The Split Delivery Vehicle Scheduling Problem
With Time Windows and Grid Network Distances

P.W. Frizzell  J.W. Giffin
Department of Mathematics,
Massey University

Abstract

We consider an extension of the Split Delivery Vehicle Routing Problem, whereby customers may have a time window on their delivery. A construction heuristic is developed which uses a look-ahead approach to solve the Split Delivery Vehicle Scheduling Problem with Time Windows. Two improvement heuristics are also given: one attempts to move customers between routes, while the other exchanges customers from one route to another. All three heuristics are implemented on a specifically developed data set and on some standard problems from the literature. Our heuristics also consider the possibility of multiple time windows, grid network distances, non-linear delivery times and bounds on the size of any split deliveries.

1 Introduction

In this paper we consider the Split Delivery Vehicle Scheduling Problem with Time Windows (SDVSPTW). Time windows extend the generic Vehicle Routing Problem (VRP) by requiring that some (or all) customers must receive their delivery within a certain time interval (or intervals). Conversely, split deliveries are a relaxation of VRP constraints in allowing the delivery of a customer to be split between two or more vehicles. With time windows, we consider scheduling, as opposed to routing, problems i.e, we permit the assignment of vehicles to more than one route.

We develop construction and improvement heuristics to obtain solutions to the SDVSPTW. Several special features are contained in our heuristics, such as: multiple time windows, grid network distances, multiple-day problems split into periods of length one day, delivery time defined as a non-linear function of the amount being delivered and use of bounds on the size of any split deliveries. We present the results of a computational study into the effectiveness of our heuristics applied to specially developed data sets and to standard problems from the literature, and draw conclusions.

2 Problem Overview

The basic form of the VRP consists of: a fleet of a given number of delivery vehicles, operating from a single depot (distribution centre); each vehicle in the fleet has the same known capacity (with the possibility for the incorporation of an operating cost per unit distance travelled and a maximum total distance allowable per route); a set of customers, each of which requires the delivery of a known amount of (homogeneous) product and
an objective of satisfying all customer requirements in a manner that minimises the total distance travelled over the planning horizon.

Bodin et al. [1] provide an excellent summary of the VRP, and many of its extensions. Assad and Golden [2] give a survey of more recent work on the VRP. To our knowledge, no work has yet been published on the SDVSPTW. However, both split deliveries and time windows have been considered in isolation, see e.g., Haimovich and Rinnooy Kan [3], Dror and Trudeau [4, 5], Frizzell and Giffin [6], Solomon and Desrosiers [7], Desrochers et al. [8] and Van Landeghem [9]. To ascertain our heuristics’ performance in relation to other work in the literature, they are applied to a series of standard problems generated for the VRPTW by Solomon [10] (Koskosidis et al. [11] and Baker and Schaffer [12] also implement their heuristics on these test problems). Solomon [13] implements several heuristics and concludes that an insertion-type heuristic performs well on the problems solved. Baker and Schaffer [12] use branch exchange procedures to improve upon various initial solutions. Koskosidis et al [11] consider optimization based heuristics which extend the cluster first / route second heuristic of Fisher and Jaikumar [14].

Desrochers et al. [15] develop a classification scheme for the VRP and many of its extensions and relaxations. Under their scheme, the problems we address may be described as $1, mw|cap|/\sum T_i, \sum c_i, \sum c_j$.

It is straightforward to obtain a formulation for the SDVSPTW by modifying existing formulations such as those of Dror and Trudeau [4, 5] and Desrochers et al. [8].

3 Description of the Heuristics

Two options available for the introduction of split deliveries to the VSPTW are the improvement of existing VSPTW solutions by introducing split deliveries or the development of constructive heuristics which allow for split deliveries. We chose to investigate the second option as the structure imposed by VSPTW solutions may be unsuitable for obtaining good SDVSPTW solutions.

To constructively solve the SDVSPTW we split the problem into time slots which are considered sequentially. The construction heuristic utilises a look-ahead approach, called Dynamic Urgency Classification (DUC), which takes into account the amount of time required to make a delivery to a customer (i.e., distance of a customer from its nearest available vehicle, delivery time and, possibly, waiting time), and the amount of time available before the end of that customer's time window. A DUC value is then assigned to each customer according to the following criteria:

1: All remaining demand must be delivered in the next time slot.
2,3: All remaining demand must be delivered in the next 2,3 time slots respectively.
4: Demand may, or need, be delivered, but it is not essential in the next 3 time slots.
5: No delivery may be assigned in the next time slot.
6,7: Some demand must be delivered in the next time slot (these are special cases for multiple time windows, where the next time window is either too short or too close to the beginning of a day for a customer's entire demand to be delivered).
Also included are return to depot checks, which attempt to reduce the overall waiting time. A vehicle is forced to return to the depot if either of the following criteria is satisfied:

- The vehicle has been idle at a customer location for longer than some predetermined default time.
- Where the remaining load is less than some predefined amount which is dependent upon the bounds on the size of any delivery splits, and the average demand for the problem.

A post-processor is also used to remove any unnecessary waiting times.

The resulting SDVSPTW construction heuristic can be described as follows:

**SDVSPTW CONSTRUCTION HEURISTIC**

**Step 1:** Set the current time slot beginning at 0 and set the initial vehicle fleet size. Order customers by their non-decreasing shortest path distance from the depot.

**Step 2:** FOR all vehicles DO a return to depot check.

**Step 3:** Assign all customers a DUC for the present time slot.
Assign demand to all customers with a DUC of 1, 6 or 7 (if the current vehicle fleet cannot service all of these customers fleet size must be augmented).
IF any vehicles from the current fleet are available THEN assign all demand possible of customers with a DUC of 2, 3 or 4 (in that priority order).

**Step 4:** IF it is the end of a day THEN return all vehicles to the depot.
IF it is the end of the problem THEN return all vehicles to the depot, remove any unnecessary waiting times and STOP. OTHERWISE go to step 2.

The main objective of the construction heuristic is to minimise total time taken, which could lead to a relatively large number of split deliveries. We also developed two heuristics which improve the final SDVSPTW construction heuristic output. The first is an adaptation of the exchange heuristics initially developed by Lin and Kernighan [16] for the TSP. It attempts to remove a customer from a vehicle schedule and insert that customer in another vehicle schedule (if a saving in total time travelled would result), and may be described as follows:

**MOVECUSTOMERS IMPROVEMENT HEURISTIC**

**Step 1:** Initialise a counter, $c := 1$ and a label, $insert := 0$. Designate all customers as unchecked.

**Step 2:** Select an unchecked customer, $j$, from a size $c$ schedule. IF no such customer exists THEN go to Step 4.

**Step 3:** Check whether the selected customer may be inserted elsewhere at a saving. IF possible, THEN insert the customer in the position resulting in the largest saving and set $insert := 1$.
Customer $j$ is now checked. Go to Step 2.
Step 4: Increment $c$ by 1.

IF $c >$ the largest vehicle schedule THEN IF $\text{insert} = 1$ THEN go to Step 1 ELSE STOP (the heuristic is finished).

IF $c \leq$ the largest vehicle schedule THEN go to Step 2.

The other improvement heuristic used is similar to that of Van Landeghem [9]. The basic heuristic description is the same as for movcustomers, except, instead of attempting to insert a customer, we attempt to exchange any two customers between any two schedules in the problem.

4 Computational Experiment

4.1 Problem Generation

To investigate the performance of our heuristics, we generated 6480 problems. The heuristics are coded in C, and have been implemented on a Sparc Station SLC. All relevant data is measured in time units, i.e., time for customer deliveries, time for reloading vehicles and a time equivalence for distance travelled.

Several special features are considered in our heuristics:

- All of our computational work is implemented on grid networks.
- Our problems are split into sections, each of which may conveniently correspond to one day. We also impose the extra requirement that every vehicle must return to the depot at the end of a section.
- Our model considers lower and upper bounds on the size of any split deliveries to a customer, as in Frizzell and Giffin [6].
- We permit, at most, two time windows for a customer's delivery.
- Delivery time is represented by a non-linear function of the delivery quantity, which exhibits economies of scale. This enables us to explicitly consider the extra cost (in units of time) of allowing any split deliveries.

Problems were generated by randomly assigning customers to locations on the arcs of each grid network, with the single depot located on the arc closest to the euclidean median of the customer set. The capacity of each vehicle is set at 500. Customer numbers considered are 50, 100, 150 and 200, and problem lengths used are 1, 2 and 5 days. Six different demand set types are included, drawn from a uniform random distribution in the ranges 25-50, 50-150, 50-250, 50-450, 250-450, 350-450. Five bound set types are used, defined as in [6].

Three different time window sets are used. The mid-point of each time window is drawn from a uniform random distribution, while the length is drawn from a random normal distribution in the ranges 0.5-2, 0.5-8 and 4-8 hours. The proportion of customers with multiple time windows is varied from 0-60 percent, depending on the length of the problem and the tightness of the time window set.

Our 6480 problems are generated on 6 grid network layouts of differing arc densities. On each of these layouts one problem is generated for every combination of the 6 demand sets, 5 bound sets, 3 problem lengths, 3 time window sets and 4 problem sizes.
4.2 Results From Our Problem Set

For this implementation five performance measures are used: Drive Time (the total time a vehicle is utilised, including travel time, loading and unloading); Route Number (the number of vehicles used); Split Deliveries (the number of customers whose demand is split); Waiting Time (the time a vehicle spends at a customer location while not in the process of making a delivery); Lag Time (the remaining time in the day which is neither drive time nor waiting time).

We analysed the effect of the improvement heuristics on our initial construction heuristic solutions, as summarised below:

- Drive time is reduced in 74.8% of the problems by an average of 10.0%.
- Fewer routes are required in 96.6% of the problems by an average of 34.0%.
- Split deliveries are reduced, on average, from 17.8% to 12.9% of all customers.
- Lag time is reduced by an average of 55.2%.
- Waiting time increased on average from 3.2 to 4.4 minutes per customer.

The following trends are evident in the solutions to our problems:

- The percentage of split deliveries increases as the number of customers is increased.
- The average waiting time per customer decreases as the number of customers is increased.
- The percentage of split deliveries increases as demand is increased as a proportion of vehicle capacity.
- The reduction in the number of split deliveries, due to the improvement heuristics, increases as demand:capacity increases.
- The average waiting time per customer decreases as average customer demand is increased. The reduction in waiting time also decreases as the demand:capacity increases.
- The percentage of problems with a reduction in drive time, and the average percentage drive time reduction, decreases as the average customer demand is increased.
- The percentage of split deliveries decreases as the bound set is made more restrictive.
- The improved results show a greater percentage of split deliveries with low bounds, as opposed to no bounds (this could be due to the low bound imposing a requirement that no split delivery can be made unless it is of a reasonable size; such split deliveries seem more likely to be advantageous).
- Tighter time windows restrict any improvements and increase waiting times.
These results provide compelling evidence that our construction heuristic is more effective on larger (as a proportion of vehicle capacity) demand sets, as is expected, due to greater flexibility derived from splitting larger demand. This also agrees with the conclusions of Dror and Trudeau [4, 5] for the standard Split Delivery Vehicle Routing Problem.

We also considered the effect of the return to depot checks. With return to depot checks, the drive time is increased by an average of 3.2%, whilst requiring an average of 8.7% fewer routes. The average percentage of split deliveries is reduced from 17.0% to 16.2% and the lag time is reduced by an average of 3.1%. The waiting time is reduced, on average, from 4.4 to 2.4 minutes per customer. These results show that the return to depot checks can produce considerable improvements in vehicle usage, percentage of split deliveries, lag time and waiting time at the cost of an increase in the drive time.

4.3 Results From Standard Test Problems

Our heuristics are implemented on problems generated by Solomon [10]. Full details of these standard problems are available from the authors. The test problems differ from our model in the following areas:

- The problems are not split into days.
- As there is no reloading of vehicles, the return to depot checks are removed.
- There are no multiple time windows.
- Euclidean distances are used (as opposed to grid network distances).
- Delivery times are not a function of the quantity being delivered.

As these problems vary greatly from our problem set, we did not expect exceptional results. However, if reasonable results were to be achieved it would bode well for the performance of our heuristics on the problems for which they were specifically developed.

The problem sets used are Rl, Cl, RC1 which correspond to random, clustered and semi-clustered problems, respectively. Some, or all, of these problems are solved by Solomon [10], Baker and Schaffer [12] and Koskosidis et al [11]. Table 1 gives a sample of these results i.e., for the problem set R1.

As expected, our heuristics did not, in general, perform quite as well as the other heuristics. Lower routing costs than those of Solomon were reported for 13 of the 21 problems, with higher routing costs being required for 7 problems, with 1 tie. However, of those problems with a lower routing cost, seven required 1 more vehicle and two required 2 extra vehicles (R102 required 1 less vehicle). In comparison with the more complex heuristics of Koskosidis et al and Baker and Schaffer an improved solution is achieved for only one problem, R102. Our solutions require an average of 0.95 extra vehicles per problem, with a standard deviation of 0.97. The required routing cost is increased by an average of 5.2%, with a standard deviation of 5.2. For both the random and semi-clustered problems the CPU time for our implementation seems considerably less than that required by Koskosidis et al or Baker and Schaffer, but consistently larger than the time required by Solomon. Our heuristic appears to take considerably longer than both Solomon and Koskosidis et al for the clustered problems, but is still much quicker than Baker and Schaffer; none of these times have been standardized, however.
### Table 1: Solution Results for Problem Set R1

<table>
<thead>
<tr>
<th>Problem</th>
<th>Statistic</th>
<th>Solomon</th>
<th>Baker and Schaffer</th>
<th>Koskosidis</th>
<th>Frizzell and Giffin</th>
</tr>
</thead>
<tbody>
<tr>
<td>R101</td>
<td>routing cost</td>
<td>1831</td>
<td>1672</td>
<td>1856</td>
<td>1720</td>
</tr>
<tr>
<td></td>
<td>vehicles used</td>
<td>21</td>
<td>20</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>CPU time (seconds)</td>
<td>2.7</td>
<td>156.6</td>
<td>299.1</td>
<td>70.4</td>
</tr>
<tr>
<td>R102</td>
<td>routing cost</td>
<td>1705</td>
<td>1531</td>
<td>1628</td>
<td>1575</td>
</tr>
<tr>
<td></td>
<td>vehicles used</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>CPU time (seconds)</td>
<td>2.8</td>
<td>257.7</td>
<td>325.6</td>
<td>144.2</td>
</tr>
<tr>
<td>R103</td>
<td>routing cost</td>
<td>1484</td>
<td>1414</td>
<td>1428</td>
<td>1335</td>
</tr>
<tr>
<td></td>
<td>vehicles used</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>CPU time (seconds)</td>
<td>3.0</td>
<td>76.4</td>
<td>369.3</td>
<td>184.6</td>
</tr>
<tr>
<td>R104</td>
<td>routing cost</td>
<td>1188</td>
<td>1119</td>
<td>1047</td>
<td>1154</td>
</tr>
<tr>
<td></td>
<td>vehicles used</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>CPU time (seconds)</td>
<td>3.4</td>
<td>304.5</td>
<td>856.4</td>
<td>97.9</td>
</tr>
<tr>
<td>R105</td>
<td>routing cost</td>
<td>1137</td>
<td>1036</td>
<td>975</td>
<td>1098</td>
</tr>
<tr>
<td></td>
<td>vehicles used</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>CPU time (seconds)</td>
<td>3.5</td>
<td>514.4</td>
<td>872.5</td>
<td>111.8</td>
</tr>
<tr>
<td>R109</td>
<td>routing cost</td>
<td>1355</td>
<td>1258</td>
<td>1244*</td>
<td>1289</td>
</tr>
<tr>
<td></td>
<td>vehicles used</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>CPU time (seconds)</td>
<td>2.9</td>
<td>343.6</td>
<td>68.56</td>
<td>81.5</td>
</tr>
</tbody>
</table>

5 Conclusions

We have developed and implemented three heuristics for the SDVSPTW. To our knowledge, this problem has not been previously considered in the literature, making absolute comparisons difficult. However, we have adapted our heuristics into a form suitable for implementation on some standard problems for the VRPTW. Our trimmed down heuristics perform sufficiently well for us to confidently expect good results on problems for which the heuristics were specifically developed.

Implementation on our own data sets reveals several interesting trends concerning the performance of our heuristics, as described in Section 4.2. Through the use of several special features we have encapsulated reality in our problem sets, without the need for actual data collection. We are presently investigating the possibility of extending our SDVSPTW model to incorporate the use of multiple depots, an area particularly suited to split deliveries, i.e., we need to consider deliveries split between different depots as well as considering deliveries split between different vehicles.

\*Some of the time windows are violated for this solution
References


