Vehicle Routing Problem: Simultaneous Deliveries and Pickups
with Split Loads and Time Windows

Yong Wang
Ph.D. Student
School of Transportation, Southeast University
Nanjing, China, 210096
Email: yongwx6@gmail.com

Xiaolei Ma
Ph.D. Student
Department of Civil and Environmental Engineering, University of Washington, Seattle, WA, 98195-2700
Email: xiaolm@uw.edu

Yunteng Lao
Ph.D. Student
Department of Civil and Environmental Engineering, University of Washington, Seattle, WA, 98195-2700
Email: laoyunteng@gmail.com

Yinhai Wang (Corresponding Author)
Professor
Box 352700
Department of Civil and Environmental Engineering, University of Washington
Seattle, WA 98195-2700
Tel: (206) 616-2696
Fax: (206) 543-1543
Email: yinhai@u.washington.edu

Haijun Mao
Professor
School of Transportation, Southeast University
Nanjing, China, 210096
Email: maohaijun@seu.edu.cn

Word count: 5363+ 1500 (3 Tables×250+ 3 Figures×250) = 6863
Submitted for Presentation and Publication at the 92nd Annual Meeting of the Transportation Research Board
Submitted on August, 1st, 2012
Resubmitted on November, 1st, 2012
ABSTRACT

The Vehicle Routing Problem with Simultaneous Deliveries and Pickups (VRPSDP) has attracted great interest in research due to its potential cost savings to transportation and logistics operators. There exist several extensions of VRPSDP, and among these extensions, Simultaneous Deliveries and Pickups with Split Loads Problem (SDPSLP) is particularly proposed for eliminating the vehicle capacity constraint, as well as allowing the deliveries or the pickups for each customer to be split into multiple visits. Although delivery and pickup activities are often constrained by time windows, few studies considered such constraints when addressing SDPSLP. To fill the gap, this paper formulates the Vehicle Routing Problem of Simultaneous Deliveries and Pickups with Split Loads and Time Windows (VRPSDPSLTW) as a Mixed Integer Programming (MIP) problem. A hybrid heuristic algorithm is developed to solve this problem. Solomon datasets are applied with minor modifications to test the effectiveness of the solution algorithm. The computational experiment results demonstrate that the proposed algorithm is superior to other solution approaches for VRPSDPSLTW in terms of the total travel cost, number of vehicles, and loading rate. The proposed formulation and solution algorithm for the VRPSDPSLTW problem may serve as a general analytical tool for optimizing vehicle routing in practice.

Key words: Vehicle Routing Problem of Simultaneous Deliveries and Pickups with Split Loads and Time Windows, Hybrid Heuristic Algorithm, Mixed Integer Programming
INTRODUCTION

Vehicle Routing Problem (VRP) is considered as an important issue in both transportation and operational research domains. The VRP can be depicted as an optimal solution-seeking procedure for vehicles serving a set of customers, where the total travel cost is minimized (1). The solution to VRP has been widely applied to reduce the transportation cost for a large number of logistics companies in real life (2). However, the traditional VRP are restricted by several assumptions, for example, the demand of each customer should be less than the vehicle capacity and each customer can be only visited once. These constraints are not able to fully meet the customers’ requirements in many aspects, for example, multiple visits for each customer could happen if single vehicle cannot accommodate the customer’s demand. Traditional countermeasure to deal with each customer’s excessive demand in VRP is to assign another vehicle with a higher capacity (3), but this strategy incurs more operating cost and thereby is infeasible in the practical implications. From customers’ perspective, customers always seek for less waiting time, so the temporal restriction should be also taken into account, known as VRP with time windows. Besides the capacity and time window constraints, customers may receive and send goods simultaneously, and a vehicle may carry a mixture of delivery and pickup goods during its visit of each customer. The split loads restriction complicates the VRP, but benefits both logistics companies and customers in terms of cost savings and convenience.

To release the aforementioned limitations of the traditional VRP, Vehicle Routing Problem of Simultaneous Deliveries and Pickups with Split Loads and Time Windows (VRPSDPSLTW) is modeled as follows: a fleet of vehicles start from a depot and need to serve a set of spatially distributed customers, and are required to return to the depot finally. Each customer may request both delivery and pickup services simultaneously within a time window, and the combined demand is possible to exceed one vehicle’s total or remaining capacity. Hereby, each customer can be visited more than once by a single vehicle on one route, or served by multiple vehicles from different routes.

To consider the constraints imposed on the traditional VRP, this paper aims to model the VRPSDPSLTW to minimize both number of the required vehicles and total travel cost. A mixed integer programming model is firstly established, and next, an initial solution-seeking algorithm
is proposed for the solution initialization, followed by a heuristic algorithm to improve the initial solution.

The rest of this paper is organized into five sections as follows: several VRP related studies are firstly reviewed and summarized, and then, VRPSDPSL TW is formulated with detailed definitions and notations. To solve the VRPSDPSL TW in an efficient and effective manner, a heuristic approach is developed to minimize both the number of vehicles and total travel cost on a basis of model formulation. The approach includes both solution initialization and improvement. Followed by the approach, Solomon’s test data (4) is used for comparisons with other solution algorithms. Finally, conclusions and future works are discussed at the end of this paper.

LITERATURE REVIEW

It is well known that the Vehicle Routing Problem (VRP) is a NP-hard problem, to tackle the VRP in an efficient fashion, a large number of studies have been conducted in the past (1,5,6). In these studies, the VRP is restricted by three assumptions in determining a set of minimum cost routes: firstly, the vehicle routes should start and end at the depot; secondly, every customer needs to be visited once by each vehicle; finally, the total demand of any route should not exceed the capacity of each vehicle. However, these constraints do not adhere to the reality, many researchers has put their endeavors to relax these restrictions, for example, Dror and Trudeau (7,8) removed the second and third conditions in VRP, i.e., the customer’s delivery demand can be split by several vehicles. This type of the VRP is called as the Split Delivery Vehicle Routing Problem (SDVRP). By comparing with the traditional VRP with only one visit permitted for each customer, split delivery is more beneficial for logistics operators in terms of both total distance traveled and number of required vehicles (3, 9). Due to the high complexity of the problem, most researchers resorted to the heuristic-based solution approach (10,11). For instance, Belenguer et al. (2) studied the SDVRP as a polyhedral and give a cutting-plane algorithm to yield good lower bounds and upper bounds for 100 customers. Archetti et al. (11) developed a Tabu search algorithm for the SDVRP. Jin et al. (12, 13) presented a two-stage algorithm with valid inequalities and a column generation approach to successfully solve SDVRP. Lee et al. (14) established a dynamic programming (DP) model with infinite states and action spaces, and then used a best-first shortest path search procedure for SDVRP.
In addition, it is common to see that each customer may request to be served in a certain time period. In this case, SDVRP is further derived as Split Delivery Vehicle Routing Problem with Time Windows (SDVRPTW). Frizzell and Giffin (15) presented two improved heuristic algorithms to solve the SDVRPTW on a grid network. Ho and haugland (16) developed a Tabu search algorithm for the SDVRPTW, the first stage of this algorithm constructed a VRP solution using node interchanges, and the second stage was to improve the VRP solution by adopting and eliminating splits. Belfiore and Yoshizaki (17) proposed a heuristic based initial solution algorithm for heterogeneous SDVRPTW in Brazil, where the capacity of each vehicle is not identical. Salani and Vacca (18) modeled the SDVRPTW as a flow-based mixed integer programming and developed a branch-and-price algorithm to address the mixed integer programming. Archetti et al. (19) also proposed a new enhanced procedure to solve the SDVRPTW through a branch-and-price cut algorithm.

Recently, more and more researchers have shifted their attentions to the Simultaneous Deliveries and Pickups problem, and demonstrate that it is necessary and feasible to find a set of routes where each vehicle conducts a mixed load of delivery and pickup at the same location (20, 21, 22). This type of problem is called the Vehicle Routing Problem with Simultaneous Deliveries and Pickups (VRPSDP). It’s a very common problem in real-life. For example, each supermarket may require goods from a depot, and return unused or outdated goods to the depot at the same time. When combined with the time window constrain, VRPSDP becomes challenging and is considered as the Simultaneous Pickup and Delivery Problem with Time Windows (SPDPTW). There are only a few studies solving this problem. For instance, Nanry and Barnes (23) presented a reactive Tabu search approach to solve the simultaneous pickup and delivery problem with time windows. Lu and Maged (24) proposed a construction heuristic algorithm to solve the pickup and delivery with time windows.

Nevertheless, in the SPDPTW, the quantity of delivery and pickup for each particular customer cannot be split into several visits (25). Relaxing this restriction yields to the SPDPTW with split loads, also known as VRPSDPSLTW defined in the previous context. To the best of our knowledge, no studies have been done to model this special extension of VRP. In our paper, we aim to solve the VRPSDPSLTW in an efficient and effective fashion. The main contributions of this paper are: (1) comprehensibly consider constraints of simultaneously splitting loads for
deliveries, pickups and time windows; (2) provide a hybrid heuristic algorithm through an initial solution to solve this problem.

**MODEL FORMULATION**

Several notations and definitions are defined in this section. \( J = \{1, 2, 3, \ldots, n\} \), \( N_0 \) is the sequence of all elements, where \( N_0 = J \cup \{0\} \), in which 0 represents the depot and 1, 2, 3, \ldots, \( n \) represent \( n \) customers. Each route is defined as a complete sequence of visited customers by each vehicle, where two consecutive customers are linked with each other. Each vehicle departs and returns to the same depot, thereby, the origin and the destination of each route is identical.

The notations used in the formulation are listed as follows.

- **\( Q \)**: Capacity of each vehicle
- **\( V \)**: Set of vehicles, \( k \in V \)
  \[
  V_k = \begin{cases} 
  1 & \text{if vehicle } k \text{ is chosen to serve customers}, \\
  0 & \text{otherwise.} 
  \end{cases}
  \]
- **\( J \)**: Set of customers: \( \{1, 2, 3, \ldots, n\} \), \( \forall i, j \in J \) and \( i \neq j \)
- **\( d_{ij} \)**: Travel cost (travel distance) between customer \( i \) to customer \( j \)
- **\( t_{ij} \)**: Travel time between customer \( i \) to customer \( j \)
- **\( v_{ij} \)**: Travel speed between customer \( i \) to customer \( j \)
- **\( D_j \)**: Delivery demand at customer \( j \)
- **\( R_j \)**: Pickup demand at customer \( j \)
  \[
  x_{ijk} = \begin{cases} 
  1 & \text{if vehicle } k \text{ travels directly from customer } i \text{ to customer } j, \\
  0 & \text{otherwise.} 
  \end{cases}
  \]
- **\( y_{ijk} \)**: Quantity of the customer \( j \)'s goods delivered from vehicle \( k \) by using the route from customer \( i \) to customer \( j \)
$z_{ijk}$: Quantity of the customer $j$'s goods picked up from vehicle $k$ by using the route from customer $i$ to customer $j$

$y_{ijk}'$: Remaining quantity of the delivery goods from customer $i$ to customer $j$ for vehicle $k$

$[a_i, b_i]$: The time window for each customer $\forall i \in N_0$

$r_{ik}$: The time when vehicle $k$ starts to service customer $i$

Model Formulation

Min $F_1(x) = \sum_{k \in V} |V_k| \sum_{j \in N_0} x_{0jk}$ \hspace{1cm} (1)

Min $F_2(x) = \sum_{k \in V} \sum_{i \in N_0} \sum_{j \in N_0} d_{ij} x_{ijk}$ \hspace{1cm} (2)

Subject to

\[ \sum_{j \in J} x_{0jk} = 1 \hspace{1cm} \forall k \in V \] \hspace{1cm} (3)

\[ \sum_{i \in N_0} x_{nik} - \sum_{j \in N_0} x_{wjk} = 0 \hspace{1cm} \forall u \in J, \forall k \in V \] \hspace{1cm} (4)

\[ \sum_{i \in N_0} x_{ijk} \geq 1 \hspace{1cm} \forall j \in J, \forall k \in V \] \hspace{1cm} (5)

\[ \sum_{k \in V} \sum_{i=0, i\neq j}^n y_{ijk} = D_j \hspace{1cm} \forall j \in J \] \hspace{1cm} (6)

\[ \sum_{k \in V} \sum_{i=0, i\neq j}^n z_{ijk} = R_j \hspace{1cm} \forall j \in J \] \hspace{1cm} (7)

\[ \sum_{k \in V} \sum_{i \in N} x_{ijk} \geq 1 \hspace{1cm} \forall j \in J, \forall k \in V \] \hspace{1cm} (8)

\[ y_{ijk}' + \sum_{i=0, i\neq j}^n z_{ijk} \leq Q \hspace{1cm} \forall j \in J, \forall k \in V \] \hspace{1cm} (9)

\[ t_{ij} = \frac{d_{ij}}{v_{ij}} \hspace{1cm} \forall i, j \in J \] \hspace{1cm} (10)

\[ r_{ik} + t_{ij} - (b_i + t_{ij} - a_j)(1-x_{ijk}) \leq r_{jk} \hspace{1cm} \forall j \in J, \forall i \in N_0, \forall k \in V \] \hspace{1cm} (11)
\begin{align}
    a_i & \leq r_i \leq b_i \quad \forall i \in N_0, \forall k \in V \tag{12} \\
    r_{ik} + t_{i0} - K_{i0} (1-x_{i0k}) & \leq b_0 \quad \forall i \in J, \forall k \in V \tag{13} \\
    r_{0k} &= a_0 \quad \forall k \in V \tag{14} \\
    x_{i0k} &= 0 \quad \forall i \in N_0, \forall k \in V \tag{15} \\
    x_{ijk} & \in \{0,1\} \quad \forall i, j \in N_0, \forall k \in V \tag{16}
\end{align}

The objective function (1) is to minimize the vehicle number, and the objective function (2) is to minimize the total travel cost. Constraint (3) ensures that the vehicle \( k \) leaves the depot and will visit the customer \( j \) in the set of customers \( J \). Constraint (4) implies that any vehicle will leave to serve another customer after serving a certain customer, until finally return to the depot. Constraint (5) states that a customer can be served more than once by a single vehicle. Constraint (6) and Constraint (7) illustrate the total delivery/pickup demand at customer \( j \) is a sum of complete delivery/pickup quantity of goods on several routes at customer \( j \). Constraint (8) means that a customer can be served more than once by multiple vehicles. Constraint (9) ensures the sum of remaining quantity of delivery goods and existing quantity of pickup goods by vehicle \( k \) is less than the capacity of vehicle. Constraint (10) computes the travel time as the ratio of travel distance and travel speed. Constraint (11) ensures that the vehicle \( k \) should arrive at customer \( j \) after time \( r_{ik} + t_{i0} \) if the vehicle \( k \) choose the route from customer \( i \) to customer \( j \), and travel time of the vehicle \( k \) from customer \( i \) to customer \( j \) is less than the time difference \( a_j - b_i \) if the vehicle \( k \) does not choose the route from customer \( i \) to customer \( j \). Constraint (12) guarantees that all customers are served within the time window. Constraint (13) ensures that each vehicle should arrive at the depot before the depot is closed. Constraint (14) ensures that all vehicles are dispatched from the depot after the open time \( a_0 \).

**HEURISTIC METHOD**

Our solution approach contains a two-step process. First, we compute an initial feasible solution on the basis of travel time and customer’s waiting time, along with quantity of delivery and pickup. Next we design a hybrid heuristic algorithm to further improve the initial solution.
Heuristic Algorithm for an Initial Solution
To solve the delivery and pickup problem with split loads and time windows modeled by the mixed integer programming, a heuristic algorithm is proposed in the section. Several notations are detailed as follows.

- $\theta_i$: The time when a vehicle starts to serve customer $i$
- $j^*$: The spatially closest unvisited customer from customer $i$
- $N^*$: Set of unvisited customers
- $DL$: Quantity of delivery goods carried by each vehicle
- $PL$: Quantity of pickup goods carried by each vehicle
- $total_{\text{delivery}}$: Total delivery demand of all customers
- $total_{\text{pickup}}$: Total pickup demand of all customers

In the heuristic algorithm, $total_{\text{delivery}}$ and $total_{\text{pickup}}$ varies by time, and their initial values represent the sum of delivery and the sum of pickup for all customers respectively, and are updated as the requirement of each customer gets satisfied. Similarly $DL$ and $PL$ are also dynamic. We assume that vehicle $k$ leaves the depot with a certain amount of delivery goods as $DL$, and it is very likely that the $DL$ is reduced and $PL$ is increased as the vehicle $k$ traverses each customer, until the vehicle $k$ finally returns to the depot, it should ideally carry no $DL$ and a certain amount of $PL$. Especially, when the customer $j^*$ is served, but vehicle $k$ is not able to accommodate the delivery/pickup demand of customer $j^*$, then the requirement of customer $j^*$ is still be partially satisfied by loading the remaining quantity of delivery/pickup goods. The rest of the demand for customer $j^*$ will be further fulfilled either by vehicle $k$ during its returning route or other vehicles on next routes.

The heuristic algorithm process is proposed as follows.

Step 1. Algorithm is initialized and set the number of iterations as $\eta = 1$. 
Step 2. Set $j^* = 0$, $\theta_0 = a_0$, $DL = Q$, $PL = 0$. The vehicle $k$ starts from the depot, if the remaining quantity of delivery goods is less than the capacity of vehicle $Q$, then $DL$ is equal to the remaining quantity of delivery goods.

Step 3. Set $i = j^*$, and find the unvisited customer $j$ that are within the time window and spatially closest to customer $i$. The customer is denoted as (15):

$$j^* \in \arg \min_{j \in \mathcal{N}} \{ t_j + \max \{ a_j - \theta_j, 0 \} \}$$

If more than one unvisited customer is found, then the customer $j^*$ with the minimum travel cost is selected.

Step 4. If the delivery demand of customer $j^*$ is less than the vehicle’s spare capacity, then set $DL = DL - D_j$, $D_j = 0$.

Step 4.1. If the pickup demand of customer $j^*$ is less than the vehicle’s spare capacity ($R_j \leq Q - DL - PL$), then set $PL = PL + R_j$, $R_j = 0$.

Step 4.2. If the pickup demand of customer $j^*$ is more than the vehicle’s spare capacity ($R_j > Q - DL - PL$), then set $R_j = R_j - (Q - DL - PL)$, $PL = Q - DL$.

Step 5. If the delivery demand of customer $j$ is more than the vehicle’s spare capacity, then set $D_j = D_j - DL$, $DL = 0$.

Step 5.1. If the pickup demand of customer $j$ is less than the vehicle’s spare capacity ($R_j \leq Q - PL$), then set $PL = PL + R_j$, $R_j = 0$.

Step 5.2. If the pickup demand of customer $j$ is more than the vehicle’s spare capacity ($R_j > Q - PL$), then set $R_j = R_j - (Q - PL)$, $PL = Q$.

Step 6. Insert the customer $j^*$ into the current route $k$, and the time when customer $j^*$ is visited can be expressed as:
$\theta_j = \theta_i + t_{ij} + \max\{a_j - \theta_i - t_{ij}, 0\}$

**Step 7.** Compute the spare capacity of the vehicle $k$, and return to **Step 3** until $PL = Q$ or no more insertions are valid.

**Step 8.** Set $\eta = \eta + 1$ and return to **Step 2**, until all customers are served.

With the above solution-seeking algorithm complete, the initial series of routes is acquired.

**A Hybrid Heuristic Algorithm for Improving the Initial Solution**

Local search method is gaining more and more popularity to solve VRPSDP (20, 26). Inspired by the local search algorithm, we apply the concepts from the local search method into the procedure of finding the VRPSPSLTW solution. Several necessary operators are illustrated as follows (21, 27).

1. **Relocate Operator**

   The relocate procedure is applied for between-route improvement. In the procedure, we suppose that customer $i$ belongs to route $R_k$, and customer $j$ belongs to route $R_l$. Customer $i$ is removed from $R_k$ and inserted after $j$ in route $R_l$, then, the new routes are denoted as $R_k' = (0, \ldots, i-1, i+1, \ldots, 0)$ and $R_l' = (0, \ldots, j, i, j+1, \ldots, 0)$. The above insertion is named as the Relocate Operator.

2. **Relocate Split Operator**

   Suppose that the demand of customer $i$ is split by route $R_k$ and route $R_l$, that is customer $i \in R_k \cap R_l$, in addition, customer $j$ belongs to route $R_k$, we remove customer $i$ from route $R_l$, denoted as $R_l = (0, \ldots, i-1, i+1, \ldots, 0)$, and split the demand of customer $j$ by route $R_k$ and route $R_l$, let $R_k$ remain unchanged. Increase the amount of customer $i$’s delivered and collected goods in $R_k$ by the same amount of customer $i$’s delivered and collected goods in $R_l$, and transfer the same quantity of delivered and collected goods at customer $j$ from $R_k$ to $R_l$.

3. **2-exchange Operator**
The 2-exchange operator is also known as the 2-opt* procedure introduced by Potvin et al. (28). At the beginning, we suppose the customer \( i \in R_k \) and customer \( j \in R_l \), and the route \( R_k \) and route \( R_l \) are denoted as \( R_k = (0, \cdots, i, i+1, \cdots, 0) \) and \( R_l = (0, \cdots, j, j+1, \cdots, 0) \). The new routes become \( R'_k = (0, \cdots, i, j+1, j+2, \cdots, 0) \), \( R'_l = (0, \cdots, j, i+1, i+2, \cdots, 0) \) with the 2-exchange operator.

4. Swap move operator

The swap move operator is described as an interchange of customers between two different routes. In the procedure, the selected sequence is moved from one route to another route or swapped between two routes with both split delivery and pickup. For example, we interchange customers \( i \in R_k \), \( h \in R_k \) and \( j \in R_l \) between route \( R_k \) and route \( R_l \). The customer \( i \) is inserted in route \( R_l \) and the customer \( j \) is inserted in a certain position of route \( R_k \). In order to balance the delivery and pickup quantity, the demand of customer \( h \) may be split by route \( R_k \) and route \( R_l \). \((m,n)\) swap move operator refers to interchange \( m \) customers and \( n \) customers between two different routes.

5. 2-opt operator

This procedure was initially proposed by Lin (29). In the procedure, two links that are not contiguous with each other are disconnected from the same route and reconnected in all possible ways. For example, the original route is \( R_k = (0, \cdots, i, i+1, g, \cdots, j, j+1, \cdots, 0) \), after the procedure, the new route is \( R'_k = (0, \cdots, i, j, g, i+1, \cdots, j+1, \cdots, 0) \).

6. Or-opt operator

Or-opt operator was introduced by Or (30). This operator is to adjust the sequence of customers within each route. In other words, two or one consecutive customer of any sequence with three consecutive customers is removed and inserted to another location in the same route. For instance, the original route is \( R_k = (0, \cdots, i, i+1, i+2, \cdots, g, \cdots, 0) \), after this procedure, the new route is \( R'_k = (0, \cdots, i, i+1, \cdots, i+2, g, \cdots, 0) \) or \( R'_k = (0, \cdots, i, \cdots, g, i+1, i+2, \cdots, 0) \).
**Hybrid Heuristic Algorithm**

Based on the above description of each operator, the hybrid heuristic solution algorithm is detailed as follows:

**Step 1.** Calculate the initial solution from the heuristic algorithm, and record the number of required routes (vehicles) and the total travel cost. Set iteration number as $n = 1$.

**Step 2.** Select those routes less than or equal to $\varphi$ customers as $N$, where $\varphi$ is a predetermined number of customers. Utilize the relocate operator to insert all customers from routes $N$ into other non-empty routes, and recalculate the total travel cost for these new routes, if the total travel cost is reduced, then the new routes are formed. Repeat the procedure until no more improvements can be found over the best known solution within $\beta$ consecutive iterations.

**Step 3.** Rearrange these routes from **Step 2** into a particular order, and each permutation contains a distinct sequence of routes. The number of permutations is defined as $p$.

**Step 4.** For each permutation, start from the first route, two in pairs, and repeat the relocate split procedure until no more improvements can be found over the best known solution within $\lambda$ consecutive iterations.

**Step 5.** Similar to **Step 4**, repeat the 2-exchange procedure until no more improvements can be found over the best known solution within $\lambda$ consecutive iterations.

**Step 6.** Repeat the 2-opt procedure within each route until no more improvements can be found over the best known solution within $\mu$ consecutive iterations.

**Step 7.** Repeat the or-opt procedure within each route until no more improvements can be found over the best known solution within $\mu$ consecutive iterations.

**Step 8.** Perform the (2, 0), (2, 1) and (2, 2) swap move recursively until no more improvements can be found over the best known solution within $\lambda$ consecutive iterations.

**Step 9.** Repeat the relocate split procedure until no more improvements can be found over the best known solution within $\lambda$ consecutive iterations.

**Step 10.** Repeat or-opt and 2-opt procedures respectively until no more improvements can be found over the best known solution within $\mu$ consecutive iterations. Record the best vehicle number as $F$, and total travel cost as $F$, $n' = n' + 1$ and determine whether $n'$ reaches the maximum number of iterations $\gamma$, if it’s not, select the best routes set, and return to **Step 3**, otherwise, the hybrid heuristic algorithm ends.
Both between-routes and within-route operations exist in the hybrid heuristic algorithm. At the beginning, permutation of the initial routes diversifies the candidate solutions, and then, between-routes operations (relocate, relocate split, 2-exchange and swap move) are applied to improving the visited customer sequence, and this step is considered as a coarse-tuning process. With each between-routes procedure completed, within-route operation is required to exchange the order of visited customers in each route for a fine adjustment. Both between-routes and within-route operations are recursively executed until the best solution is found.

**COMPUTATIONAL RESULTS**

This section presents the heuristic methods’ computational results tested by the modified Solomon Data (23). The Hybrid Heuristic Method (HHM) (20), the Construction Heuristic Algorithm (CHA) (24) and the Reactive Tabu Search Algorithm (RTSA) (23) are selected to compare with our proposed heuristic method.

**Modified Solomon Dataset**

The Solomon datasets (4) are widely utilized as a benchmark to compare a number of vehicle routing problems and their extensions (16, 23). However, Solomon datasets are not initially designed for VRPSDPSLTW. Necessary modifications are needed to test our algorithms.

For all instances of classes R1, C1 and RC1. The delivery demands from R101–R108 can be assumed as the pickup demands of RC101–RC108. Following the similar logic, we reconstruct the pickup demands of R101–R108, C101–C108 using the delivery demands of C101–C108, RC101–RC108 respectively. The new composed datasets are defined as RRC101–RRC108, RCC101–RCC108 and CRC101–CRC108 respectively for testing and comparison purpose. In addition, we assume the distance unit is mile.

**Parameter Settings for Heuristic Method**

Several parameters are necessary to predefine before the algorithm execution. Parameter settings are chosen based on the studies by Ho and Haugland (16), Nanry and Barnes (23), Salani and Vacca (18) on the related Vehicle Routing Problem with Time Windows and Split Delivery (VRPTWSD) and Discrete Split Delivery Vehicle Routing Problem with Time Windows (DSDVRPTW) problems.
1. $p = 100$ is the number of permutations used to increase the diversity of initial routes.
2. $\varphi = 7$ is the number of customers to insert into other non-empty routes.
3. $\beta = 6$ is the maximum iterations as a stopping criterion to insert customers into other non-empty routes.
4. $\lambda = 10$ is the maximum iterations to get the best known solution without any improvement between two routes.
5. $\mu = 5$ is the maximum iterations to get the best known solution without any improvement within routes.
6. $\gamma = 1000$ is the maximum iterations to end the hybrid heuristic algorithm.
7. $v_{ij}$ is the time dependent travel speed from customer $i$ to customer $j$ (31, 32). The travel speed varies by the time of day as follows:
   a) Travel time is set as $v = 50\text{mph}$ from 4:01 to 7:00.
   b) Travel time is set as $v = 30\text{mph}$ from 7:01 to 10:00.
   c) Travel time is set as $v = 40\text{mph}$ from 10:01 to 15:00.
   d) Travel time is set as $v = 30\text{mph}$ from 15:01 to 18:00.
   e) Travel time is set as $v = 40\text{mph}$ from 18:01 to 21:00.
   f) Travel time is set as $v = 60\text{mph}$ from 21:01 to 4:00.

Results Analysis and Comparison

For the comparison purpose, we also implement and test HHM (20), CHA (24) and RTSA (23) using the $RRC101-RRC108$, $RCC101-RCC108$ and $CRC101-CRC108$ from the Solomon dataset respectively. The comparison results are shown as Tables 1-3 and Figures 1-3. The measures of effectiveness from the results include the total travel cost, number of vehicles and average loading rate (the ratio of the delivery/pickup loads and vehicle capacity). Based on the results, one can find that

- The average loading rate from the proposed algorithms is greater than that of the other three compared algorithms. For example, the average loading rate on $RRC101-RRC108$ ranges from 85% to 93% from the proposed algorithms, and 76% to 84% from the HHM.
- The number of vehicles from the proposed algorithms is smaller than that of the other three compared algorithms. For example, the number of vehicles decreases 1 to 3 from compared algorithms to the proposed algorithms.
• The travel costs from the proposed algorithms decrease in all 24 scenarios, compared to the other three algorithms. For example, the cost in the scenario of \( RRC101 \) decreases from 1815.32 mile to 1611.72 mile by using the proposed algorithms.

In addition, to evaluate the effectiveness of split loads, we also implemented the proposed algorithms with and without split loads. Compared with the scenario without split loads, on average, the proposed algorithm with split loads reduces the travel cost by 10%, requires less vehicles by 20%, and increases more than 10% loading rate per vehicle. Therefore, the proposed algorithm with split loads presents its superiority in terms of reducing the operational cost for those logistics companies.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Proposed algorithms</th>
<th>HHM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Travel cost</td>
<td>No. of vehicles</td>
</tr>
<tr>
<td>RRC101</td>
<td>1611.72</td>
<td>14</td>
</tr>
<tr>
<td>RRC102</td>
<td>1535.65</td>
<td>13</td>
</tr>
<tr>
<td>RRC103</td>
<td>1272.31</td>
<td>12</td>
</tr>
<tr>
<td>RRC104</td>
<td>1112.42</td>
<td>9</td>
</tr>
<tr>
<td>RRC105</td>
<td>1516.18</td>
<td>12</td>
</tr>
<tr>
<td>RRC106</td>
<td>1283.34</td>
<td>12</td>
</tr>
<tr>
<td>RRC107</td>
<td>1102.56</td>
<td>9</td>
</tr>
<tr>
<td>RRC108</td>
<td>1032.89</td>
<td>10</td>
</tr>
<tr>
<td>Overall</td>
<td>1308.38</td>
<td>11.38</td>
</tr>
</tbody>
</table>

**TABLE 1 Comparison between proposed algorithms and HHM on RRC101 – RRC108**
FIGURE 1 Comparison between proposed algorithms and HHM on $RRC101 - RRC108$

TABLE 2 Comparison between proposed algorithms and CHA on $RCC101 - RCC108$

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Proposed algorithms</th>
<th>CHA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Travel cost</td>
<td>No. of vehicles</td>
</tr>
<tr>
<td>RCC101</td>
<td>1655.31</td>
<td>17</td>
</tr>
<tr>
<td>RCC102</td>
<td>1512.92</td>
<td>14</td>
</tr>
<tr>
<td>RCC103</td>
<td>1326.78</td>
<td>12</td>
</tr>
<tr>
<td>RCC104</td>
<td>1095.33</td>
<td>10</td>
</tr>
<tr>
<td>RCC105</td>
<td>1397.16</td>
<td>12</td>
</tr>
<tr>
<td>RCC106</td>
<td>1263.86</td>
<td>12</td>
</tr>
<tr>
<td>RCC107</td>
<td>1157.02</td>
<td>10</td>
</tr>
<tr>
<td>RCC108</td>
<td>1022.41</td>
<td>10</td>
</tr>
<tr>
<td>Overall</td>
<td>1303.85</td>
<td>12.13</td>
</tr>
</tbody>
</table>
FIGURE 2 Comparison between proposed algorithms and CHA on RCC101 – RCC108

TABLE 3 Comparison between proposed algorithms and RTSA on CRC101 – CRC108

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Proposed algorithms</th>
<th>RTSA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Travel cost</td>
<td>No. of vehicles</td>
<td>Average loading rate</td>
</tr>
<tr>
<td>CRC101</td>
<td>858.82</td>
<td>11</td>
<td>88%</td>
</tr>
<tr>
<td>CRC102</td>
<td>828.94</td>
<td>10</td>
<td>91%</td>
</tr>
<tr>
<td>CRC103</td>
<td>906.26</td>
<td>11</td>
<td>85%</td>
</tr>
<tr>
<td>CRC104</td>
<td>862.12</td>
<td>10</td>
<td>87%</td>
</tr>
<tr>
<td>CRC105</td>
<td>841.64</td>
<td>12</td>
<td>82%</td>
</tr>
<tr>
<td>CRC106</td>
<td>879.73</td>
<td>10</td>
<td>92%</td>
</tr>
<tr>
<td>CRC107</td>
<td>838.56</td>
<td>10</td>
<td>88%</td>
</tr>
<tr>
<td>CRC108</td>
<td>828.94</td>
<td>10</td>
<td>91%</td>
</tr>
<tr>
<td>Overall Average</td>
<td>855.63</td>
<td>10.5</td>
<td>88%</td>
</tr>
</tbody>
</table>
In summary, the total travel cost, number of required vehicles and average loading rate are significantly improved using our proposed heuristic method. Especially for RRC data, more than 11.5% travel cost is reduced, 18% less vehicles are needed, and each vehicle is better utilized to load delivery and pickup goods with an increased average loading rate of 12.8%. These evidences demonstrate that our proposed approach outperforms other algorithms in terms of both travel cost (total travel mileage) and operating cost (number of vehicles and loading rate) reduction.

The improvements of the proposed algorithms are due to two major reasons. One of the reason is that more realistic factors are introduced in the problem formulation, for example, by splitting both delivery and pickup demands, each customer is allowed to be served more than once, even the customer’s demand exceeds the total or remaining vehicle capacity, the vehicle is still able to satisfy customers’ partial requirements by delivering or picking up the remaining loads. Thereby, this strategy gains more benefits for transportation and logistics operators. Another reason is that the hybrid heuristic algorithm used for improving the initial solution can further reduce the travel cost. For instance, the relocate operator in the hybrid heuristic algorithm
can reduce travel cost by relocating the customer into another different route. Meanwhile, the vehicles with a small number of customers may be removed after relocating all their customers to other routes. In this situation, the number of vehicles needed for deliveries and pickups can be reduced, and thus, the total cost will be decreased. Moreover, the relocate split operator in the hybrid heuristic algorithm can also reduce travel cost by relocating the split location.

CONCLUSIONS

This study aims to tackle the Vehicle Routing Problem of Simultaneous Deliveries and Pickups with Split Loads and Time Windows (VRPSDPSLTW). Due to the multiple complex restrictions, VRPSDPSLTW is considered as a challenge issue, and very scarce studies have been conducted in the past. We attempt to establish a Mixed Integer Programming (MIP) formulation to reflect both the split loads and time window constraints for VRP with simultaneous deliveries and pickups. To solve MIP in an effective manner, a heuristic algorithm was proposed to obtain an initial solution. A hybrid heuristic algorithm is further developed to improve the initial solution based on the local search method. The computational experiments are conducted using the modified Solomon datasets \textit{RRC101 – RRC108}, \textit{RCC101 – RCC108} and \textit{CRC101 – CRC108}, and demonstrates the superiority of combining both the heuristic algorithm and hybrid heuristic solution algorithm to solve MIP. The proposed heuristic method has significantly reduced the total travel cost and number of required vehicles, and increases the average loading rate. Especially in the scenarios of \textit{RRC} data, more than 11.5% travel cost is reduced, 18% less vehicles are needed, and 12.8% more average loading rate is increased. The results also imply that splitting deliveries and pickups is beneficial for transportation and logistics operators to reduce the number of vehicles, total travel cost and increase the vehicle loading rate.

One challenge for the implementation of simultaneous deliveries and pickups with split loads is that customers may be reluctant to allow multiple visits for delivery trucks. Although it is found that splitting the deliveries is the most beneficial when mean customer demand is half of the vehicle capacity and little demand variance exists (3), customers’ willingness may still impact the revenue for the logistics operators. Further research on this issue is to incorporate additional constraint into the optimization procedure, such as considering the maximum number of visits for each customer. Besides the customers’ reactions to split deliveries and pickups, customers’ geospatial location and properly clustering the customers also play a vital role for the deliveries and pickups strategy making. Categorizing the customers with similar behaviors into one cluster will potentially enhance the delivery efficiency and reduce the operational costs for logistics companies. Application of VRPSDPSLTW in the real situation will be the future study for this research.
ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China (Project No. 51028802). Special thanks to Jianxin Fan for her valuable suggestions and revisions.
REFERENCES


