The Service-Time Restricted Capacitated Arc Routing Problem

Lise Lystlund
Aarhus University
Aarhus, Denmark

Sanne Wøhlk
CORAL - Centre of OR Applications in Logistics,
Aarhus School of Business, Aarhus University,
Aarhus, Denmark
sanw@asb.dk

Abstract

The Capacitated Arc Routing Problem (CARP) is the problem of servicing a set of edges with demand using a fleet of capacitated vehicles, such that each edge is serviced by exactly one vehicle, the vehicle capacities are respected, and the total routing cost is minimized.

We consider a variation of the problem with the additional constraint that any deadheading of an edge must take place at a later time than the time of service of the edge. We denote this problem as the Service-Time Restricted Capacitated Arc Routing Problem (STR-CARP). This problem has practical applications in snow removal.

We discuss the difficulties of modeling the problem and suggest a heuristic solution procedure.

1 Introduction

The Capacitated Arc Routing Problem (CARP) is the problem of servicing a set of streets with demand using a fleet of capacitated vehicles. In this paper we consider a variation of the problem where the streets must be serviced during the first traversal, i.e. deadheading is only allowed after the street has been serviced. We refer to this problem as the Service-Time Restricted Capacitated Arc Routing Problem (STR-CARP).

The problem is motivated by applications in snow removal, where such additional constraints can be advantageous.

Formally, the STR-CARP is stated as follows: given a connected undirected graph $G = (V, E, C, T, \tilde{T}, Q)$, where $V$ is the set of nodes, $E$ is the set of edges, $C$ is a cost matrix, $T (\tilde{T})$ is a matrix of travel times (service times), and $Q$ is a demand matrix, and given a number of identical vehicles each with capacity $W$, find a number of tours such that 1) each edge with positive demand is serviced by exactly one vehicle, 2) the sum of demands of those edges serviced by each vehicle does not exceed $W$, 3) no deadheading of an edge with positive demand starts before the service of that edge is completed, and 4) the total cost of the tours is minimized.
The classical CARP was first suggested by Golden and Wong in 1981 [7] and has since been studied in numerous papers. See for instance Assad and Golden [1], Dror [5], or Wolhk [11] for a survey. To the best of our knowledge, this is the first paper to address the STR-CARP. Because the STR-CARP is an extension of the classical CARP, the problem is \( \mathcal{NP} \)-hard since the CARP is so.

The remainder of the paper is organized as follows. In Section 2 we present a mathematical model for the STR-CARP. In Section 3 we suggest a construction heuristics for solving the problem and in Section 4 we present our computational results. Finally, in Section 5 we give some concluding remarks.

## 2 Mathematical Model

Let node 0 be a special depot node. For each edge \((i, j)\) in \(G\) we let \(c_{ij}\) denote the cost of traversing the edge. We let \(q_{ij} \geq 0\) be the demand of \((i, j)\). If \(q_{ij} > 0\), the edge is said to be required. We denote by \(E_D \subseteq E\) the set of required edges.

We use \(\tilde{t}_{ij}\) to denote the time that it takes to service the edge \((i, j)\), including the traversal, and let \(t_{ij}\) denote the deadheading time. Let \(K\) be a set of identical vehicles each with a capacity of \(W\). The vehicles start in the depot node at time zero.

In the model we use a set \(S = \{1, 2, \ldots\}\), indexed by \(s\), to denote the number of steps performed by a vehicle. Each step will correspond to the traversal of a single edge. If a vehicle traverses edge \((i, j)\) in the direction from node \(i\) to node \(j\) in step \(s\), it will arrive in node \(j\) at step \(s + 1\) and can traverse an edge from \(j\) to \(k\) in step \(s + 1\). If node \(j\) is the depot, the vehicle can stay there.

In the model we use five types of variables defined as follows:

\[
\begin{align*}
    x_{ij}^{ks} &= \begin{cases} 
    1 & \text{if vehicle } k \text{ traverses } (i, j) \text{ from } i \text{ to } j \text{ in step } s \\
    0 & \text{otherwise}
    \end{cases} \\
    y_{ij}^{ks} &= \begin{cases} 
    1 & \text{if vehicle } k \text{ services } (i, j) \text{ from } i \text{ to } j \text{ in step } s \\
    0 & \text{otherwise}
    \end{cases}
\end{align*}
\]

The next two variables are related to the time for each vehicle as follows. We use \(\nu_{ij}^{ks}\) to denote the time at which vehicle \(k\) starts traversing edge \((i, j)\) in direction from \(i\) to \(j\) at step \(s\), if it does so. If vehicle \(k\) does not perform this traversal at this step, the variable is free to take any value. Similarly, we use \(\mu_{i}^{ks}\) to denote the time at which vehicle \(k\) arrives in node \(i\) at step \(s\), if it does so. If vehicle \(k\) does not arrive in \(i\) at step \(s\), the variable is free.

Finally, for required edges \((i, j) \in E_D\), we use \(\omega_{ij}\) to denote the time at which service of the edge starts. Note that this variable is independent of vehicle index and hence there is only one such variable per required edge.

The most commonly used objective function in arc routing problems, and in routing problems in general, is to minimize the total travel cost, which results in the objective function given in (1). This is the objective function used in our model.
\[
\sum_{k \in K} \sum_{s \in S} \sum_{(i,j) \in E} c_{ij} x_{ij}^{ks}
\]  

(1)

Alternative objective functions include minimization of the time that it takes for the last vehicle to finish. This is known as minimizing the makespan and results in the objective function stated in (2).

\[
\max_{k \in K, s \in S} \mu_{0}^{ks}
\]  

(2)

Minimizing the sum of the working time for the vehicles, defined as the time from start to the time when the vehicles return to the depot at the end of their routes, is another obvious choice of objective. This objective function is given in (3).

\[
\sum_{k \in K} \left( \max_{s \in S} \mu_{0}^{ks} \right)
\]  

(3)

Note here that we must take into account that a vehicle may pass through the depot node at any time during its route prior to actually returning to the depot. Hence, for each vehicle we must take the last time at which the vehicle enters the depot.

Finally, it may be useful to minimize the waiting time in the system, i.e. the time the vehicles are waiting in nodes. This objective function is stated in (4). We assume here that waiting at the first step, i.e. before the vehicles leave the depot the first time, is not included.

\[
\sum_{k \in K} \sum_{s \in S \setminus \{1\}} \sum_{(i,j) \in E} \left( \nu_{ij}^{ks} - \mu_{i}^{ks} \right)
\]  

(4)

Note that using (4) alone will result in unnecessary deadheading in order to avoid waiting. Therefore, this objective function is most likely to be used in combination with other objectives such as (1).

The constraint set of the STR-CARP can be formulated as follows, where we assume that waiting is allowed for any vehicle in any node.
\[ \sum_{s \in S} \sum_{(i,j) \in E} q_{ij} x_{ij}^{ks} \leq W \quad \forall k \in K \]  
\[ \sum_{k \in K} \sum_{s \in S} (y_{ij}^{ks} + y_{ji}^{ks}) = 1 \quad \forall (i,j) \in E_D \]  
\[ y_{ij}^{ks} \leq x_{ij}^{ks} \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \sum_{(i,j) \in E} x_{ij}^{ks} \leq 1 \quad \forall k \in K, \forall s \in S \]  
\[ \sum_{j \in V} x_{ij}^{ks} - \sum_{j \in V} x_{ji}^{ks+1} = 0 \quad \forall k \in K, \forall s \in S, \forall i \in V \setminus \{0\} \]  
\[ \sum_{s \in S} \sum_{j \in V} x_{ij}^{ks} - \sum_{j \in V} x_{0j}^{ks} = 0 \quad \forall k \in K \]  
\[ \mu_{ij}^{ks} - \nu_{ij}^{ks} \leq M(1 - x_{ij}^{ks}) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \nu_{ij}^{ks} + t_{ij} - \mu_{ij}^{ks+1} \leq M(1 - y_{ij}^{ks}) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \mu_{ij}^{ks+1} - (\nu_{ij}^{ks} + t_{ij}) \leq M(1 - y_{ij}^{ks}) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \nu_{ij}^{ks} + t_{ij} - \mu_{ij}^{ks+1} \leq M(1 - (x_{ij}^{ks} - y_{ij}^{ks})) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \mu_{ij}^{ks+1} - (\nu_{ij}^{ks} + t_{ij}) \leq M(1 - (x_{ij}^{ks} - y_{ij}^{ks})) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \nu_{ij}^{ks} - \omega_{ij} \leq M(1 - y_{ij}^{ks}) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \omega_{ij} + t_{ij} - \nu_{ij}^{ks} \leq M(1 - (x_{ij}^{ks} - y_{ij}^{ks})) \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \mu_{0j}^{k1} = 0 \quad \forall k \in K \]  
\[ x_{ij}^{ks}, y_{ij}^{ks} \in \{0,1\} \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \mu_{ij}^{ks} \geq 0 \quad \forall k \in K, \forall s \in S, \forall i \in V \]  
\[ \nu_{ij}^{ks} \geq 0 \quad \forall k \in K, \forall s \in S, \forall (i,j) \in E \]  
\[ \omega_{ij} \geq 0 \quad \forall (i,j) \in E \]  

Here, constraint (5) is the capacity restriction of the single vehicle and (6) ensures that all required edges are serviced by exactly one vehicle. Constraint (7) ensures that if a vehicle services a given edge in a given direction at a given step, then that same vehicle traverses the edge at the same step in the same direction and (8) ensures that each vehicle performs at most one task at each step. (9) is the continuity constraint and (10) ensures that the number of vehicles leaving the depot in the first step equals the number of vehicles entering the depot in subsequent steps.

Constraint (11) ensures that if a vehicle traverses edge \((i,j)\) from \(i\) to \(j\) in step \(s\), then the start time of the traversal cannot be before the arrival time of the vehicle in node \(i\) at step \(s\). Constraints (12) and (13) together ensure that if a vehicle services edge \((i,j)\) from \(i\) to \(j\) at step \(s\), then the arrival time in node \(j\) equals the start time of the service plus the service time. These constraints are non-restrictive if the vehicle does not service this edge in the given direction at this step. Constraints (14) and (15) together ensure that if a vehicle deadheads edge \((i,j)\) from \(i\) to \(j\) at step \(s\), then the arrival time in node \(j\) equals the start time of the
traversal plus the travel time. These constraints are non-restrictive if the vehicle does not traverse this edge in the given direction at this step and if the vehicle services the edge.

In constraint (16) we fix the service time variable \( \omega_{ij} \) to equal the time at which the servicing vehicle starts its service of the edge. Constraint (17) ensures that any deadheading (traversing but not servicing) of a required edge \((i, j)\) starts no earlier than the finish time of the service of this edge. Note that the constraint is not binding for vehicles servicing the edge. Constraint (18) ensures that the vehicles start in the depot at the first step at time zero. Constraints (19) through (22) ensure that traversal and service variables are binary and that all time variables are non-negative.

If a model is needed where waiting time is not allowed for the vehicles, constraint (23) should be added to the above model. Together with constraint (11) this prohibits waiting when leaving a node. Constraints (12) and (13) and constraints (14) and (15) prohibit waiting when arriving at a node after servicing and deadheading an edge, respectively.

\[
\nu_{ij}^{ks} - \mu_{i}^{ks} \leq M(1 - x_{ij}^{ks}) \quad \forall k \in K, \forall s \in S \setminus \{1\}, \forall (i, j) \in E
\]  

Note that with (23) added, waiting is still allowed at the first step. Hence, vehicles are allowed to wait in the depot before starting the tour. This can be necessary to ensure feasibility and is a reasonable assumption. Note also that if waiting is prohibited in all nodes except the depot, there is a risk that vehicles will have to deadhead unnecessarily while waiting to traverse a specific edge. In any implementation, the allowance or prohibition of waiting should be carefully considered. Changing the objective function to include minimization of waiting time may be a better choice than forbidding waiting.

## 3 Heuristic for STR-CARP

The heuristic for the STR-CARP is inspired by the Path-Scanning procedure for CARP [2]. Starting with an empty solution, the procedure builds tours by expanding the partial tours with edges to be serviced. At each iteration, the graph is scanned for unserviced demand edges close to the depot or close to the end node of a partial route. When the set of the nearest unserviced demand edges is found, the edges are evaluated with regard to expansion of the solution. The edge that is evaluated to be the most significant for the further expansion of the solution is to be inserted into the solution. If the edge is closest to the depot, a new route is established. If instead the edge is closest to an end node of an established partial route, the edge is added to this route.

A graph \( G' = (V, E') \) is constructed to separate the demand edges that should be serviced later from the edges that have been serviced or have no demand. The graph \( G' \) consists of the same node set as \( G \) and the edge set is the demand edges that have been serviced and the edges with no demand. Thus, at each iteration, \( G' \) contains the edges where deadheading is allowed.

We estimate the distance between a node, \( v \), and an edge, \( e = (i, j) \), in \( G \) as:
\[ l_{v,e} = \min\{SPL(v,i), SPL(v,j)\}, \]

where \( SPL(v,i) \) denotes the length of a shortest path between node \( v \) and \( i \) in the network \( G \). Similarly, we estimate the distance between a node, \( v \), and an edge, \( e = (i,j) \), in \( G' \) as:

\[ l'_{v,e} = \min\{SPL'(v,i), SPL'(v,j)\}, \]

where \( SPL'(v,i) \) denotes the length of a shortest path between node \( v \) and \( i \) in the network \( G' \). In \( l'_{v,e} \) we use another estimate if the edge \( e \) is not reachable from \( v \). This is explained in detail below.

### 3.1 Evaluation of the edges

To evaluate the importance of the edges, we introduce a parameter called Adjusted Sum, which is determined for each edge in \( G \). Adjusted Sum for an edge is calculated by removing the edge from \( G \) and then summing the distance from the depot to every edge left in \( G \). If the resulting graph is unconnected after the edge has been removed, the distance from the depot and an unreachable edge is set equal to the longest distance between the depot and an edge in the original graph. Put more precisely:

\[ l_{\max} = \max_{e \in E} \{l_0,e\} \]

Normally, the distance between two nodes in different components is set to infinite. But to make sure that the Adjusted Sum is consistent with the importance of the edge, it is essential that the Adjusted Sum is not infinite only because one edge is isolated from the depot. This is due to the fact that we want the Adjusted Sum to indicate the number of edges being isolated from the depot when an edge is being removed, which is not possible if the length of the shortest path between the depot and an edge in a different component is equal to infinity.

The Adjusted Sum for each \( e' \in E_D \) can be calculated using the procedure below:

1. Let \( G_{e'} = (V, E_{e'}) \), where \( E_{e'} = E \setminus \{e'\} \).
2. Calculate the length of the shortest path in \( G_{e'} \) from the depot to every edge as
   \[ l_0,e = \begin{cases} 
   \min\{SPL(0,i), SPL(0,j)\} & \text{if } e \text{ is in the same component as the depot} \\
   l_{\max} & \text{else} 
   \end{cases} \]
3. Calculate the Adjusted Sum as
   \[ AdjSum(e') = \sum_{e \in E_{e'}} l_0,e \]

The Adjusted Sum indicates the importance of the edge, \( e' \), with respect to further expansion of the solution. The larger the parameter is the more detours are necessary to reach the rest of the network without using the edge \( e' \).
3.2 Construction of a STR-CARP solution

When a route is established, an index, \( k \), is assigned to the route. We use \( \tau(k) \) to denote the current end node of the partial route \( k \). Initially, the end node is the depot. Each time an edge is added to the route, the end node is updated. We use \( D(k) \) to denote the total demand assigned to route \( k \) and \( C(k) \) to denote the total cost of the route. We use \( K \) to denote the set of routes under consideration. Note that \( K \) will always contain an empty route.

Let \( V_\tau = \bigcup_{k \in K} \tau(k) \), i.e. the set of nodes where at least one partial route ends. The edge set \( E_{\min}' \) is the set of edges closest to the depot or to an established route and is defined as follows:

Let \( l_{\min} = \min_{k \in K} \{ \min_{(i,j) \in E \setminus E'} q_{ij} + D(k) \leq W \{ l'_{\tau(k),(i,j)} \} \} \), then we can define \( E_{\min}' \) as:

\[
E_{\min}' = \{(i,j) \in E \setminus E' \mid \exists k \in K: q_{ij} + D(k) \leq W \text{ and } l'_{\tau(k),(i,j)} = l_{\min}\}
\]

The heuristic for the STR-CARP is given below:

1. Initialization
   (a) \( E' = \{(i,j) \in E : q_{ij} = 0\} \), thus \( G' = (V, E') \) is initially the graph consisting of every node and every edge with no demand from \( G \).
   (b) Create an empty route indexed by \( k = 1 \).
2. Scan the graph for the unserviced demand edges closest to the node set \( V_\tau' \) as follows.
   (a) Calculate \( l_{\min} \)
   (b) Determine \( E_{\min}' \)
3. Select an edge, \( e \in E_{\min}' \), that maximizes the Adjusted Sum:

\[
e^* = \arg \max_{e \in E_{\min}'} \{ AdjSum(e) \}
\]
4. Determine the route, \( k^* \), that \( e^* = (i^*, j^*) \) should be added to, such that

\[
q_{i^*j^*} + D(k^*) \leq W \text{ and } l'_{\tau(k^*),(i^*,j^*)} = l_{\min}
\]

If \( v^* \) is the depot, a new empty route is established.
5. Update route \( k^* \) as follows:
   (a) Add the edges of the shortest path between \( v^* \) and \( e^* \) to the route.
   (b) Add \( e^* \) to the route.
   (c) Set \( D(k^*) := D(k^*) + q_{i^*j^*} \)
   (d) Set \( C(k^*) := C(k^*) + l_{v^*,e^*} + c_{i^*j^*} \)
   (e) Set \( \tau(k^*) := j^* \) (or \( i^* \), depending on the direction of \( e^* \))
6. Update the graph as follows:
E' := E' ∪ \{e^*\}

7. If E \ E' = ∅ (every demand edge has been added to a route), proceed to step 8, else return to step 2.

8. For every route, find the shortest path from \(\tau(k)\) to the depot and add the path to the associated route. Update the cost by setting

\[C(k) := C(k) + SPL'(\tau(k), 0)\]

The heuristic does not take the service time and deadheading time of the edges into account when the solution is being expanded. Therefore, it may be necessary for some vehicles to wait in nodes in the final solution. Even though this can happen, it may not occur in every solution found by the heuristic. The edges are added iteratively, which can imply that the waiting time will be minimized automatically.

If the heuristic were required to take into account that waiting time is not allowed, it would be necessary to introduce variables that indicate the service time and deadheading time for each edge. But to implement this in the heuristic requires a comparison between deadheading time and service time each time a vehicle wants to deadhead an edge. This leads to a large number of comparisons each time an edge is inserted into the solution, which complicates the procedure severely.

4 Computational Results

We have tested the algorithm on three data sets for the classical CARP: the A instances presented in [10], the GDB instances [4], and the KSHS instances [8]. The data is available at [3] and [9]. The algorithm has been coded in C++ using the LEDA library [6].

Table 1 shows the results obtained by the heuristic described in Section 3. The lower bounds given in the table are the best known lower bounds for the classical CARP problem. Here, an asterisk indicates that the lower bound value corresponds to the optimal objective value for the classical CARP.

Note that even when the lower bound corresponds to the optimal CARP value, the size of the gap will originate from two sources: 1) the quality of the proposed heuristic (there is likely to be a gap between the heuristic solution and the optimal solution of the STR-CARP), 2) the gap between the optimal value of the STR-CARP and the optimal value of the classical CARP. This gap is expected to be significant as the STR-CARP is very restrictive compared to the classical CARP.

5 Concluding Remarks

The STR-CARP requires a significantly more complex model than the corresponding classical CARP. The service time restriction means that time stamps are required in nodes and on edges. This is also the case for the CARP with time windows (CARP-TW), but here the model can be built on a node duplicated extension of the network [10]. As a result, it is
<table>
<thead>
<tr>
<th>Data set</th>
<th>Lower bound</th>
<th>Heu</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>A10A</td>
<td>80</td>
<td>107</td>
<td>33.8</td>
</tr>
<tr>
<td>A10B</td>
<td>66</td>
<td>92</td>
<td>39.4</td>
</tr>
<tr>
<td>A10C</td>
<td>*56</td>
<td>75</td>
<td>33.9</td>
</tr>
<tr>
<td>A10D</td>
<td>*50</td>
<td>75</td>
<td>50.0</td>
</tr>
<tr>
<td>A13A</td>
<td>159</td>
<td>241</td>
<td>51.6</td>
</tr>
<tr>
<td>A13B</td>
<td>112</td>
<td>171</td>
<td>52.7</td>
</tr>
<tr>
<td>A13C</td>
<td>*100</td>
<td>126</td>
<td>26.0</td>
</tr>
<tr>
<td>A13D</td>
<td>*100</td>
<td>118</td>
<td>28.0</td>
</tr>
<tr>
<td>A15A</td>
<td>146</td>
<td>212</td>
<td>45.2</td>
</tr>
<tr>
<td>A15B</td>
<td>117</td>
<td>199</td>
<td>21.3</td>
</tr>
<tr>
<td>A15C</td>
<td>105</td>
<td>138</td>
<td>31.4</td>
</tr>
<tr>
<td>A15D</td>
<td>*102</td>
<td>138</td>
<td>35.3</td>
</tr>
<tr>
<td>A20A</td>
<td>214</td>
<td>278</td>
<td>29.9</td>
</tr>
<tr>
<td>A20B</td>
<td>164</td>
<td>199</td>
<td>21.3</td>
</tr>
<tr>
<td>A20C</td>
<td>*142</td>
<td>172</td>
<td>21.2</td>
</tr>
<tr>
<td>A20D</td>
<td>*138</td>
<td>172</td>
<td>24.6</td>
</tr>
<tr>
<td>A24A</td>
<td>242</td>
<td>331</td>
<td>36.8</td>
</tr>
<tr>
<td>A24B</td>
<td>199</td>
<td>279</td>
<td>40.2</td>
</tr>
<tr>
<td>A24C</td>
<td>176</td>
<td>219</td>
<td>24.4</td>
</tr>
<tr>
<td>A24D</td>
<td>170</td>
<td>219</td>
<td>28.8</td>
</tr>
<tr>
<td>A27A</td>
<td>314</td>
<td>483</td>
<td>53.8</td>
</tr>
<tr>
<td>A27B</td>
<td>266</td>
<td>374</td>
<td>40.6</td>
</tr>
<tr>
<td>A27C</td>
<td>241</td>
<td>309</td>
<td>28.2</td>
</tr>
<tr>
<td>A27D</td>
<td>235</td>
<td>309</td>
<td>31.5</td>
</tr>
<tr>
<td>A31A</td>
<td>540</td>
<td>860</td>
<td>59.3</td>
</tr>
<tr>
<td>A31B</td>
<td>402</td>
<td>609</td>
<td>51.5</td>
</tr>
<tr>
<td>A31C</td>
<td>344</td>
<td>482</td>
<td>40.1</td>
</tr>
<tr>
<td>A31D</td>
<td>328</td>
<td>426</td>
<td>29.9</td>
</tr>
<tr>
<td>A40A</td>
<td>741</td>
<td>1178</td>
<td>59.0</td>
</tr>
<tr>
<td>A40B</td>
<td>514</td>
<td>751</td>
<td>46.1</td>
</tr>
<tr>
<td>A40C</td>
<td>426</td>
<td>575</td>
<td>35.0</td>
</tr>
<tr>
<td>A40D</td>
<td>410</td>
<td>474</td>
<td>15.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data set</th>
<th>Lower bound</th>
<th>Heu</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gdb1</td>
<td>*316</td>
<td>337</td>
<td>6.7</td>
</tr>
<tr>
<td>Gdb2</td>
<td>*339</td>
<td>424</td>
<td>25.1</td>
</tr>
<tr>
<td>Gdb3</td>
<td>*275</td>
<td>332</td>
<td>20.7</td>
</tr>
<tr>
<td>Gdb4</td>
<td>*287</td>
<td>329</td>
<td>14.6</td>
</tr>
<tr>
<td>Gdb5</td>
<td>*377</td>
<td>393</td>
<td>4.2</td>
</tr>
<tr>
<td>Gdb6</td>
<td>*298</td>
<td>370</td>
<td>24.3</td>
</tr>
<tr>
<td>Gdb7</td>
<td>*325</td>
<td>398</td>
<td>22.5</td>
</tr>
<tr>
<td>Gdb8</td>
<td>344</td>
<td>416</td>
<td>20.9</td>
</tr>
<tr>
<td>Gdb9</td>
<td>*303</td>
<td>304</td>
<td>0.0</td>
</tr>
<tr>
<td>Gdb10</td>
<td>*275</td>
<td>338</td>
<td>22.9</td>
</tr>
<tr>
<td>Gdb11</td>
<td>*395</td>
<td>481</td>
<td>21.8</td>
</tr>
<tr>
<td>Gdb12</td>
<td>450</td>
<td>571</td>
<td>26.9</td>
</tr>
<tr>
<td>Gdb13</td>
<td>536</td>
<td>648</td>
<td>20.9</td>
</tr>
<tr>
<td>Gdb14</td>
<td>*100</td>
<td>113</td>
<td>13.0</td>
</tr>
<tr>
<td>Gdb15</td>
<td>*58</td>
<td>68</td>
<td>17.2</td>
</tr>
<tr>
<td>Gdb16</td>
<td>*127</td>
<td>145</td>
<td>14.2</td>
</tr>
<tr>
<td>Gdb17</td>
<td>*91</td>
<td>101</td>
<td>11.0</td>
</tr>
<tr>
<td>Gdb18</td>
<td>*164</td>
<td>201</td>
<td>22.6</td>
</tr>
<tr>
<td>Gdb19</td>
<td>*55</td>
<td>63</td>
<td>14.6</td>
</tr>
<tr>
<td>Gdb20</td>
<td>*121</td>
<td>137</td>
<td>13.2</td>
</tr>
<tr>
<td>Gdb21</td>
<td>*156</td>
<td>181</td>
<td>16.0</td>
</tr>
<tr>
<td>Gdb22</td>
<td>*200</td>
<td>211</td>
<td>5.5</td>
</tr>
<tr>
<td>Gdb23</td>
<td>*233</td>
<td>256</td>
<td>9.9</td>
</tr>
<tr>
<td>KSCHS1</td>
<td>*14661</td>
<td>17155</td>
<td>17.0</td>
</tr>
<tr>
<td>KSCHS2</td>
<td>*9863</td>
<td>10093</td>
<td>2.3</td>
</tr>
<tr>
<td>KSCHS3</td>
<td>*9320</td>
<td>10216</td>
<td>9.6</td>
</tr>
<tr>
<td>KSCHS4</td>
<td>11098</td>
<td>13236</td>
<td>19.2</td>
</tr>
<tr>
<td>KSCHS5</td>
<td>*11903</td>
<td>12821</td>
<td>7.7</td>
</tr>
<tr>
<td>KSCHS6</td>
<td>*10197</td>
<td>13921</td>
<td>36.5</td>
</tr>
</tbody>
</table>

Table 1: Results obtained by heuristic.

possible to limit the number of traversals of each edge and the number of visits in each node to one and hence only a single time variable is needed for each node and for each edge. The service time restriction means that we cannot limit the number of visits in a single node in the same way and indeed we cannot guarantee that shortest paths are chosen for traversal. In this paper we have suggested a model for the STR-CARP where an additional dimension is added to the variable in order to keep track of the activity steps performed by the system.

A construction heuristic is suggested for the STR-CARP. In this algorithm we repeatedly add an edge to the solution. The added edge is chosen among those that can be reached from the current situation while respecting the service time restrictions. Among those edges, we choose one that is relatively close to the end of the current partial solutions and is relatively
important. Here, the importance of an edge is determined by measuring to which extent it harms the system not to have the edge included. In the STR-CARP vehicles depend on important edges being added to the solution at an early stage since only then can other vehicles use these edges for deadheading. This is the major difference between the STR-CARP and the classical CARP, where it is enough to focus on the routing of each vehicle separately.

Two interesting directions of further research in relation to the STR-CARP are 1) to extend the solution procedure in the directions of Meta Heuristics and 2) to make a more thorough study of the different objectives and their consequences for the solution.

References


