customer $i2$'s demand is satisfied by vehicles $k2$ and $k3$, which leads to non-worse latency due to the triangular inequality assumption. □

3.3.2. Weighted arrival time latency

The second objective function is based on the weighted latency multiplying the latency by corresponding delivery quantity and importance weight $w_c$of the commodity. This objective function is formulated as

$$\sum_{k\in K}\sum_{i\in V'}\sum_{c\in C} w_c z_{ic}^k t_i^k \tag{31}$$

This objective function can be used for distribution systems in business or disaster situations in which the goods are heterogeneous in urgency and consumed as soon as delivered. In such situations, more critical products have to be delivered earlier. A proposition is introduced as follows, which contributes to significantly reducing solution space and hence enhancing algorithm efficiency.

Proposition 1. *The demand of a customer has to be fulfilled in decreasing order of w_c/q_c.*

Proof. Consider two commodities *c1* and *c2* with weights *w1* and *w2* and unit volumes *q1* and *q2*, and assume $w_1/q_1 > w_2/q_2$. Suppose, without loss of generality, vehicle *k1* visits customer *i1* at time t_{i1}^{k1} and vehicle *k2* visits *i1* at t_{i1}^{k2}, and $t_{i1}^{k1} < t_{i1}^{k2}$. We consider two cases: in the first case commodity *c1* is first selected for delivery while in the second case commodity *c2* is selected first. The latency at customer *i1* in the two cases are given in (32) and (33) respectively

$$t_{i1}^{k1}.w_1.\phi_{i1,c1}^{k1} + t_{i1}^{k2}.(w_1.\phi_{i1,c1}^{k2} + w_2.\phi_{i1,c2}^{k2}) \tag{32}$$

$$t_{i1}^{k1}.w_2.\bar{\phi}_{i1,c2}^{k1} + t_{i1}^{k2}.(w_2.\bar{\phi}_{i1,c2}^{k2} + w_1.\bar{\phi}_{i1,c1}^{k2}) \tag{33}$$

If we assume that the demand of the customer $i1$ is $D_{i1,c1}$ and $D_{i1,c2}$ for $c1$ and $c2$, then the latency becomes

$$t_{i1}^{k1}.w_1.\phi_{i1,c1}^{k1} + t_{i1}^{k2}.(w_1.(D_{i1,c1} - \phi_{i1,c1}^{k1}) + w_2.D_{i1,c2}) \tag{34}$$

$$t_{i1}^{k1}.w_2.\bar{\phi}_{i1,c2}^{k1} + t_{i1}^{k2}.(w_2.(D_{i1,c2} - \bar{\phi}_{i1,c2}^{k1}) + w_1.D_{i1,c1}) \tag{35}$$

By simplifying these values, we obtain

$$w_1.\phi_{i1,c1}^{k1}(t_{i1}^{k1} - t_{i1}^{k2}) \tag{36}$$

$$w_2.\bar{\phi}_{i1,c2}^{k1}(t_{i1}^{k1} - t_{i1}^{k2}) \tag{37}$$

And (36) is less than (37) due to $w_1/q_1 > w_2/q_2$. This means $c1$ has to be delivered first. □

Using this proposition, commodities are assigned a priority at the point of delivery. When a vehicle visits a customer, commodities with the highest priority are delivered first. Afterwards, commodities with the second highest priority are delivered. In other words, the order of serving commodities is defined by this ratio.

4. The heuristic approach

The TRP was proved to be NP-Hard (Sahni & Gonzalez, 1976), therefore, the CMVRPSD is NP-hard as well. Though small-size problems can be solved using the proposed mathematical models, solving large-scale problems essentially requires an efficient heuristic algorithm because of the NP-hardness. This section presents a heuristic algorithm which incorporates two well-known heuristic methods, the Variable Neighborhood Search (VNS) and the Simulated Annealing (SA). The combination of SA and VNS has been shown to be successful on different problems, such as routing problems (Xiao, Zhao, Kaku, & Mladenovic, 2014; Xu & Jiang, 2014), job scheduling problems (Rabiee, Rad, Mazinani, & Shafaei, 2014), and facility location problems (Hosseini, Al Khaled, & Vadlamani, 2014). The idea behind each algorithm is briefly described in the following.

VNS is a heuristic method that systematically searches different neighborhoods in the solution space (Mladenović & Hansen, 1997). Starting from an initial solution, it searches for the best solution in the neighborhood of the initial solution. If no better solution can be found after local search, the neighborhood is changed using the *Shaking* operator and the local search is repeated. This procedure of changing neighborhood and applying local search is iterated continually until certain termination conditions are met.

SA is a heuristic procedure developed by Kirkpatrick (1984). The main idea of SA is escaping from local optimal solutions by probabilistic movements in the solution space. Starting from an initial solution, this algorithm generates a new solution and the objective values of the new and current solutions are compared. If the new one improves, it is considered as a new current solution. On the other hand, if the new solution does not improve the objective value, rather than just discarding it, the algorithm accepts it as a new current solution with a small probability $\exp(-\Delta/T)$, where Δ is the difference in objective value between the current and new solutions and T is a parameter denoting the temperature of the process.

The hybrid algorithm takes advantage of desirable properties of both algorithms. We adopt temperature from the SA for global exploration capability and the search strategy of VNS for local exploitation capability. The problem-specific optimality conditions presented are also incorporated into the hybrid algorithm to add more efficiency in the search process.

4.1. Initial solution

To generate an initial solution, we use a greedy heuristic (Fig. 2) in which a vehicle with the largest remaining capacity is selected and the closest customer to this vehicle is chosen. Commodities are delivered to the selected customer by applying the Proposition 1. Split delivery may occur during this process. The process continues until all demands are fulfilled. Selecting a vehicle with the largest remaining capacity leads to less number of split deliveries and consequently less total latency according to the Property 1.

4.2. Shaking

Shaking is a process to diversify the search neighborhood by systematically perturbing the best current solution. Several

Algorithm: Initial Solution Generation

1: Until all demands are satisfied
2: Select a vehicle (k) with the greatest residual capacity
3: Find the closest customer (i) to the current location of vehicle k
4: **If** the demand can be satisfied completely
5: Assign customer i to the vehicle's route
6: Assign all demands of customer i to the selected vehicle
7: Compute arrival time of vehicle k at customer i
8: **Else**
9: Assign as much demands as possible to vehicle k considering w_c/q_c
10: Assign the demands of the customer to vehicle k
11: Compute arrival time of vehicle k at customer i
12: **End If**
13: End Until
14: Compute total waiting time

Fig. 2. Pseudocode of the heuristic for generating initial solutions.

operators were tested for the Shaking step and among those, Relocation, Exchange, 3-Reloc, and CrossMove yielded the best results. The operators are illustrated in Fig. 3. The Relocation operator moves one segment of a route to another route. We used two different approaches to select a node for relocation. The node can be selected either at random (*rand_reloc*) or according to the incremental value to the objective function (*incr_reloc*); the more the incremental value is, the higher the probability is for selecting as a relocation node, in order to minimize the resulting incremental value. The Exchange operator swaps randomly selected segments of two routes. The *3-Reloc* involves three routes and one node in each route to exchange them among the routes. Finally, CrossMove first selects two routes and one point in each route, and then all of the subsequent customers after the selected point in the route are moved to the other route. The numbers of chosen customers from each route could be different. The operators are applied on randomly selected routes. Note that using these shaking operators, a customer may appear twice on a same route. To eliminate such unreasonable routing solutions, the algorithm merges multiple visits to each customer into one. The best sequence of shaking operators by some preliminary experiment is Exchange, incr_reloc, rand_reloc, 3-Reloc, and CrossMove. The shaking process is applied totally G times in each iteration.

Properties 2 and 3 enable to reduce the number of customers with split delivery. To achieve this, we consider merging splits to improve the objective value. A node's demand is split if it appears on more than one route. To reduce the number of splits, a customer with split delivery is selected from a route and moved to another route with the same customer. As a result, the customer's demand is now satisfied by one less vehicle. To perform these moves, Relocation and Exchange operators are utilized. The resulting solution could be infeasible according to the limited capacity of vehicles. We will explain how to deal with these infeasible solutions in Section 4.4. It should be noted that using Property 3, the number of customers with split delivery is less than the number of vehicles and any solution violating this condition cannot be optimal.

In case of the maximum arrival time latency as performance objective, if a perturbation affects the arrival time of a vehicle at a node with split delivery, the latency at the node potentially needs to be updated because the latency at a node is the latest arrival time of vehicles visiting the node. Thus, after updating arrival times, latencies at all affected nodes as well as all nodes with split delivery must be updated. Note that if a node is moved to a route, the latencies at all of the subsequent nodes in the route are affected.

4.3. Local search

Local search operators, 2-Opt, Swap, and Insertion, are applied to a perturbed solution to search for locally optimal solutions (Fig. 3). The 2-Opt reverses the sequence of visits to a set of randomly selected customers in a route. The Swap operator swaps two customers in a selected route, and the Insertion operator moves a node from one location to another in a route. The sequence of these operators is 2-Opt, Swap, and Insertion. To reduce processing time, these operators are only applied to the routes that are perturbed during the Shaking process. The local search process is then followed by the Exchange and CrossMove operators since we found this sequence promising in the local search.

4.4. Acceptance criteria

Due to the capacity constraints, infeasible solutions may be produced during the Shaking process. The pseudocode for acceptance process is shown in Fig. 4. If a solution yields a better objective value, its feasibility is checked. In case it is feasible, the best so far solution and current solution are updated with this new solution. On the other hand, if the new solution is non-improving, it is accepted with probability $\exp(-\Delta/T)$, where Δ is the difference in objective value between the current and new solutions and T is a parameter denoting the temperature of the process. The result of a preliminary experimentation showed that accepting infeasible solutions, whether improving or non-improving, could yield better solutions. Therefore, we also accept infeasible solutions according to the acceptance probability but after they are penalized by adjusting their objective values with the amount of excess over the capacity multiplied by a coefficient. Hence, the algorithm accepts, with a small probability, both non-improving solutions and infeasible improving solutions.

The proposed algorithm uses temperature T for managing acceptance of non-improving or infeasible solutions. By a preliminary experimentation, we noticed that the algorithm performs well in low temperature values. Therefore, the algorithm starts the cooling process with rate α ($T_{new} = \alpha T_{\text{old}}$) at a temperature that is not very high. After reaching the minimum temperature (*Min_T*), we start increasing the temperature at the same rate α ($T_{new} = T_{old}/\alpha$). As a result, the algorithm starts assigning higher probability for accepting low quality or infeasible solutions.

4.5. Termination conditions

If the algorithm cannot find a feasible improving solution in either the increase-temperature or decrease-temperature loop, the algorithm is terminated.

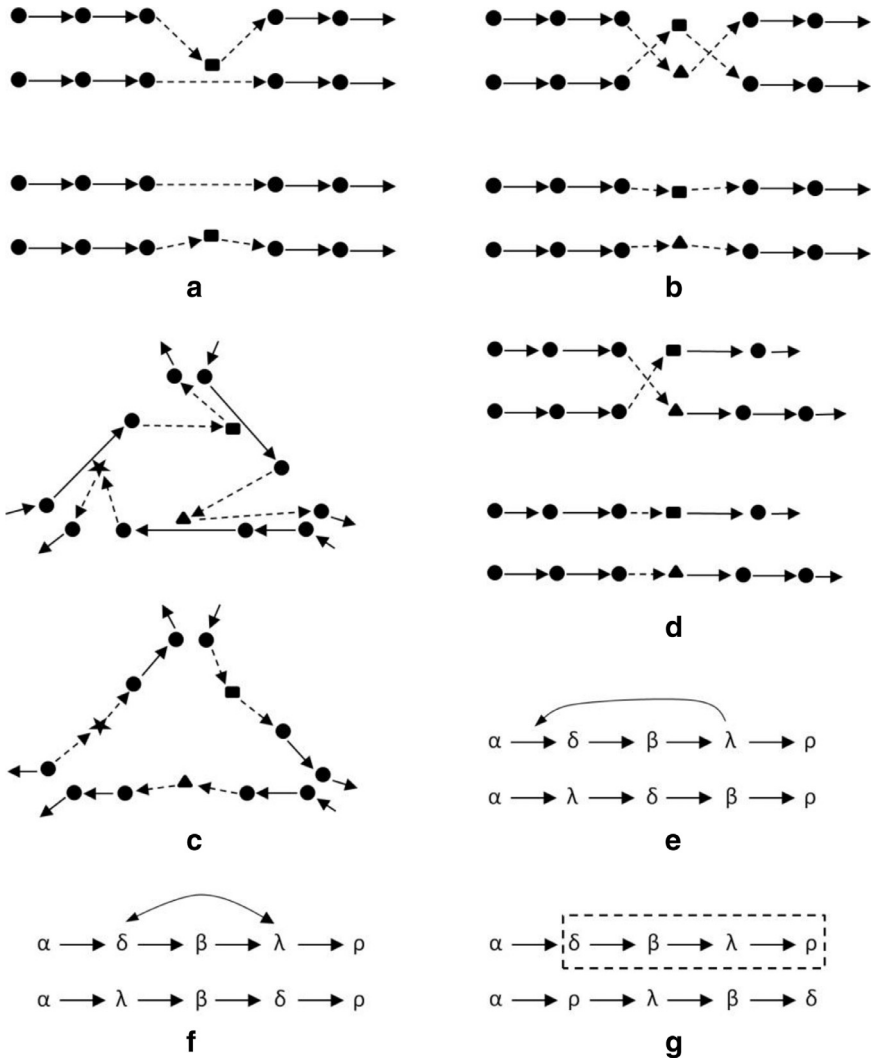

Fig. 3. Perturbation and local search operators: a) Relocation, b) Exchange, c) 3-Reloc, d) CrossMove, e) Insertion, f) Swap, and g) 2-Opt.

Acceptance Criteria

If $f(x_{best}) > f(x_p)$

  If x_p is feasible

    $x_{best} \leftarrow x_p,\ f(x_{best}) \leftarrow f(x_p)$

    $current\_sol \leftarrow x_p,\ f(current\_sol) \leftarrow f(x_p)$

    Terminate = **False**

  Else

    Move to infeasible solution with probability $\exp(f(current\_sol) - f(x_p))/T$,

    $current\_sol \leftarrow x_p,\ f(current\_sol) \leftarrow f(x_p)$

  End

Else

  Move to worse solution with probability $\exp(f(current\_sol) - f(x_p))/T$,

  $current\_sol \leftarrow x_p,\ f(current\_sol) \leftarrow f(x_p)$

End

Fig. 4. Pseudocode of the acceptance criteria for accepting non-improving and/or infeasible moves.

4.6. Overview of the hybrid algorithm

Fig. 5 shows the pseudocode of the proposed algorithm. It starts with defining the neighborhood structures and operators. The parameters of the algorithm are initialized. In step 2, *G* and *L* are the numbers of times that the shaking and local search operators are applied, respectively. *Min_T* and *Max_T* represent the minimum and maximum temperatures. Next, the heuristic shown in Fig. 2 is used to generate an initial solution, which undergoes initial local search for potential improvements. The algorithm has two main

Hybrid Algorithm

1: Define neighborhood structures and operators,
2: Set **G**, **L**, **Min_T**, **Max_T**, and operators' probabilities
3: Generate an initial solution,
4: Apply initial Local Search (x_l)
5: $x_{best} \leftarrow x_l$, $f(x_{best}) \leftarrow f(x_l)$ *//Initialize the Best Solution*
6: *Decrease_Loop* = **True**
7: *Terminate* = **False**
8: **Do** while *Terminate* = **False**
9: **If** *Decrease_Loop* = **True**
10: *//Temperature decrease loop*
11: *Terminate* = **True**
12: **Do** while ***T>Min_T***
13: $current\_sol \leftarrow x_{best}$
14: **For** *i* = 1 to ***G***
15: Select and apply one of the **Shaking** operators (x_p)
16: Acceptance criteria *//Accept or reject the move*
17: **End For**
18: **For** *i* = 1 to ***L***
19: Apply **local search** on the perturbed routes of the $current\_sol \cdot (x_p)$
20: Acceptance criteria *//Accept or reject the move*
21: **End For**
22: $T=\alpha T$
23: **End Do**
24: **Else**
25: *//Temperature increase loop*
26: Terminate = **True**
27: **Do** while ***T<Max_T***
28: $current\_sol \leftarrow x_{best}$
29: **For** *i* = 1 to ***G***
30: Select and apply one of the **Shaking** operators (x_p)
31: Acceptance criteria *//Accept or reject the move*
32: **End For**
33: **For** *i* = 1 to ***L***
34: Apply **local search** on the perturbed routes of the $current\_sol \cdot (x_p)$
35: Acceptance criteria *//Accept or reject the move*
36: **End For**
37: $T=T/\alpha$
38: **End Do**
39: **End If**
40: **End Do**

Fig. 5. Pseudocode of the Hybrid algorithm.

loops. The first loop starts with the maximum temperature and it decreases with rate α until it reaches the minimum temperature. In this loop, different neighborhoods of the solution space are searched using shaking operators. In each neighborhood, local search operators are used to effectively find better local solutions. After finding a solution, the Acceptance Criteria is used to determine whether to accept or not the move with a probability function that involves the temperature. When the temperature reaches its minimum, the second loop starts in which the temperature increases and the shaking and local operators are similarly applied. If the best solution found does not improve in either increase-temperature or decrease-temperature loop, the run is terminated.

5. Computational results

In this section, the mathematical models are evaluated on small size problems. The SA and VNS are used to evaluate the performance of the hybrid algorithm on large size problems. The VNS uses the same strategies for initial solution, perturbation, and local search as the hybrid algorithm. The VNS, however, does not have the notion of temperature; therefore, non-improving or infeasible solutions are not accepted at all. The VNS uses the same shaking and local search operators as the Hybrid algorithm. The SA algorithm uses temperature to potentially move to non-improving or infeasible solutions and utilizes the same perturbation strategy to generate neighborhood solutions. However, no local search is performed in SA and it terminates similarly to the Hybrid algorithm. The parameters of each algorithm are set separately by a series of preliminary experimentations.

A greedy upper bound is computed based on the Nearest Neighbor (*NN*) policy in order to evaluate the algorithms. In each iteration, the algorithm searches for an unassigned customer with the shortest distance to the current locations of the vehicles and selects the corresponding vehicle and customer as the next vehicle's destination. The process terminates when all demands are fulfilled.

5.1. Problem instances

To compare between the two mathematical models, a set of small-size problem instances with 10 customers, G1, were generated. We assumed a 100×100 region and uniform-randomly located all the vertices. Then, the Euclidean distance between each pair of vertices was calculated. Five vehicles and one type of commodity were considered for all the instances in G1.

Table 1
Problems' specifications.

Problem set	Number of customers	Number of vehicles	Number of commodities	Number of instances
G1	10	5	1	7
CMT1	50	5	1	1
CMT2	75	10	1	1
CMT3	100	8	1	1
CMT4	150	12	1	1
CMT5	199	17	1	1
CMT6	120	7	1	1
CMT7	100	10	1	1
mCMT1	50	5	3	1
mCMT2	75	10	3	1
mCMT3	100	8	3	1
mCMT4	150	12	3	1
mCMT5	199	17	3	1
mGWKC1	240	9	3	1
mGWKC2	320	10	3	1
mGWKC5	200	5	3	1
mGWKC6	280	7	3	1
mGWKC9	255	14	3	1
mGWKC10	323	16	3	1
mGWKC17	240	22	3	1
mGWKC18	300	27	3	1
CCM1	50	varies	1	7
CCM2	75	varies	1	7
CCM3	100	varies	1	7
CCM4	150	varies	1	7
CCM5	199	varies	1	7
CCM6	120	varies	1	7
CCM7	100	varies	1	7

In order to evaluate the Hybrid algorithm, we use it to solve three sets of problems from the literature. First, we evaluate our algorithm on problem instances CMT from Christofides, Mingozzi, and Toth (1979) and compare with the best-known solution by Lysgaard and Wøhlk (2014). Lysgaard and Wøhlk (2014) disallowed split delivery, but we can use the problem instances for evaluating our proposed algorithm which is designed for both split and non-split delivery problems. Next, we adopted large-scale problem instances from Christofides et al. (1979) and Golden, Wasil, Kelly, and Chao (1998) with the number of customers varying from 50–323 shown in Table 1 as mCMT and mGWKC. Three commodities have been assumed in these problems and thew_c,q_c, and D_{ic} were generated at random while keeping capacity constraints tight with less than 3% extra capacity for enabling split delivery. Finally, the Hybrid algorithm will be used to solve CCM problems generated by Campos, Corberán, and Mota (2008). This set assumes seven different networks and seven instances generated for each network. In each instance, customers' demands have been generated in different ranges with respect to the capacity of the vehicle to enable split delivery.

5.2. Performance of the mathematical models

The evaluation of the two mathematical models has been performed using CPLEX 12.4 (Branch and Bound) solver on a PC with a 2.66 GHz Intel processor and 2.00 GB of RAM. In this evaluation, the two mathematical models were evaluated on problems G1 with the *maximum arrival time latency* as objective function, and the results are given in Table 2. The maximum allowed runtime is 260,000 seconds and we report the time taken to make the last improvement in the objective value. Though no dominance is present, Model 2 (the linear version) seems better in overall in terms of computation time. The results also indicate that the CMVRPSP is very complex in that even 10-node problems could not be solved optimally in reasonable time. We also use the mathematical model to solve smaller size problems. Based on the experimentation with 5–10 customers, 3 vehicles, and 1 commodity, the largest problem that was solved optimally had 6 customers with computation time 440 seconds.

Table 2
Average CPU time comparison on problems G1 (in seconds).

Problem	Model 1		Model 2	
	Best solution value	CPU Time	Best solution value	CPU Time
1	847.53	158,951	849.03	**2,582**
2	**611.4**	**16,112**	**611.4**	53,192
3	**871.95**	**13,336**	**871.9**	41,464
4	**824.6**	45,392	**824.7**	**10,511**
5	775.08	**5,527**	772.12	41,134
6	**850.68**	75,033	**850.68**	**12,053**
7	**609.81**	10,230	**609.81**	**1,683**
Average		46,369		23,231

Table 3
Taguchi Scheme L_{27} used for parameter setting.

Experiment	Min_T	Max_T	α	G	L
1	5	100	0.995	10	50
2	5	100	0.995	10	60
3	5	100	0.995	10	70
4	5	120	0.999	14	50
5	5	120	0.999	14	60
6	5	120	0.999	14	70
7	5	140	0.9995	18	50
8	5	140	0.9995	18	60
9	5	140	0.9995	18	70
10	10	100	0.999	18	50
11	10	100	0.999	18	60
12	10	100	0.999	18	70
13	10	120	0.9995	10	50
14	10	120	0.9995	10	60
15	10	120	0.9995	10	70
16	10	140	0.995	14	50
17	10	140	0.995	14	60
18	10	140	0.995	14	70
19	20	100	0.9995	14	50
20	20	100	0.9995	14	60
21	20	100	0.9995	14	70
22	20	120	0.995	18	50
23	20	120	0.995	18	60
24	20	120	0.995	18	70
25	20	140	0.999	10	50
26	20	140	0.999	10	60
27	20	140	0.999	10	70

5.3. Heuristics results

The Hybrid, SA, VNS, and NN algorithms are applied to solve the large-scale problems. To set up the proposed Hybrid algorithm for best results, we first conducted a preliminary experimentation to recognize the approximate ranges of parameters that are promising to produce best results. We designed, then, a parameter tuning experimentation based on the L_{27} Taguchi scheme (Table 3) for the five parameters: *Min_temp, Max_temp, Temp. change rate (α), Shaking iteration (G),* and *Local Search (L)*. We selected a mid-size problem and three levels for each parameter: $Min\_T \in \{5, 10, 20\}$, $Min\_T \in \{100, 120, 140\}$,$\alpha \in \{0.995, 0.999, 0.9995\}$, $G \in \{10, 14, 18\}$, and $L \in \{50, 60, 70\}$. The results of this experimentation are shown in Fig. 6, where 1, 2, and 3 indicate the levels of parameters. The parameters are set as in Table 4 through this parameter tuning, and other parameters such as operators' probabilities and penalty coefficient were readjusted by trial and errors after the five parameters are fixed. The parameters of SA and VNS are set similarly. All algorithms are coded in MATLAB.

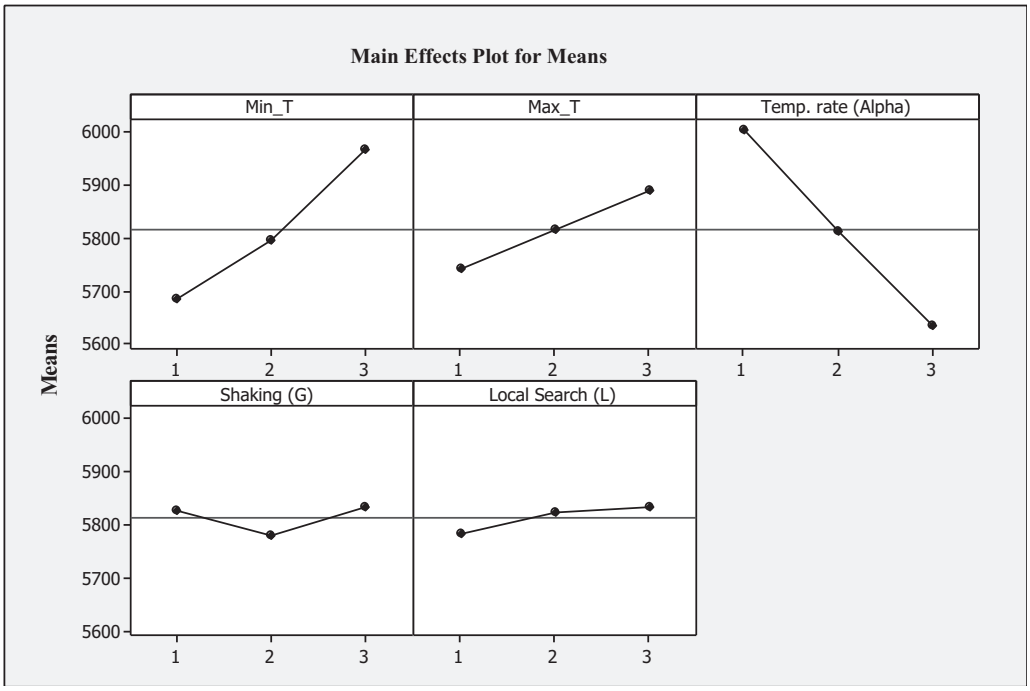

Fig. 6. Experimental results of setting five parameters based on Taguchi Scheme L_{27}.

Table 4
Algorithms parameter setting.

Parameter	Hybrid	VNS	SA
[Minimum, Maximum] temperature	[5, 100]	-	[5, 100]
Temperature change rate (α)	0.9995	-	0.9995
Shaking iteration (G)	14 times	25 times	14 times
[Incr_reloc, Rand_reloc, 3-reloc, CrossMove] probability	[0.2, 0.2, 0.2, 0.4]	[0.2, 0.2, 0.2, 0.4]	[0.2, 0.2, 0.2, 0.4]
Local Search (L)	50 each route	50 each route	-
Penalty coefficient	100 (*max_latency*), 150 (*weighted_latency*)	100 (*max_latency*), 150 (*weighted_latency*)	100 (*max_latency*), 150 (*weighted_latency*)
Total number of iterations	-	2500	-

Table 5
Comparing the results of the Hybrid algorithm with the best known solution by Lysgaard and Wøhlk (2014) on Christofides et al. (1979) problems.

Problem	Number of customers	Number of vehicles	BKS	Hybrid Algorithm	% Gap
CMT1	50	5	2,230.35	2,243.8	0.60
CMT2	75	10	2,391.63	2,440.1	2.03
CMT3	100	8	4,045.42	4,099.3	1.33
CMT4	150	12	4,987.52	5,113.3	2.52
CMT5	199	17	5,809.59	5,988.10	3.07
CMT6	120	7	7,314.55	7,559.2	3.34
CMT7	100	10	3,558.92	3,586.6	0.78
Average			**4334.00**	**4432.91**	**1.95**

The Hybrid algorithm was applied to CMT problems, from the literature on the Cumulative Capacitated Vehicle Routing Problem and the results are compared with their best known solutions (BKS), as shown in Table 5 where the gap is calculated as $(Best_{Hybrid} - Best_{BKS})/\ Best_{BKS}$. In overall the gap is small with average less than 2%. It is also noteworthy that the Hybrid algorithm is designed to handle both problems with and without split delivery and the CMT problems are non-split delivery problems.

The algorithm was then applied to the problems of mCMT and mGWKC for both objective functions, *maximum arrival time latency and weighted arrival time latency*. Table 6 shows the results obtained for *maximum arrival time latency*. The *Best* and *Avg* columns indicate the best and average of ten independent runs, and *UB* indicates the upper bound solution by the *NN* algorithm. The *Relaxed Model 1* provides the best solutions obtained from solving Model 1 in CPLEX with no capacity constraints and allowing 7200 seconds at maximum. The results show that the hybrid method exhibits dominance in producing the best solutions over other algorithms. The *Relaxed Model 1* found solutions in four problems but with large gap to the solutions from other heuristic algorithms, and no feasible solution was found for the rest of the problems in 7200 seconds, supporting the difficulty in solving large-size problems even with relaxed formulation and therefore underpinning the need for efficient heuristic algorithms for this class of problems.

Table 6
Evaluation results on objective function *max_arrival* latency.

Problem set	SA		VNS		Hybrid		UB	Relaxed Model 1
	Best	Avg.	Best	Avg.	Best	Avg.		
mCMT1	2,338.1	2,357.4	2,341.7	2,341.7	**2,311.7**	2,327.9	2,843.1	3,611.5
mCMT2	2,528.1	2,591.5	2,586.5	2,605.9	**2,463.4**	2,541.0	3,460.3	9,122.4
mCMT3	4,733.3	4,802.5	4,697.4	4,777.7	**4,593.3**	4,653.0	4,977.6	21,838
mCMT4	5,622.7	5,681.3	5,578.5	5,644.3	**5,482.6**	5,573.8	6,645.6	20,203.9
mCMT5	6,543.9	6,642.6	6,321.1	6,456.4	**6,218.8**	6,348.3	7,517.6	-
mGWKC1	57,451.3	57,883.2	56,637.4	57,104.4	**56,525.4**	56,904.4	75,687.5	-
mGWKC2	104,630.2	105,613.1	104,280.4	104,549.3	**103,252.70**	103,672.8	142,948.7	-
mGWKC5	128,730.5	129,309.5	126,240.2	127,157.2	**121,180.6**	125,834.8	136,227.4	-
mGWKC6	147,430.9	148,480.2	146,530.8	147,675.7	**144,300.8**	145,683.1	183,718.8	-
mGWKC9	5,193.4	5,269.7	5,228.2	5,265.5	**5,062.4**	5,142.5	6,127.0	-
mGWKC10	7,194.1	7,284.2	7,291.0	7,321.1	**7,115.6**	7,166.7	8,947.0	-
mGWKC17	3,336.2	3,389.0	3,353.4	3,394.5	**3,278.2**	3,355.5	4,055.4	-
mGWKC18	4,472.5	4,524.1	4,475.6	4,524.2	**4,418.2**	4,475.4	5,541.1	-

Table 7
Evaluation results on objective function *weighted_arrival* latency.

Problem set	SA		VNS		Hybrid		UB
	Best	Avg.	Best	Avg.	Best	Avg.	
mCMT1	78,995.0	79,421.2	78,058.0	78,507.1	**75,980.7**	76,745.0	91,808.2
mCMT2	46,960.4	46,976.3	46,829.0	47,095.1	**46,072.3**	46,531.5	62,574.8
mCMT3	109,122.3	110,090.0	108,493.9	109,274.1	**106,209.7**	107,772.7	114,671.6
mCMT4	106,908.5	108,742.3	105,628.8	107,219.8	**103,276.6**	105,275.8	121,254.5
mCMT5	121,851.8	123,537.1	114,563.6	119,156.8	**113,711.6**	114,446.6	132,879.8
mGWKC1	2,091,757.9	2,117,226.5	2,080,483.8	2,094,100.8	**2,056,249.9**	2,077,110.5	2,720,143.7
mGWKC2	2,053,618.4	2,066,766.1	2,026,498.6	2,030,937.5	**2,014,814.1**	2,023,418.9	2,793,927.3
mGWKC5	3,418,095.8	3,466,034.4	3,184,712.3	3,293,736.2	**3,160,926.5**	3,231,849.6	3,917,647.9
mGWKC6	2,899,082.8	2,937,804.8	2,837,827.1	2,862,869.5	**2,814,332.2**	2,835,153.0	3,647,267.2
mGWKC9	135,580.8	136,514.2	133,860.2	134,627.4	**130,108.9**	131,978.0	155,335.1
mGWKC10	219,940.3	220,950.6	220,276.0	221,229.6	**213,676.9**	216,275.7	263,834.6
mGWKC17	104,753.2	107,438.8	104,459.2	106,858.3	**102,717.8**	104,864.2	124,240.2
mGWKC18	155,183.9	155,989.1	154,632.6	155,172.1	**152,057.5**	153,166.9	187,314.8

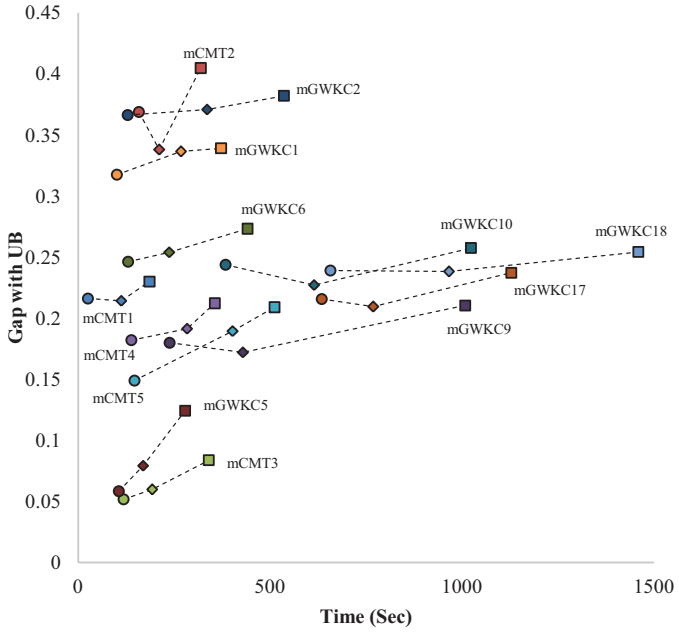

Fig. 7. Time-Gap tradeoff on objective function *max_arrival*: Hybrid (Square), VNS (Diamond), SA (Circle).

Fig. 7 illustrates the tradeoff between the gap and computation time for *maximum arrival time latency*. The vertical axis is for the gap of the algorithms with upper bound, and the horizontal axis is for computation time. The gap is calculated as $(UB - Algorithm_{best})/UB$ in which $Algorithm_{best}$ is the best solution obtained by an algorithm. The square, diamond, and circle indicate Hybrid, VNS, and SA, respectively. For the problems with low computation time (small-size problems), the gap values are low for some problems (mCMT3 and mGWKC5) and high for other problems (CMT2, mGWKC1, mGWKC2). Hence, it can be concluded that there is no relation between the problem size and gap values. However, the computation time of the hybrid algorithm is higher than other algorithms due to multiple search operators. The hybrid algorithm is also able to obtain gap more than 40% with *UB* in problem mCMT2. Similar pattern could be observed for *weighted arrival time latency*. As shown in Table 7 and Fig. 8, the hybrid method outperforms all the other three algorithms with respect to the best solution.

The average computation time is summarized in Table 8 for both objectives. The results illustrate that computation time of the algorithms tends to be larger with problem size. It is also noticeable that, as shown in Fig. 9 and Fig. 10, the computation time of the hybrid algorithm is about 94% of the sum of the computation times of SA and VNS, as the hybrid algorithm is a combination of the two.

To evaluate the effect of the components of the hybrid algorithm, we generated four variants of the algorithm. Case (a) assumes a typical SA termination criterion. In this case, the algorithm starts with an initial temperature and decreases until it becomes less than a preset temperature. Case (b) does not accept any infeasible or non-improving solutions in the acceptance criteria, and case (c) does not use local search in the algorithm. Finally, case (d) considers all these three cases together, i.e., the proposed hybrid algorithm without any of the aforementioned elements. Table 9 shows the percentage gap with the best solution produced by the Hybrid algorithm, $\%Gap = \%(Best_{Modfied\_Algorithm} - Best_{Hybrid})/Best_{Hybrid}$.

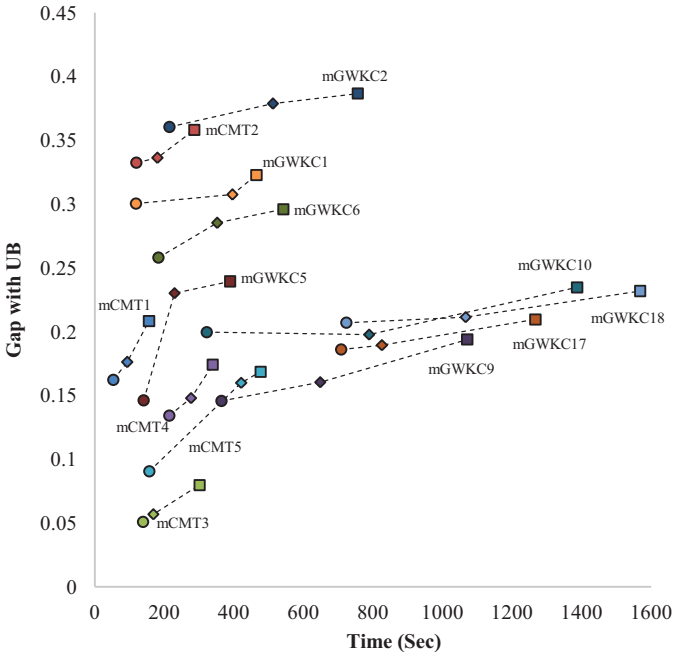

Fig. 8. Time-Gap tradeoff on objective function *weighted_arrival*: Hybrid (Square), VNS (Diamond), SA (Circle).

Table 8
Average computation time for objective functions *max_arrival* and *weighted_arrival* latency (seconds).

Problem set	*Weighted_arrival* latency			*Max_arrival* latency		
	SA	VNS	Hybrid	SA	VNS	Hybrid
mCMT1	53.7	93.7	156.2	25.4	112.1	185.9
mCMT2	119.9	180.5	286.5	158.2	211.6	319.4
mCMT3	139.2	168.5	301.4	118.3	193	340.5
mCMT4	214.7	276.7	339.4	138.6	283.5	355.9
mCMT5	157.3	421.5	476.8	146.7	402.5	511.7
mGWKC1	118.9	396	465.1	100.8	267.7	372.3
mGWKC2	215.1	512	756.6	129.0	335.9	536.2
mGWKC5	140.9	229.6	389.4	105.9	169.2	278.8
mGWKC6	183.4	351.9	542.5	130.1	237.5	441.2
mGWKC9	364.7	648.5	1,071.2	238.2	429.5	1,008.4
mGWKC10	322.4	788.9	1,387.0	384.0	614.8	1,023.6
mGWKC17	708.9	826	1,267.3	634.5	768.9	1,128.6
mGWKC18	723.8	1,066.3	1,568.5	657.2	966.3	1,459.4

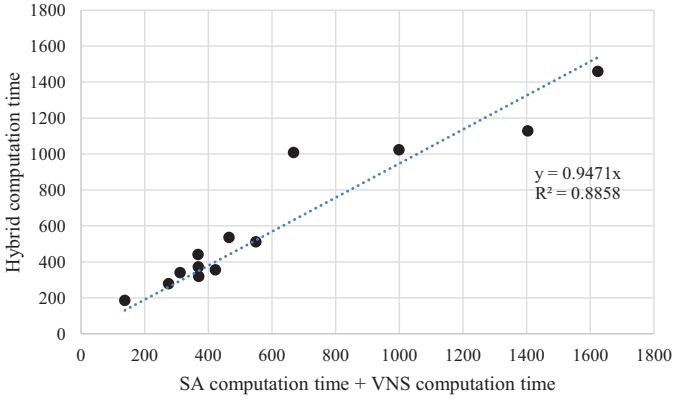

Fig. 9. Comparing computation time of Hybrid and sum of SA and VNS on *max_arrival* latency.

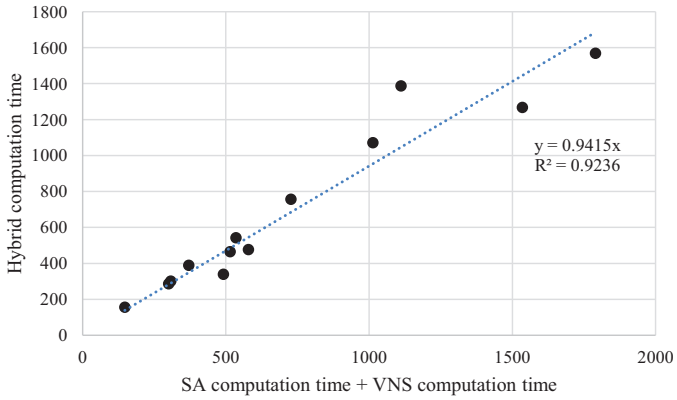

Fig. 10. Comparing computation time of Hybrid and sum of SA and VNS on *weighted-arrival* latency.

Table 9
Evaluation of the Hybrid algorithm's operators on performance improvement.

Problem set	%Gap			
	a) Termination Condition	b) Acceptance Criteria	c) Local Search	d) All three
mCMT1	1.30	1.64	2.40	4.72
mCMT2	4.86	7.12	8.63	7.57
mCMT3	2.47	4.52	4.18	5.22
mCMT4	2.91	3.07	3.64	4.62
mCMT5	3.50	4.48	7.98	9.88
mGWKC1	0.70	0.64	1.18	4.64
mGWKC2	1.39	0.72	2.53	3.33
mGWKC5	4.07	6.32	7.32	8.01
mGWKC6	1.22	1.50	1.94	4.80
mGWKC9	3.72	2.11	3.92	5.52
mGWKC10	3.02	2.28	2.82	4.84
mGWKC17	2.34	4.54	3.08	4.15
mGWKC18	3.35	2.26	2.58	4.18
Avg.	**2.68**	**3.17**	**4.02**	**5.50**

Table 10
Results of the Hybrid algorithm on Campos et al. (2008) problems.

Problem set	a	b	c	d	e	f
CCM1 (50)	3435.8	1517.4	1393.9	1434.1	1390.9	1322.4
Number of vehicles	3	10	15	25	25	40
CCM2 (75)	4742.1	2192	2111.9	2052.8	2096.8	2004.9
Number of vehicles	4	15	22	37	37	60
CCM3 (100)	6150.8	2841.8	2803.4	2917.2	2871.9	2824.2
Number of vehicles	5	20	29	48	49	80
CCM4 (150)	6892.9	4392.8	4391.1	4491.4	4645.2	4507.4
Number of vehicles	8	29	43	71	73	119
CCM5 (199)	8446	5628	6030.5	6013.6	5888.5	5703.2
Number of vehicles	10	38	56	93	96	158
CCM6 (120)	7882.2	6476.7	6468.1	6643.6	6450.6	6568.6
Number of vehicles	6	23	34	56	58	95
CCM7 (100)	6063.3	3151	3192.3	3303.7	3291.4	3282
Number of vehicles	5	20	29	48	49	80

The termination condition is the least effective and the local search is the most effective, while the effect of multiple components is beyond all individual effects. The results indicate the importance of incorporating individual elements together and the improvement could reach almost 10% in some problem instances.

Finally, we used the Hybrid algorithm to solve problems CCM generated by Campos et al. (2008) and the results (best obtained solutions) are given in Table 10 for objective function *Max_arrival*. The number of customers in each network is given in the first column and the number of vehicles (R) used in each problem is the minimum number of vehicles needed, that is, $R = \lceil \frac{\sum_i D_i}{Q} \rceil$. It can be seen that for larger networks, the total latency is larger. How-

ever, the latency is not necessary comparable for problems on the same network because customer demands are generated in different ranges. For example, CCM4e has higher latency than CCM4d although it has more vehicles. The results can be used as future benchmark.

6. Conclusions

In this paper, we considered the Customer-centric, Multi-commodity Vehicle Routing Problem with Split Delivery (CMVRPSD). Two decisions are made in this problem: routing vehicles to customers and quantifying commodities to load and unload. Due to split delivery, we proposed two different latency-based objective functions. Two mathematical models of the problem were presented and compared on several small-size problem instances. The experimental results showed that Model 1 (based on the network flow problem formulation) is less efficient than Model 2 (based on the assignment problem formulation), in terms of computation time.

We presented a hybrid heuristic algorithm which incorporates Simulated Annealing (SA) and Variable Neighborhood Search (VNS). The performance of the hybrid algorithm, SA, VNS, and Nearest Neighborhood (NN) heuristic were compared on large-scale problem instances. The hybrid algorithm outperformed the other three algorithms, while achieving gap less than 2% of the best known solution in the literature.

As future research, one can consider multiple depots that are specialized to specific types of commodities due to the facility constraints or cost effectiveness. The decision in this problem includes assignment of vehicles to depots in addition to those already considered in the CMVRPSD. Incompatibility among commodities can be also considered as a practical constraint. Incompatible commodities cannot be loaded together on a same vehicle, and this constraint would lead to higher latency outcomes.

Acknowledgements

The authors are grateful to the two anonymous referees, whose constructive feedback was helpful in improving the quality of the paper.

References

Afrati, F., Cosmadakis, S., Papadimitriou, C. H., Papageorgiou, G., & Papakostantinou, N. (1986). The complexity of the travelling repairman problem. *RAIRO Inf. théor., 20*(1), 79–87.

Ambrosino, D., & Sciomachen, A. (2007). A food distribution network problem: A case study. *IMA J. Manage Math., 18*(1), 33–53.

Angel-Bello, F., Alvarez, A., & García, I. (2013). Two improved formulations for the minimum latency problem. *Appl. Math. Model., 37*(4), 2257–2266.

Archetti, C., & Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *Int. transact. oper. res., 19*(1–2), 3–22.

Archetti, C., Bianchessi, N., & Speranza, M. G. (2014). Branch-and-cut algorithms for the split delivery vehicle routing problem. *Eur. J. Oper. Res., 238*(3), 685–698.

Archetti, C., Bianchessi, N., & Speranza, M. G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Comput. Oper. Res., 64*, 1–10.

Archetti, C., Bouchard, M., & Desaulniers, G. (2011). Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transport. Sci., 45*(3), 285–298.

Archetti, C., Speranza, M. G., & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transport. Sci., 40*(1), 64–73.

Ban, H. B., Nguyen, K., Cuong Ngo, M., & Nguyen, D. N. (2013). An efficient exact algorithm for the minimum latency problem. *Progr. Inform., 10*, 167–174.

Berbotto, L., García, S., & Nogales, F. J. (2014). A randomized granular tabu search heuristic for the split delivery vehicle routing problem. *Ann. Oper. Res., 222*(1), 153–173.

Bertsimas, D. J., & Van Ryzin, G. (1991). A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Oper. Res., 39*(4), 601–615.

Bianco, L., Mingozzi, A., & Ricciardelli, S. (1993). The traveling salesman problem with cumulative costs. *Networks, 23*(2), 81–91.

Bjelić, N., Vidović, M., & Popović, D. (2013). Variable neighborhood search algorithm for heterogeneous travelling repairmen problem with time windows. *Expert Syst. Appl., 40*(15), 5997–6006.

Blum, A., et al. (1994). The minimum latency problem. In *Proceedings of ACM Symposium on Theory of Computing* (pp. 163–171).

Bulgak, A. A., & Bektas, T. (2009). Integrated cellular manufacturing systems design with production planning and dynamic system reconfiguration. *Eur. J. Oper. Res., 192*(2), 414–428.

Camci, F. (2014). The travelling maintainer problem: Integration of condition-based maintenance with the travelling salesman problem. *J. Oper. Res. Soc., 65*, 1423–1436.

Campbell, A. M., Vandenbussche, D., & Hermann, W. (2008). Routing for relief efforts. *Transport. Sci., 42*(2), 127–145.

Campos, V., Corberán, A., & Mota, E. (2008). A scatter search algorithm for the split delivery vehicle routing problem. *Adv. Comput. Intell. Transport Logist. Supply chain manage., 144*, 137–152.

Chen, P., Dong, X., & Niu, Y. (2012). An Iterated Local Search Algorithm for the Cumulative Capacitated Vehicle Routing Problem. In H. Tan (Ed.), *Technology for education and learning. Advances in intelligent and soft computing* (pp. 575–581). Berlin/Heidelberg: Springer.

Chen, S., Golden, B., & Wasil, E. (2007). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks, 49*(4), 318–329.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The Vehicle Routing Problem. In N. Christofides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial Optimization* (pp. 315–338). Chichester: Wiley.

Coene, S., & Spieksma, F. C. R. (2008). Profit-based latency problems on the line. *Oper. Res. Lett., 36*(3), 333–337.

Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C. R., & Vansteenwegen, P. (2013). Heuristics for the traveling repairman problem with profits. *Comput. Oper. Res., 40*(7), 1700–1707.

Dror, M., Laporte, G., & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Appl. Math., 50*(3), 239–254.

Golden, B., Wasil, E., Kelly, J., & Chao, I. (1998). Metaheuristics in Vehicle Routing. In T. Crainic, & G. Laporte (Eds.), *Fleet Management and Logistics* (pp. 33–56). Boston: Kluwer Academic Publishers.

Heilporn, G., Cordeau, J.-F., & Laporte, G. (2010). The delivery man problem with time windows. *Discrete Optim., 7*(4), 269–282.

Ho, S. C., & Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Comput. Oper. Res., 31*(12), 1947–1964.

Hosseini, S., Al Khaled, A., & Vadlamani, S. (2014). Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem. *Neural Comput. Appl., 25*(7–8), 1871–1885.

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

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *J. Stat. Phys., 34*(5–6), 975–986.

Lee, S. (2011). The role of preparedness in ambulance dispatching. *J. Oper. Res. Soc., 62*(10), 1888–1897.

Lysgaard, J., & Wøhlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *Eur. J. Oper. Res., 236*(3), 800–810.

Martínez-Salazar, I., Angel-Bello, F., & Alvarez, A. (2014). A customer-centric routing problem with multiple trips of a single vehicle. *J. Oper. Res. Soc., 66*(8), 1312–1323.

Mattos Ribeiro, G., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res., 39*(3), 728–735.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Comput. Oper. Res., 24*, 1097–1100.

Mladenović, N., Urošević, D., & Hanafi, S. (2013). Variable neighborhood search for the travelling deliveryman problem. *4OR, 11*(1), 57–73.

Moshref-Javadi, M., & Lee, S. (2013). A taxonomy to the class of minimum latency problems. In *Proceedings of 2013 Industrial and Systems Engineering Research Conference, May 18-22*.

Nagarajan, V., & Ravi, R. (2008). The Directed Minimum Latency Problem. *Proceedings of the 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques* (pp. 193–206).

Ngueveu, S. U., Prins, C., & Wolfler Calvo, R. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res., 37*(11), 1877–1885.

Nowak, M., Ergun, Ö., & White Iii, C. C. (2008). Pickup and delivery with split loads. *Transport. Sci>, 42*(1), 32–43.

Psaraftis, H. N., Solomon, M. M., Magnanti, T. L., & Kim, T.-U. (1990). Routing and scheduling on a shoreline with release times. *Manage. Sci., 36*(2), 212–223.

Rabiee, M., Rad, R. S., Mazinani, M., & Shafaei, R. (2014). An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines. *Int. J. Adv. Manuf. Tech., 71*(5–8), 1229–1245.

Ribeiro, G. M., Laporte, G., & Mauri, G. R. (2012). A comparison of three metaheuristics for the workover rig routing problem. *Eur. J. Oper. Res., 220*(1), 28–36.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *J. ACM, 23*(3), 555–565.

Salehipour, A., Sörensen, K., Goos, P., & Bräysy, O. (2011). Efficient GRASP+ VND and GRASP+ VNS metaheuristics for the travelling repairman problem. *4OR, 9*(2), 189–209.

Sierksma, G., & Tijssen, G. A. (1998). Routing helicopters for crew exchanges on off-shore locations. *Ann. Oper. Res., 76*, 261–286.
Silva, M. M., Subramanian, A., & Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Comput. Oper. Res., 53*, 234–249.
Silva, M. M., Subramanian, A., Vidal, T., & Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. *Eur. J. Oper. Res., 221*(3), 513–520.
Song, S. H., Lee, K. S., & Kim, G. S. (2002). A practical approach to solving a newspaper logistics problem using a digital map. *Comput. Ind. Eng., 43*(1), 315–330.
Stålhane, M., Andersson, H., Christiansen, M., Cordeau, J.-F., & Desaulniers, G. (2012). A branch-price-and-cut method for a ship routing and scheduling problem with split loads. *Comput. Oper. Res., 39*(12), 3361–3375.
Tsitsiklis, J. N. (1992). Special cases of traveling salesman and repairman problems with time windows. *Networks, 22*(3), 263–282.
Wang, H., Du, L., & Ma, S. (2014). Multi-objective open location-routing model with split delivery for optimized relief distribution in post-earthquake. *Transport. Res. Part E: Log, 69*, 160–179.
Xiao, Y., Zhao, Q., Kaku, I., & Mladenovic, N. (2014). Variable neighborhood simulated annealing algorithm for capacitated vehicle routing problems. *Eng. Optim., 46*(4), 562–579.
Xu, Y., & Jiang, W. (2014). An Improved Variable Neighborhood Search Algorithm for Multi Depot Heterogeneous Vehicle Routing Problem based on Hybrid Operators. *Int. J. Cont. Autom., 7*(3), 299–316.