Construction Heuristics for the Semi-Periodic Capacitated Arc Routing Problem

Lone Kiilerich*[1]

[1]Cluster for Operations Research And Logistics, Department of Economics and Business Economics, Aarhus University, Denmark

August 15, 2018

Abstract

Keywords: SP-CARP, Arc Routing Construction Heuristic, Randomized, Path Scanning

1 Introduction

When doing arc routing, edges are traditionally treated as if they consist of one unit or customer, but edges can be road segments with several customers, and thus possibly several individual customers. An example is within waste collection where the demand of an edge is the joint demand of the households on a street segment. This can imply that the service schedules of those households vary. Consider, for example, a street segment where three customers request collection once every two weeks (A), two customers every week (B), and one customer twice a week (C), and assume for sake of simplicity that the demand of each customer is 1 unit. Then the demand of the edge will be 1 unit two times a week plus two additional units once a week, plus three additional units every second week. This is the problem investigated in this paper. The purpose of this paper is to present two constructive heuristics: The Edge Order heuristic and the Master-Route Heurisitc for this real world problem, which is denoted the Semi-Periodic Capacitated Arc Routing Problem (SP-CARP) and was introduced in [5], where a detailed formulation of the problem can be found. Alternatively, the problem could be modeled as a periodic node routing problem. However, this would result in a significant increase in graph size as each household would have to be modeled as a separate node.

In the SP-CARP, we have a network $G = (\mathcal{N}, \mathcal{E})$ where a set of demand edges $\mathcal{E}_R$ are to be serviced by a fleet $\mathcal{K}$ of homogeneous vehicles. During a set of days $\mathcal{T}$ all demand must be collected at least one time. In order to specify the exact frequency, we define a set of service schedules $\mathcal{H}$. Each schedule $h \in \mathcal{H}$ specifies the number of service times that must occur in the final plan and the minimum a_h and maximum b_h number of days between two consecutive services. In the introduction example,

*Corresponding author: lone.ki.ch@econ.au.dk

we would have 3 schedules, one for customers (A), (B) and (C) respectively. For example, we will have $a_h = b_h = 14$ for customer (A). Each demand edge $e \in \mathcal{E}_R$ may thereby have demand belonging to several service schedules. The demand belonging to schedule $h \in \mathcal{H}$ is denoted q_e^h and is the demand collected at each service.

The problem consists of creating routes that collect all required demand in such a way that the maximum number of vehicles used on any day and the related routing costs are minimized.

In the SP-CARP described above, the demand to be collected remains the same regardless of the duration between two consecutive services of an edge. This assumption can be justified by the fact that the problem is inspired by waste collection where we have experienced a fixed duration between service times and thereby fixed demand size. However, a natural extension of this problem is to let the demand be time dependent as is the case in [3].

The remaining part of this section describes the notation used in this paper, and an overview of the notation is found in Table 1.

1.1 Day combinations

For each service schedule $h \in \mathcal{H}$, it is possible to find several sets of days that satisfy the service requirements given by the schedule h. For customer (B) in the introductory example the sets are: Every Monday, Tuesday, Wednesday, Thursday, or Friday in a planning horizon with a five day workweek. Let $D^h \subseteq \mathcal{T}$ be such a set of days, henceforward denoted a day combination. For each $h \in \mathcal{H}$, D^h is a valid day combination if it contains as many days as the number of times an edge must be serviced with respect to service schedule h, all pairs of days $t, t' \in D^h$ are at least a_h days apart, and for each $t \in D^h$ there exists a day $t' \in D^h \setminus \{t\}$ such that $|t - t'| \leq b_h$. Define $\mathcal{D}^h$ to be all such valid day combinations for service schedule h.

We define $h < h'$ for $h, h' \in \mathcal{H}$ if h is less frequent than h' and $h = h'$ if they are equally frequent. For service schedules $h < h'$ define $\mathcal{D}^h(D^{h'}) := \{D^h \in \mathcal{D}^h | D^h \subseteq D^{h'}\}$, that is, for a given day combination $D^{h'}$, the set $\mathcal{D}^h(D^{h'})$ contains all the valid day combinations for service schedule h that are contained in day combination $D^{h'}$. E.g. let h' be a schedule that requires service twice a week and h a schedule that requires service every week. In a planning horizon of two weeks $D^{h'} = \{1, 3, 8, 11\}$ is a valid day combination for h' and given this day combination we have; $\mathcal{D}^h(D^{h'}) = \{\{1, 8\}, \{3, 8\}\}$ as these are the only valid day combinations for h with all days contained in $D^{h'}$.

The remaining part of this paper is structured as follows. First, Sections 2 and 3 introduce two construction heuristics: The Edge Order heuristic and the Master-Route heuristic. Both heuristics contain several components and methods for each component. Depending on which method you choose for each component, you will get different characteristics for the solution found. Section 4 focuses on deciding which methods to use for each heuristic component. Finally, Section 5 shows the results from testing the algorithms presented.

Index Sets and Indices	
$\mathcal{N}$	The set of nodes, indexed by i and j.
$\mathcal{E}$	The set of edges, indexed by the ordered pair (i, j) with $i < j$.
$\mathcal{H}$	The set of service schedules, indexed by h.
$\mathcal{E}_R$	The set of required edges, i.e., $\mathcal{E}_R = \{(i, j) \in \mathcal{E} : \exists h \in \mathcal{H}, q_e^h > 0\}$.
$\mathcal{E}_R^h$	The set of required edges wrt. $h \in \mathcal{H}$, i.e., $\mathcal{E}_R^h = \{(i, j) \in \mathcal{E} : q_e^h > 0\}$.
$\mathcal{K}$	The set of vehicles, indexed by k.
W	Capacity of each vehicle.
$\mathcal{T}$	The set of days in the time horizon, indexed by t.
$\mathcal{T}_0$	The set of days on which service can not be performed.
$\mathcal{D}^h$	All valid day combinations for service schedule h, i.e., $\mathcal{D}^h = \{D^h \subseteq \mathcal{T} \setminus \mathcal{T}_0 \mid \exists\, t^* \in D^h : \lvert t - t^* \rvert \leq b_h, a_h < \lvert t - t' \rvert \; \forall t, t' \in D^h \; t \neq t'\}$.
$\mathcal{D}^h(D^{h'})$	The set of all valid day combinations contained in $D^{h'}$ for service schedule h , i.e., $\mathcal{D}^h(D^{h'}) := \{\delta^h \in \mathcal{D}^h \lvert \delta^h \subseteq D^{h'}\}$.
Parameters	
c_{ij}	Traversal cost of edge $(i, j) \in \mathcal{E}$, $c_{ij} > 0$.
a_h	Minimum number of days between two consecutive services of $h \in \mathcal{H}$ on the same edge.
b_h	Maximum number of days between two consecutive services of $h \in \mathcal{H}$ on the same edge.
q_e^h	Demand of edge $(i, j) \in \mathcal{E}_R$ with service schedule $h \in \mathcal{H}$.
Variables	
π	A permutation $\rho : e \in \mathcal{E} \to \mathbb{N}$.
Q^t	Total demand assigned to day $t \in \mathcal{T}$
Q_r^t	Demand on current route
R^l	Master-route number l
u^h	Number of edges added to routes since service schedule $h \in \mathcal{H}$ occurred.
D^h	A valid day combination.
$\tilde{D^h}$	The last chosen day combination for service schedule $h \in \mathcal{H}$.
$\hat{D^h}$	The preferred day combination for service schedule $h \in \mathcal{H}$.
r^t	The number of routes on day $t \in \mathcal{T}$.

Table 1: Overview of notation used.

2 The Edge Order heuristic

In this section we introduce a construction heuristic denoted the Edge Order heuristic. The overall procedure is to first choose a day combination for all demand schedules on each edge. Subsequently the edge is added to routes on the days in this day combination. The first part of this section considers three main aspects of this algorithm. Firstly, an outline of the algorithm describing how routes are constructed. Secondly, the order in which the edges are considered. Lastly, the criteria for choosing day combinations. The remaining two subsections discuss the characteristics of the algorithm based on which method is chosen for each component, and a post-optimization procedure that can be called after finding an initial solution.

In Section 4 we test the performance of different combinations and choose three combinations as the final versions of the Edge Order heuristic.

2.1 Basic procedure

In this subsection the overall idea of the Edge Order heuristic is described. First, the edges in $\mathcal{E}_R$ are ordered using a given permutation rule. Then an empty route only containing the depot is created for all days $t \in \mathcal{T} \setminus \mathcal{T}_0$ these routes are ready to service non serviced edges and are denoted active routes. When no more edges can be serviced by the current active route that route returns to depot and a new empty active route is created on that day.

Consider the edges $e \in \mathcal{E}_R$ in the order given by the permutation. Choose a day combination for each $h \in \mathcal{H}$ where $q_e^h > 0$ using the chosen day combination choice strategy. Having found a day combination for all service schedules, consider each schedule in turn starting with the most frequent. Add the edge to the end of the active route of each day in the chosen day combination. If the edge is already in the route (added due to another service schedule), only add the demand to the route. If the active route in a day t lacks capacity in order to service q_e^h demand, close the currently active route and create a new route and add the edge to this new route. Continue until all service schedules have been added to routes.

It may happen that an edge is serviced by more than one route on a given day. This could occur if we have several service schedules with positive demand on an edge. The first schedule is serviced by the currently active route. The second schedule exceeds the capacity of that route so that it is closed and a new route is created. The edge and demand of the second schedule are added to this newly created route. If the demand of the third schedule can be serviced by the (closed) route servicing schedule one, then the demand will be added to that route. So demand can be added to a closed route, if that route already services this edge. Sections 2.2 and 2.3 describe which edge permutations and day combination choice strategies are used in this algorithm.

2.2 Edge permutation

The order in which the edges are treated has an effect on the final solution for two reasons. Firstly, the edges are added to the end of the current route before continuing to the next edge. Secondly, one of the criteria used for choosing day combinations takes into consideration the day combinations that were chosen for the previous

edges. Five different rules of permutation $\pi : e \in \mathcal{E} \rightarrow \mathbb{N}$ were considered and are described next.

Random The first permutation rule is to make the order completely random. Set $\mathsf{E} = \{0, \cdots, |\mathcal{E}_R| - 1\}$. For each $e \in \mathcal{E}_R$, draw $i \in \mathsf{E}$ randomly and set $\pi(e) = i$ and $\mathsf{E} = \mathsf{E} \setminus \{i\}$. Continue until $\mathsf{E} = \emptyset$.

Depth first search The second and third rules work by using a depth first search (see [2]) on the graph. The only difference between the two is that the second rule uses the depot node as the root node and the third rule uses a random node as the root node. Set $i = 0$ and perform a depth first search from the chosen root node. Each time the depth first search traverses an edge $e \in \mathcal{E}_R$ not already visited by the algorithm, set $\pi(e) = i$ and $i = i + 1$.

Breadth first search The fourth and fifth rules are very similar to the two above except that a breadth first search (see [2]) is used to generate the order instead of a depth first search.

2.3 Choose day combination

Now we look into how to choose a day combination among the valid day combinations for all edges and each service schedule with positive demand on the individual edge. For most service schedules we have $|\mathcal{D}^h| > 1$, and in those cases we have to choose one of the day combinations $D^h \in \mathcal{D}^h$. The way these day combinations are selected has a direct influence on the final solution as the choice of day combination specifies the service days of an edge. This section describes three selection rules used to choose day combinations, one that selects a day combination at random, one that seeks to spread the total demand evenly on all days in the planning horizon and one that uses information on what has been chosen for the previously treated edges.

Random In this selection rule, the day combinations for an edge $e \in \mathcal{E}_R$ are chosen at random. We first consider the most frequent service schedule $h \in \mathcal{H}$ with positive demand on e. For this schedule we select a day combination D^h from the set $\mathcal{D}^h$ at random. For all other service schedules $h' \in \mathcal{H}$ with positive demand on e, the day combination $D^{h'}$ is chosen randomly among the day combinations in $\mathcal{D}^{h'}(D^h)$. When choosing one day combination from a set of day combinations all day combinations in the set are equally likely to be chosen.

Spread demand The second selection rule seeks to distribute the total demand evenly on all service days and thereby avoid having days that require many vehicles and other days requiring few or non vehicles. This is accomplished by introducing a variable to hold the demand assigned to day t, Q^t with initial value 0 for all $t \in \mathcal{T} \setminus \mathcal{T}_0$. Start again from the most frequent service schedule $h \in \mathcal{H}$ with positive demand on the edge. Choose the day combination for h that satisfies:

$$\min_{D^h \in \mathcal{D}^h} \max_{t \in D^h} Q^t$$

For all other schedules h', considered in decreasing order regarding frequency, choose the day combination that satisfies:

$$\min_{D^{h'} \in \mathcal{D}^{h'}(D^h)} \max_{t \in D^{h'}} Q^t$$

If $\mathcal{D}^{h'}(D^h) = \emptyset$ the day combination for h' is chosen in the same way as for h. After day combinations have been found for all schedules, set $Q^t = Q^t + q_e^h$ for each $t \in D^h$ and for all $h \in \mathcal{H}$.

History The last selection rule considers the day combinations chosen for previously treated edges, when choosing a day combination for the current edge. This strategy is also used in the Master-Route heuristic in Section 3.3. There are, however, minor differences. The concept is to consider the selections made for the last few edges and to seek a day combination similar to what was chosen for those edges. If the overall algorithm considers the edges in an order such that edges treated after each other are close in the graph (using edge order given by a depth first search), this ensures that edges near each other are more likely to be added to the same service days.

Before we go into detail with this strategy, some additional notation is needed. Define u^h to be the number of edges assigned to any route since the previous occurrence of positive demand for schedule $h \in \mathcal{H}$. If the schedule has not occurred yet, set $u^h = \infty$. Similarly, define $\tilde{D^h}$ to be the last chosen day combination for the demand schedule h. If no day combination has yet been chosen for the schedule, set $\tilde{D^h} = \emptyset$. These values are used to help select the day combination for the next edge. Furthermore, define $\hat{D^h}$ to be the preferred day combination to be chosen for demand that is serviced by schedule h. These values are used when $\tilde{D^h} = \emptyset$, which implies that a new empty route was created. The preferred day combination for a demand schedule h is the demand schedule that minimizes the largest number of routes on any day included in the day combination. More formally stated, define r^t to be the number of routes on day t. For each h, $\hat{D^h}$ is chosen such as to satisfy

$$\hat{D^h} = \arg\min_{D \in \mathcal{D}} \max_{t \in D} r^t$$

where

$$\mathcal{D} = \begin{cases} \mathcal{D}^h(D^{h'}) & h \neq h',\ h' \geq h\ \forall\ h \in \mathcal{H} \\ \mathcal{D}^h & \textit{otherwise} \end{cases}$$

Lastly, define Q_r^t to be the total demand added to the currently active route on day t.

For each edge, the day combination is chosen based on these three values: Last chosen day combination $\tilde{D^h}$, number of edges added since the demand schedule occurred with positive demand u^h, and in the cases where $u^h = \infty$ and $\tilde{D^h} = \emptyset$ the preferred day combination $\hat{D^h}$. See a pseudo code in Algorithm 1.

Each demand schedule with positive demand on the edge is treated separately, starting from the most frequent schedule and ending with the least frequent schedule. To decide what schedule should be chosen for the most frequent schedule $h^* \in \mathcal{H}$

with positive demand on the current edge, we need to identify the last chosen day combination for a schedule that is at least as frequent.To do this we first identify the most recent schedule that has as high a frequency as h^*:

$$u^{h'} = \min_{\nu \in \mathcal{H}, h^* \leq \nu} u^{\nu}$$

If $u^{h'} = \infty$, that is, no day combination was chosen since last reset, we choose the preferred day combination to be the day combination for service schedule h^* on e:

$$D^{h^*} = \hat{D^h}$$

Otherwise the day combination we choose for h^* on the current edge should be contained in the day combination last chosen for schedule h'

$$D^{h^*} = \arg \max_{D \in \mathcal{D}^{h^*}(\tilde{D}^{h'})} \min_{t \in D} (W - Q_r^t)$$

That is, the day combination for the previously visited service schedule, at least as frequent as h^*, define the day combination for the current demand. If several day combinations are available, the one that has the most remaining capacity is chosen. Note that the above implies that if $u^{h'} < \infty$ and $h^* = h'$, then $D^{h^*} = \tilde{D}^{h*}$.

For all other demand schedules $h \in \mathcal{H}$ with positive demand on edge e, considered in decreasing order, the day combination is chosen such that it is contained in the chosen day combination on this edge for schedule $h*$:

$$D^h = \arg \max_{D \in \mathcal{D}^h(D^{h^*})} \min_{t \in D} (W - Q_r^t)$$

If $D^h = \infty$, the last chosen day combination $\tilde{D^h}$ is chosen. If $\tilde{D^h} = \emptyset$, the preferred day combination $\hat{D^h}$ is chosen.

For each service day t in the chosen day combination, we set $Q_r^t = Q_r^t + q_e^h$ to keep track of remaining capacity in the current route. If adding demand q_e^h exceeds the capacity of the vehicle, the route is closed and a new route is created on day t setting $r^t = r^t + 1$. Furthermore, u^h, $\tilde{D^h}$, and Q_r^t are reset to the initial values and new preferred day combinations are found.

2.4 Characteristics of the final heuristics

When we combine the five edge orderings above and the three selection rules for choosing day combinations above, we get 15 variations of the heuristic and each of them yields different solution characteristics. The procedures that use random edge ordering and random choice of day combination for a given edge are used to generate completely random solutions. The solutions will vary and thereby represent a large part of the solution space. These solutions may be useful in a population based meta heuristic procedure.

The procedures that choose day combinations in order to spread demand on all days seek to find solutions that favor the first objective, which is to use the least number of vehicles on every day in the planning horizon.

When ordering the edges according to a depth first search, the next edge to be considered will often be close to the edge just added to the routes. Therefore it

Algorithm 1 Choose day combination history

Require: Edge set $\mathcal{E}$, ordered schedule set $\mathcal{H}$ with most frequent schedule first.
$u^h = \infty$, $\tilde{D}^h = \emptyset$, $\hat{D}^h = \arg\min_{D\in\mathcal{D}} \max_{t\in D} r^t \ \forall h \in \mathcal{H}$
$Q_r^t = 0 \ \forall t \in \mathcal{T}$
for all $e \in \mathcal{E}_R$ **do**
freqFound=FALSE
$D^{h^*} = \emptyset$ chosen schedule for most frequent demand schedule on edge e.
$D^h = \emptyset \ \forall h \in \mathcal{H}$.
for all $h \in \mathcal{H}$ with $q_e^h > 0$ **do**
if freqFound=FALSE **then**
freqFound=TRUE
Find $h' \in \mathcal{H}$ such that:
$u^{h'} = min_{\nu\in\mathcal{H}, h\leq\nu} u^\nu$
if $u^{h'} = \infty$ **then**
$D^{h^*} = D^h = \hat{D}^h$
else
$D^{h^*} = D^h = \arg\max_{D\in\mathcal{D}^h(\tilde{D}^{h'})} \min_{t\in D}(W - Q_r^t)$
end if
else
$D^h = \arg\max_{D\in\mathcal{D}^h(D^{h^*})} \min_{t\in D}(W - Q_r^t)$
if $D^h = \emptyset$ **then**
if $\tilde{D}^h = \emptyset$ **then**
$D^h = \hat{D}^h$
else
$D^h = \tilde{D}^h$
end if
end if
end if
end for
reset = FALSE
for all $h \in \mathcal{H}$ with $q_e^h > 0$ **do**
for all $t \in D^h$ **do**
$Q_r^t = Q_r^t + q_e^h$
if $Q_r^t > W$ **then**
reset=TRUE
$Q_r^t = q_e^h$
end if
end for
end for
if reset=TRUE **then**
$u^h = \infty$, $\tilde{D}^h = \emptyset \forall H \in H$
$\hat{D}^h = \arg\min_{D^h\in\mathcal{D}} \max_{t\in D^h} r^t \ \forall h \in \mathcal{H}$
end if
end for

is likely that an optimal solution would add them to the same route. When we combine these orderings with the procedure that chooses the day combination based on what was chosen in recent iterations, this implies that edges close to each other are serviced on the same days, and hence these procedures yield solutions with a low value on the second objective, i.e., minimization of routing cost. In order to avoid all demand to be assigned to the same days, the day combination procedure is reset whenever a route is filled. The next choice of day combination is guided towards spreading demand on days by the choice of preferred day combination and thereby also seeking to favor the value of the first objective.

2.5 Post-improve using the classical CARP Path Scanning on individual days

The routes built by the above procedure heavily depend on the order in which the edges were considered. This implies that routes on individual days may be improved. So in order to improve the solution found, the following post-optimization process was tested. As only individual days are to be improved and no reassignment of day combinations will take place, each day will represent a CARP. So a popular construction heuristic for the CARP, the Path Scanning heuristic, will be used to build routes on each day separately. We start by outlining the different versions of Path Scanning available for the CARP. At the end of this section we will describe how the algorithm is used as a post-optimization procedure on an SP-CARP.

The Path Scanning was first introduced in [4]. Here, a route is built starting from the depot and adding an adjacent required edge. This is repeated until the vehicle capacity has been reached or until there are no adjacent edges to the end node of the route. In both cases, the route will return to the depot and a new route will be initiated. In cases with more than one adjacent edge, ties are broken using one of the following five rules, where q_e is the demand on edge e:

1) Maximize $\frac{c_{ij}}{q_e}$.

2) Minimize $\frac{c_{ij}}{q_e}$.

3) Minimize distance back to depot after service.

4) Maximize shortest distance back to depot after service.

5) If vehicle is less than half full, use rule 4, else use rule 3.

The heuristic works by constructing one solution using each of the five criteria in turn and returning the best of these five solutions.

Two random versions of the Path Scanning are described in [1], the first is called Path Scanning with Random Criteria. Here, one of the five tie breaking criteria above is chosen at random each time ties are broken. The second version is called Path Scanning with Random Task, in which ties are broken at random, and hence the five criteria above are no longer used.

Later, [6] modified the algorithm allowing it to jump to the nearest non-serviced demand edge when no demand edges are adjacent to the end node of the route.

In case the vehicle capacity has almost been used up, the search for edges with low demand may result in routes that zigzag across the graph when the heuristic

is allowed to find the nearest node with non-serviced adjacent edges. To prevent that, [7] improved the random task version by including the ellipse rule. This rule forces the route to serve edges close to the path back to the depot when the vehicle is almost full.

In [8], Path Scanning was incorporated with an Augment Merge algorithm to form the Double Outer Scan heuristic. The idea is to first build routes by choosing the non-serviced demand edge furthest away from the depot (total distance from both end nodes to depot). Then each end is scanned in a Path Scanning way in which edges incident to either end node are considered. The incident edge furthest away from the depot is added to the route. This continues until no more incident unserviced demand edges exist or until the capacity limit is reached. In both cases each end is connected to the depot. Next, routes are merged in order to improve cost.

We post-improve using Path Scanning on the individual days using three versions of path scanning. One version of the Path Scanning builds a solution for each of the five traditional tie-breaking criteria and returns the best of these five. The two other versions are the random versions described above. In all three versions we allow jumps to the nearest non-serviced edge if no adjacent non-serviced edges exist, and a simplified version of the ellipse rule is included such that we only consider edges that are no further away than the depot when the vehicle is filled to a given limit, αW where $\alpha \in [0, 1]$.

When an edge is serviced, all demand schedules that are to be serviced on that day will be serviced. That is when running the Path Scanning on day t the demand for edge e is $q_e = \sum_{h \in \tilde{\mathcal{H}}} q_e^h$, where $\tilde{\mathcal{H}}$ denotes all service schedules in which the chosen day combination includes t.

Furthermore, each of the three Path Scanning procedures above was used in a Double Outer Scan setting where instead of the merge phase we scan from both ends of the route (i and j) and where the nearest edge (adjacent to the route or not) is included in the route. Ties are broken in the same way as for the normal Path Scanning. This continues until no more edges can be included in the route and then both ends are connected to the depot. The method is developed in collaboration with Hani Zbib, Aarhus University, who has used it for another variation of the CARP.

Altogether, we thus have six Path Scanning procedures that were tested to improve routing cost on individual days and thereby reduce the overall routing costs:

- Path scanning from depot:
  - Return best of five solutions found using traditional tie breaking criteria.
  - Return best of ten solutions found by breaking ties randomly.
  - Return best of ten solutions found by randomly choosing one of the five traditional criteria each time ties are broken.
- Path scanning from unserviced edge furthest away from depot:
  - Return best of five solutions found using traditional tie breaking criteria.
  - Return best of ten solutions found by breaking ties randomly.
  - Return best of ten solutions found by randomly choosing one of the five traditional criteria each time ties are broken.

3 The Master-Route heuristic for the SP-CARP

In this section, we present The Master-Route heuristic for the SP-CARP. It uses components from the Path Scanning procedures for the CARP, [1, 4, 6, 7, 8], which are described in Section 2.5. We first outline the structure of the heuristic. Secondly, we specify how to break ties when adding edges. Finally, we explain the criteria for choosing day combinations.

3.1 Master-routes and scanning technique

In this SP-CARP version, routes are built on several days simultaneously by building a Master-route R_l with l being a counter $0 \leq l \leq s$ where s is the number of Master-routes built. Like in the algorithm described in Section 2 each day $t \in \mathcal{T} \setminus \mathcal{T}_0$ contains an active route. These active routes are connected to the active Master-route. A Master-route is built by adding the current edge to the end of the active route in all the days given by the chosen day combinations simultaneously. If the active route in one day does not have sufficient capacity to service the edge, all open routes (that service at least one edge) on all days are closed and new routes are created, (in Section 2 we only closed the route lacking capacity). So the Master-route determines which edge to service next and builds routes on several (possibly all) days at the same time. The Master-route currently built is denoted the active Master-route. To this active Master-route we connect an active route for each service day in the planning horizon. Edges are added to these active routes for the given days. By using this Master-route concept, we only have one end node to consider when scanning for the next edge to add, even though the active routes on individual days might not end at the same node.

The Master-route starts from the depot (or from the unserviced edge furthest away from the depot). This will result in end node(s) i (and j). Like in the normal Path Scanning, we consider the nearest unserviced edges to the end node(s). Only edges that have a valid day combination where demand can be added to the active route of all days in the given day combination are considered.

For each $h \in \mathcal{H}$ with $q_e^h > 0$, a valid day combination is chosen using one of the selection rules described in Section 3.3, and e is added to the end of the active route of each day in the chosen day combinations with demand q_e^h. If demand from several demand schedules is added to the same day, the total demand will be serviced at once. The new end node of the Master-route is updated and becomes the end node of e furthest away from the current end node(s). This continues until no more edges can be added to the Master-route or until all demand edges are serviced. No more edges can be added to the Master-route if no edge in the graph (or within the limit) has a day combination, where all the open routes on the days in the day combination have enough capacity. Similar to the Path Scanning procedures used in Section 2.5, the edges further away from the end node than the depot are ignored when the Master-route is filled to a given limit. This limit is determined by the active route that has the least remaining capacity. A Master-route is filled when one active route connected to the Master-route is filled. When this happens, the Master-route is closed; all active routes that service any edges are closed (no longer active) and new empty active routes are created on these days and paired with a new (active) Master-route along with the empty routes of all the days that did not

service anything in the old Master-route.

To formalize the algorithm above we need to solve two aspects. The first is how to break ties between non-serviced edges that are equally close to the end node(s) and the second is how to choose day combinations for the selected edge. Section 3.2 explains the tie breaking strategies used for choosing edges, and Section 3.3 describes the rules used for choosing day combinations.

3.2 Breaking ties when choosing next edge

Given a partial Master-route with end node i, the next edge to be added will always be the one closest to i. However, it might be the case that several edges are equally close to i. In that case tie breaking rules are needed.

When working with SP-CARP, edges do not only vary in cost, demand, and travel distance back to depot, but also in the service schedules for demand on the edge. Therefore, we consider tie breaking criteria relevant for the Master-Route heuristic. As cost, demand, and return distance to depot are still highly relevant in this CARP variation, the five traditional tie breaking criteria described in Section 2.5 will still be considered. In addition to these criteria we suggest the following four criteria;

7) Choose the edge that has the most frequent service schedule.

8) Choose the edge with the least frequent service schedule.

9) Use criteria 7 if the vehicle is less than half full, and criteria 8 otherwise.

10) Use criteria 8 if the vehicle is less than half full, and criteria 7 otherwise.

Criteria 7 seeks to favor edges that have demand that are to be serviced very often and therefore have less valid day combinations as this gives less flexibility regarding the days to service the given demand. Hence, if a day combination exists, where the active Master-route has remaining capacity on all days included in the day combination, then that edge should be serviced. Criteria 8 on the other hand, exploits the flexibility of an edge that has demand that is to be serviced few times during the planning horizon and thereby has several days on which it can be serviced. Criteria 9 and 10 combines 7 and 8.

If several edges are equally good according to the chosen criteria, one of them is chosen randomly.

3.3 Choose day combination

When an edge has been chosen, it must be decided which day combination to use for this edge. We use three difference strategies that are very similar to those used for the Edge Order heuristic presented in Section 2. One strategy that chooses a day combination completely at random, one that spreads demand evenly on all days, and one that takes the previous choice into consideration. All versions were described in Section 2.3. However, minor modifications were made to the last one in order to exploit the Master-routes used to build the individual routes. These differences will be explained in this section.

The difference is that variables are defined and updated according to the Master-route and not according to events on individual days. So the variables used to determine the previous choice are redefined according to Master-routes: Define u_M^h to be the number of edges visited on the current Master-route since the previous use of demand schedule $h \in \mathcal{H}$. If the schedule has not occurred on the current Master-route, then $u_M^h = \infty$. Similarly, define D_M^h to be the previously chosen day combination for the demand schedule h on the current Master-route. If the demand schedule have not yet occurred on the Master-route, then $D_M^h = \emptyset$. Furthermore, we define $\hat{D}_M^h$ to be the preferred day combination to be chosen for demand that is serviced by schedule h; these are used when no demand belonging to demand schedule h has occurred on the current Master-route.

When building the Master-route, these values are used to determine the day combination for the next edge in the same way as in Section 2.3. The difference is that they are now reset every time a new Master-route is created, whereas for the Edge Order heuristic they were reset each time a route on any day was filled.

4 Tuning findings

This section first gives an overview of the components of the two algorithms and their combinations and then it outlines the combinations of components that should be studied further after the initial tuning. We discuss these issues for each of the two algorithms separately.

4.1 The Edge Order heuristic

We first focus on the algorithm presented in Section 2. To recap, there are three main components. The order in which the edges are considered, the rule used to choose day combination for each demand type on an edge, and whether to try to reduce cost by running Path Scanning on all days individually. This gives a total of 30 variations of the Edge Order heuristic to test. An overview of the different variations to use in each of these components can be found in Table 2. The letters in brackets will be used when referring to these variations in the remaining part of this section.

Component	**Versions**
Edge order	DFS depot root (DFS-D) DFS random root (DFS-R) Random (R) BFS random root (BFS-D) BFS depot root (BFS-R)
Day comb. rule	Random (R) Spread demand (SD) History (H)
Post-opt.	Yes No

Table 2: Overview over components for the Edge Order heuristic described in Section 2

To find the combinations that resulted in the best performing algorithms, each combination of components was run on 15 of the 249 SP-CARP instances presented

in [5]. Table 3 list the instances used. For each combination, we evaluated the performance regarding the two objectives and runtime. A combination was said to outperform another combination if it performed better on at least one of these measures and not worse on any of the others. These initial tests showed the following:

Graph	F13_g O17_g	F15_g O6_p	F6_g O6_p	F9_g O9_g	K11_g S1_g	K3_g S11_g	N4_g S6_p	N9_p
Vehicle	2-8 2-2	5-24 1-4	4-8 5-4	6-24 6-2	6-2 1-2	2-2 5-2	4-2 3-4	5-4

Table 3: SP-CARP graphs and vehicles that make up the 15 instances used for initial tuning

- Using rule SD to choose day combinations is always better than using rule R.
- Treating the edges in an order given by rule DFS-D or BFS-D is always better than using rule R or BFS-R.
  - In the case where days are chosen using rule SD, also the order given by using rule DFS-R is worse than the two stated above.
- When post-optimization is not used, using rule DFS-D is better than BFS-D. Combined with the previous statement this leaves DFS-D as the best ordering when post-optimization is not used.
- When the edge order is given by DFS-D and the days are chosen according, H, post-optimization does not give a significant improvement in total cost and is therefore not used in the remaining part of the paper. In other cases post-optimization seems to be beneficial and will be used.

This leaves us with the following variations that are not generally outperformed. Two combinations when post-optimization is not chosen:

- DFS-R combined with SD.
- DFS-R combined with H.

Five combinations when post-optimization is chosen:

- DFS-D combined with SD.
- BFS-D combined with SD.
- DFS-D combined with H.
- BFS-D combined with H.
- DFS-R combined with H.

We wish to select three of these seven combinations for further study. They are chosen so as to have different strengths. We choose:

- DFS-R combined with H without post-opt. (denoted EO3).

  It was chosen as it is a fast method and is able to compete with the same combination with post-opt..

- DFS-R combined with H including post-opt. (EO2)

- DFS-D combined with SD including post-opt. (EO1)

  These were chosen in order to get methods that perform the best on one of the two objectives.

For the two combinations where the post-improvement phase is included in the algorithm, different settings for the Path Scanning ware also tested (these settings are not summarized in Table 2). Initial tests showed that using a Path Scanning with Random Criteria on individual days did well. The Double Outer Scan version was not beneficial compared to the traditional version. Furthermore, the initial tuning showed that $\alpha = 0.8$ is a good choice.

4.2 The Master-Route heuristic

For the algorithm described in Section 3 we have four main components: How to choose day combinations, how to build routes, which tie breaking criteria to use, and the vehicle fill rate to use when limiting the allowed jumps when finding un-serviced edges. An overview of these components can be found in Table 4.

Component	**Versions**
Day comb. rule	Random (R) Spread demand (SD) History (H)
PS method	Traditional (T) Double (D)
Tie break	Trad 5 (T) Random Criteria (RC) Random (RC) Most freq. (MF) Least freq. (LF) MF until half (MFH) LF until half (LFH)
Fill rate	$(0, 1]$

Table 4: Overview over components for the Master-Route heuristic described in Section 3

Again, all combinations of the first three components were tested on the same set of SP-CARP instances as was used for the Edge Order heuristic. For the fill rate (α) we tested six different values spread over the valid interval; 0.2, 0.4, 0.5, 0.6, 0.8, and 1. Initial testing showed that α should be 0.8 or 1. Decreasing it further does not result in better cost and increases the number of routes needed. The main difference between setting α to 1 or 0.8 is that the runtime is higher for $\alpha = 1$ and the number of vehicles used is highest for $\alpha = 0.8$ whereas the cost is lowest for the latter. Thus with two values for α we have a total of 84 variations of the algorithm. For variations including SD or H and MF, LF, MFH, or LFH (in total 32 variations) we have deterministic rules in every step. However, when two edges are

the same according to these rules, either can be chosen. This gives some degree of randomness. Therefore these variations were run with different number of iterations (1 and 10 iterations) to evaluate the benefit of multiple runs.

Further testing resulted in the following regarding the other components:

- Scanning from depot (T strategy) performs better than scanning from an edge far away (D strategy).
- The choice of tie breaking criteria does not seem to have a huge impact on the final solution. This might be due to the fact that all variations have some degree of randomness included.
- For the 32 variations run with two different numbers of iterations,we found that the total cost could be improved by increasing the number of runs. Increasing the number of iterations improved the cost by 3.3 to 6.6 percent on average. The number of vehicles used were not influenced.

This leaves us with 42 variations. Seeking three variations with different strengths we chose one variation for each day combination choice rule combined with PS method T. The tie breaking strategy and α were chosen such that these three variations varied the most:

- Master-Route heuristic with R day combination rule with tie breaking criteria T and $\alpha = 1$ (denoted MR-R).
- Master-Route heuristic with SD day combination rule with tie breaking criteria MFH and $\alpha = 1$ (denoted MR-SD).
- Master-Route heuristic with H day combination rule with tie breaking criteria MF and $\alpha = 0.8$ (denoted MR-H).

The final choice of tie breaking criteria and fill rate was based on minor variations in the tuning results.

5 Computational results

At this point, we have chosen three versions of each of the two algorithms presented in this paper. This section shows the performance of these six variations when tested on all 249 SP-CARP instances presented in [5]. We focus on the performance regarding the two objectives, i.e. (1) the number of vehicles used and (2) the total cost. In addition we consider the runtime. To some extend all algorithms have random elements. Therefore, Section 5.1 looks into the benefit of multiple runs. Section 5.2 is dedicated to a comparison of the performance of the different algorithms as regards the two objectives and the runtime.

5.1 Benefit of multiple runs

As all six algorithms contain random elements, we chose to run the algorithms 10 times and return the best out of these 10 solutions. The best of the 10 solutions is defined as the one using the lowest number of vehicles and secondly the one with

		EO1	EO2	EO3	MR-R	MR-SD	MR-H
Add. vehicles	F	0.10	3.00	3.10	0.36	0.07	1.55
	N	0.19	10.15	10.37	0.17	0.02	5.39
	O	0.15	23.05	24.18	0.10	0.00	15.31
	K	0.18	15.33	15.23	0.07	0.02	8.30
	S	0.14	6.93	7.04	0.04	0.04	4.79
	all	0.15	11.41	11.65	0.14	0.03	6.82
% increase cost	F	5%	8%	4%	13%	11%	12%
	N	3%	6%	3%	8%	8%	8%
	O	3%	2%	3%	7%	7%	4%
	K	3%	5%	3%	8%	8%	6%
	S	4%	6%	4%	10%	8%	8%
	All	4%	5%	3%	9%	8%	8%

Table 5: Compare best solution to worst performance in 10 runs for each algorithm

the lowest cost. In this section we see how much the number of vehicles and the cost vary during these 10 runs, which should give a better understanding of the actual benefit of these multiple runs.

The top part of Table 5 shows the number of additional vehicles used in the worst solution compared to the best solution from the 10 runs as an average over all instances. In Figure 1 the same information is shown at an instance level. The number of vehicles used in a solution varies the most for the algorithms EO2, EO3, and MR-H. All of these algorithms choose the days an edge is serviced based on which days were chosen for the last edge. In order to avoid that all edges are added to the same days, the last chosen day combination is reset every time a route is filled. It could be this resetting factor that leads to the big variation in the number of vehicles used in different runs of the algorithm. Furthermore, it is noted that the variations in the number of vehicles are by far the highest for the O area followed by the K area. These two areas vary in structure as the K area is semi-urban and the O area is urban. Hence, it is not the structure of the area that affects the variation in number of vehicles.

Next we consider the lower part of Table 5, which shows the average increase in cost (in percentages) for the solution giving the highest cost compared to the best found solution (note that this solution is not necessarily the solution with the lowest cost as the number of vehicles has the highest priority). These averages are also accompanied by plots showing the performance for each graph (see Figure 2). As an addition, individual plots are made for areas F, K, N, O, and S to make it a bit more clear. In this case it is not straightforward to conclude if one or two algorithms in general provide a better improvement compared to the other algorithms. This could be due to the fact that all algorithms first seek to improve the number of vehicles used and secondly seek to improve the cost. From a cost perspective it could be better to service everything on a few days, even though more vehicles will be needed on those days. The averages can therefore only be used to give an overall idea of the impact regarding cost of multiple runs. Furthermore, there are some variations from area to area regarding which algorithm has the largest variation in cost. For

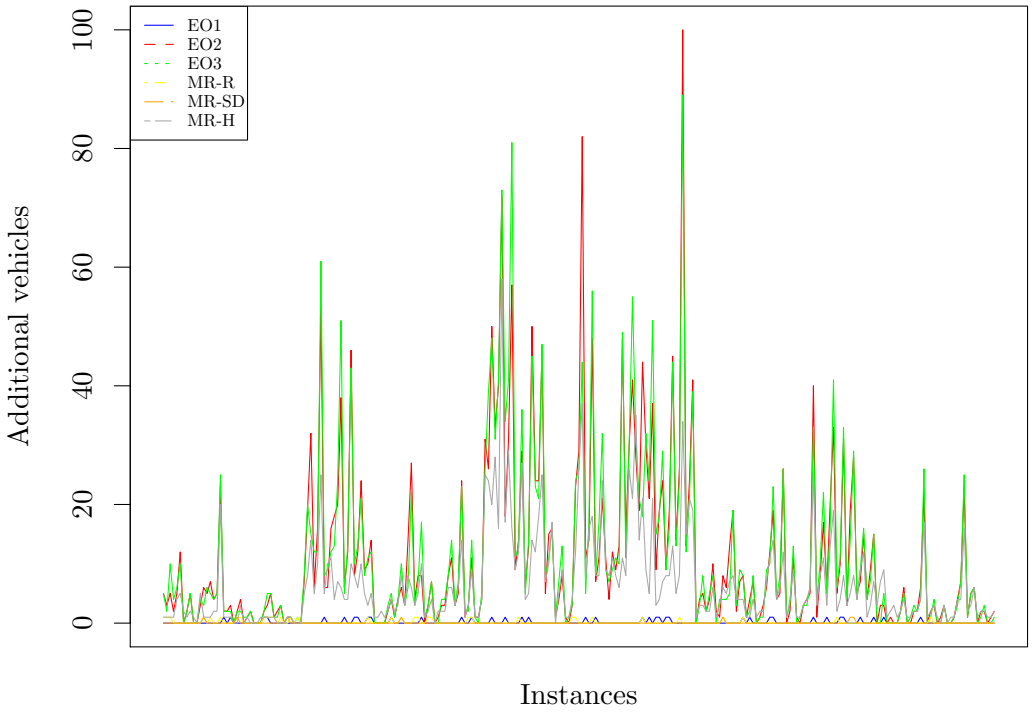

Figure 1: Additional number of vehicles in worst solution compared to best solution out of 10 runs

instance MR-H is among those with the largest variation in the F area and the least variation in the O area. Both of these areas are urban in structure so the explanation cannot be found in differences in the geographical areas represented by the graphs.

Looking at the averages both for all instances and separated into areas, we see that the largest benefit of multiple runs regarding cost is obtained for the SP-PS versions. This tendency is also obvious in the plots as they are not close to zero for any instances. However, for the EO2 version, we see that there is a great degree of variation of the benefit of multiple runs as for some instances we hardly gain anything, whereas for other instances this algorithm gains the most.

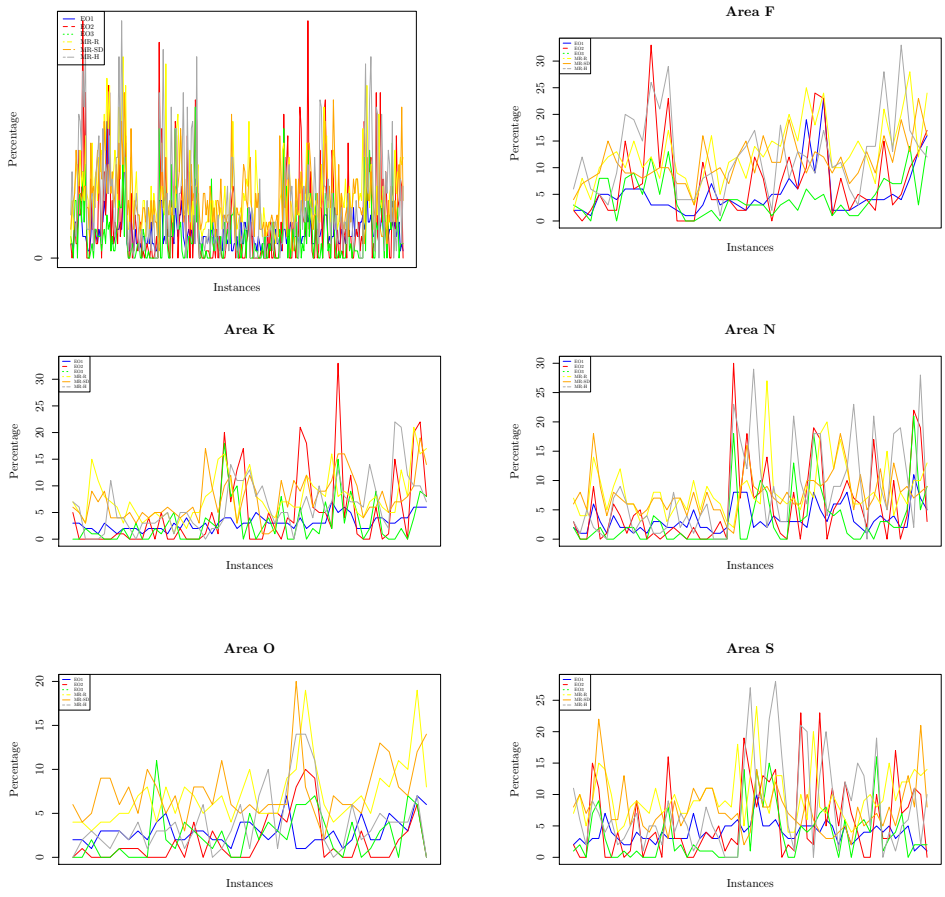

Figure 2: Additional cost in percentages in worst solution compared to best solution out of 10 iterations

As all the versions of our algorithm vary in number of vehicles used or total cost (or both), we find it beneficial to do multiple runs. This also imply that our algorithm to some extend generate solutions that have different attributes which can be useful when creating a population of solutions in a meta heuristic. However, we note that the runtime of EO1 is quite high (see Section 5.2) and it is one of the versions that improve the least in both number of vehicles and cost. Therefore, it could be argued that one iteration of the algorithm is sufficient and that this version is less useful when generating many initial solutions for a meta heuristic.

5.2 Performance

The focus will now turn to the performance of each of the six versions of our algorithms when performing 10 runs and returning the best solution found. As no study has ever solved the 249 instances in [5], we cannot compare the results of our algorithms to the optimal solutions or to previously found lower bounds. Instead we compare the six algorithms among themselves in order to investigate their performance regarding the two objective values and their runtimes. This will give some insight into the attributes of each algorithm.

In order to compare the performance among the heuristic variations for each instance, we define the following measure:

$$\frac{\pi_a - \min_{\tilde{a} \in \mathcal{A}} \pi_{\tilde{a}}}{\min_{\tilde{a} \in \mathcal{A}} \pi_{\tilde{a}}}$$

where $\mathcal{A}$ is the set of algorithms and π_a is the objective value or runtime for algorithm a for the given instance. The average performance over all instances for each measure is stated in Table 6. To see that these averages are representative for the overall performance, Figures 3, 4, and 5 gives the information for each instance.

		EO1	EO2	EO3	MR-R	MR-SD	MR-H
Vehicles	F	7.1	83.8	80.2	24.5	0.5	75.9
	N	10.2	147.0	146.4	0.6	0.0	84.8
	O	2.3	232.3	223.2	0.0	0.0	125.4
	K	3.3	202.1	210.7	0.0	0.0	119.5
	S	5.8	166.5	177.3	0.5	0.0	55.3
	All	5.8	166.8	169.0	4.4	0.1	90.8
Cost	F	38	2	51	73	72	14
	N	90	6	36	123	128	1
	O	111	10	27	157	168	0
	K	119	8	31	158	168	1
	S	122	8	32	152	158	2
	All	99	7	35	134	141	3
Runtime	F	13,933	7,090	0	90	68	167
	N	58,790	27,830	0	4,663	4,022	11,892
	O	83,322	36,545	0	8.598	7,632	19,346
	K	186,318	97,969	0	11,231	9,819	17,065
	S	52,085	22,932	0	3,874	3,077	4,715
	All	82,725	40,631	0	5,831	5,031	10,623

Table 6: Average percentage increase in number of vehicles, cost, and runtime compared to the algorithm that performed best each the instance.

5.2.1 Vehicles

The first objective value to be optimized is the number of vehicles needed to execute the solution. This is defined by the maximum number of routes included in any day in the planning horizon.

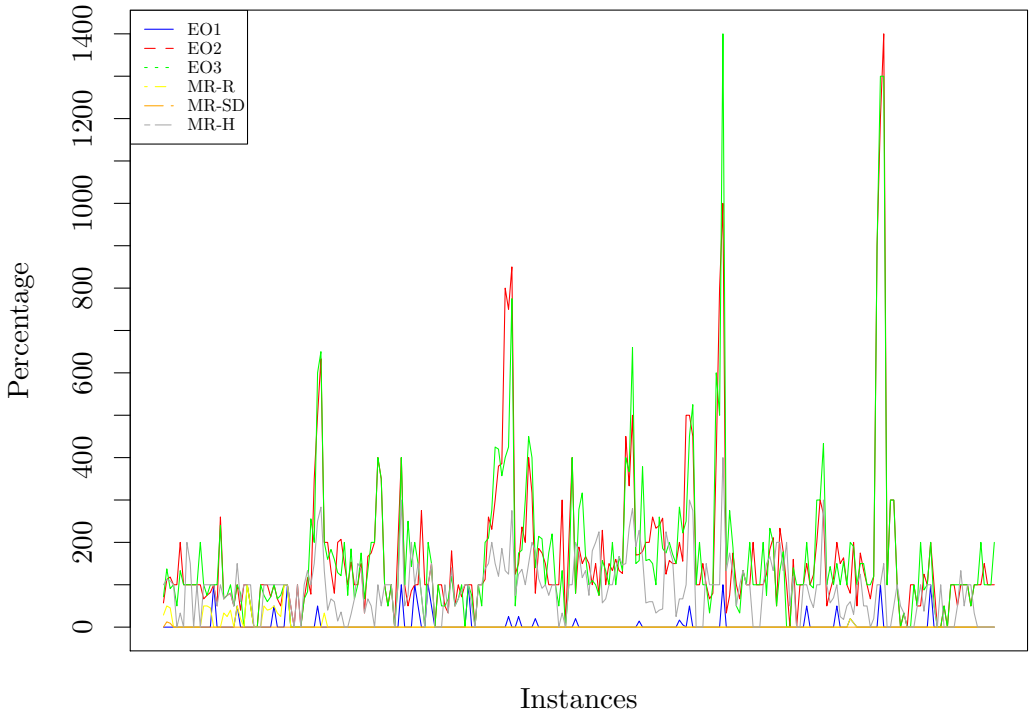

Figure 3: Percentage of vehicles in solution found compared to the algorithm with the lowest number of vehicles.

From the top part of Table 6, we note that MR-SD is the best algorithm in this respect. This algorithm is almost always the one that finds the solution with the lowest number of vehicles (ith just a few exceptions in the F area). If we consider Table 5, we see that the variation in the number of vehicles is low and therefore it is likely that we have high performance on the number of vehicles needed even with only one run.

The runners-up are MR-R and EO1, which are only marginally worse. They also seek to spread demand on all days either by assigning days at random or by choosing days with the least total demand so far. It could have been expected that EO1 would perform better than the SP-PS versions as it also seeks to distribute demand evenly across all days and it only returns to the depot if the edge currently added to the solution cannot be serviced by this route on this day. In contrast, the SP-PS versions close routes on several days as soon as a route on one day is not able to service the current edge. One explanation for the observed behavior could be that the SP-PS seeks to spread demand evenly on all days and favors edges with frequent demand when the vehicles have plenty of capacity. Thereby less frequent edges are added to days that are less occupied, which eventually lowers the number of vehicles needed on busy days. So presumably the tie breaking criterion has an impact here. Furthermore, the SP-PS chooses the next edge to add based on the remaining capacity and can thereby find an edge with less demand if necessary whereas EO1 is forced to try to add the edge given by the edge order even though a nearby edge could have been serviced by the current route.

Table 7 shows the average fill rate of the vehicles over all instances. When we combine this table with the top part of Table 6, we clearly see that there is no relation between the ability of an algorithm to utilize the capacity of all vehicles and its ability to find solutions that only need few vehicles.

		EO1	EO2	EO3	MR-R	MR-SD	MR-H
Fill rates	F	0.67	0.76	0.78	0.54	0.67	0.59
	N	0.74	0.84	0.85	0.77	0.78	0.80
	O	0.82	0.89	0.89	0.83	0.83	0.79
	K	0.75	0.86	0.85	0.76	0.76	0.73
	S	0.65	0.81	0.82	0.66	0.68	0.80
	All	0.72	0.83	0.84	0.71	0.74	0.75

Table 7: Average vehicle fill rates

The remaining three algorithms perform relatively poorly regarding the number of vehicles needed. This is as expected as they all seek to lower cost by choosing days to service depending on what was chosen for the last edge and thereby they focus more on cost compared to number of vehicles. Returning to Table 5 we note that without running the algorithm several times this is likely to be even worse regarding the number of vehicles needed. Furthermore, we see that the performance of MR-H is somewhere between the three best versions and the two worst. The results in Table 8, which shows the percentages of unused service days in the planning horizon, may explain this finding. The three best versions use all service days. We see that MR-H is better at exploiting the available days compared to the two worst and thereby it

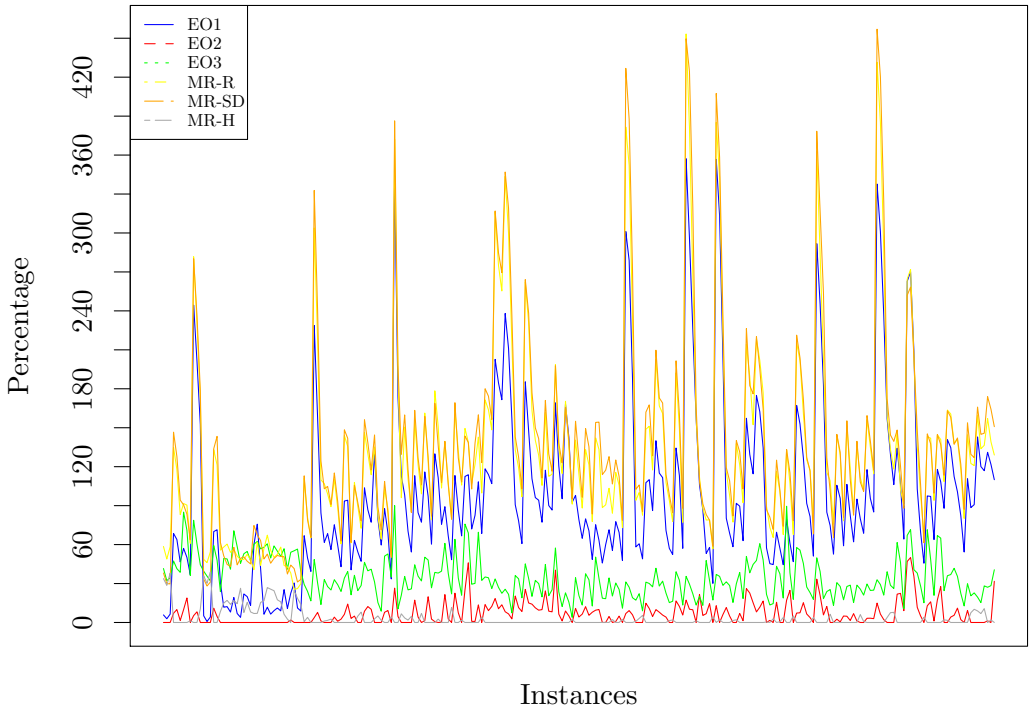

Figure 4: Additional cost in percentage in solution found by algorithm compared to algorithm with lowest cost.

can use fewer vehicles. This might be explained by the way we reset the last chosen day combination in the two algorithms.

	EO1	EO2	EO3	MR-R	MR-SD	MR-H
Average	0	36	35	0	0	24
Max	0	95	95	0	0	80
Min	0	0	0	0	0	0

Table 8: Percentage of service days without service. Average over best found solution in all instances.

5.2.2 Cost

The second objective value that we wish to optimize is the total routing cost. The performance regarding this value can be found in the middle part of Table 6 and in Figure 4. Here, the performance of the six versions of our algorithms is reversed compared to the first objective function. Hence the three algorithms that performed well regarding vehicle minimization now perform the worst, and the three that performed the worst as regards minimizing the number of vehicles perform well when considering routing cost. Again, this is due to the fact that the three that performed

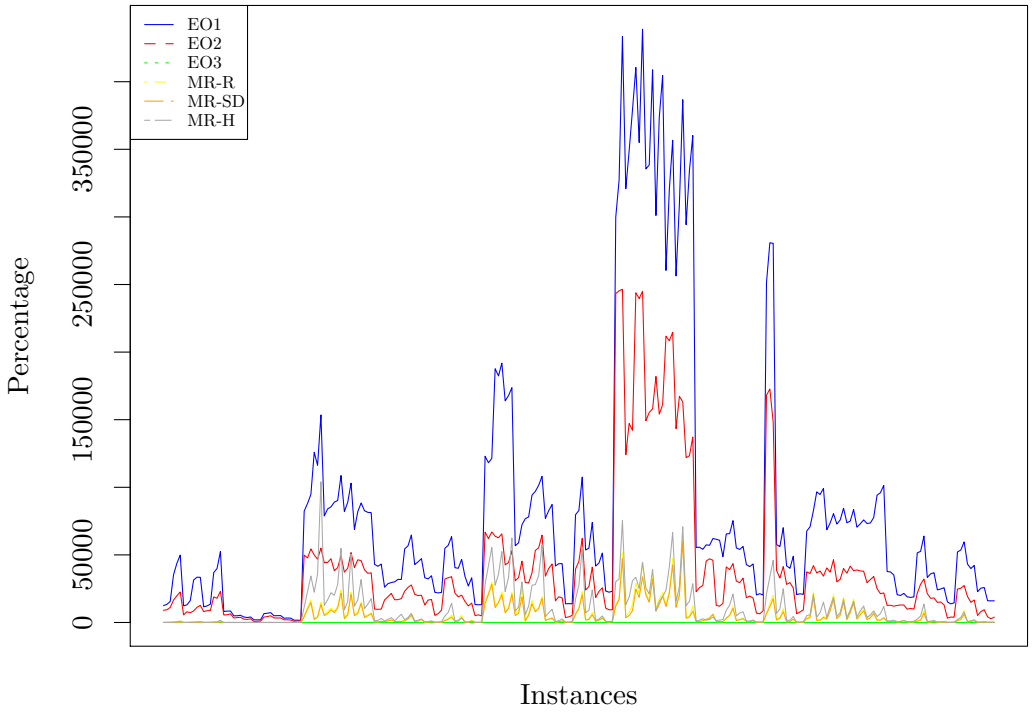

Figure 5: Additional runtime in percentage in solution found by algorithm compared to algorithm with lowest runtime.

well on vehicles focus on distributing the demand on all days and thereby trying to find solutions that need few vehicles on each day, whereas the other algorithms focus on adding edges that are geographically close to each other to the same day and thereby most likely to the same route. The latter is very likely to decrease cost.

EO2 and MR-H perform the best regarding total routing cost. In general, MR-H is best, but for the F area EO2 is better. Figure 4 reveals that at some point, the green curve drops below the blue curve and the gray curve drops below or to the red curve; this happens when we change from F area instances to N area instances. Hence, the relative performance of the algorithms seems to depend on whether we look at F area instances or the other areas. The averages in Table 6 tell the same story. This might be due to the fact that the service patterns of the F area are quite different compared to the other areas as explained in [5].

5.2.3 Runtime

The final measure we use to evaluate the algorithms is runtime. This is an important factor if we are to use any of the algorithms to generate many solutions for meta heuristics or if an updated solution have to be found quickly in a real world setting due to changes in the instance.

The results for this measure are shown in the bottom part of Table 6 and are supported by the plot in Figure 5. Here we see that EO3 is by far the fastest. To

get a better idea of how much faster, see Table 9 that shows the longest time each algorithm spends on any instance. We note that even for the large graphs, only nine seconds is needed when using EO3, whereas the slowest algorithm, EO1, needs 4.9 hours, which is very high if used as a constructive heuristic. The SP-PS versions all spend less than 30 minutes on solving the largest graphs, which is reasonable. Both EO1 and EO2 are extremely slow. This is due to the fact that, for each solution found, a post-optimization phase is run that seeks to improve each day individually by running a Path-Scanning for the daily CARP. This is very time consuming.

EO1	EO2	EO3	MR-R	MR-SD	MR-H
17,610	9,849	9	1,775	1,360	1,791

Table 9: Maximum runtime in seconds

The percentage of service days where cost was improved by running a Path-Scanning can be found in Table 6. It is seen that for EO1, the cost is improved for almost all days in all instances. This is due to the fact that this algorithm makes little effort to make good routes. For EO2 that actually tries to create good routes immediately (due to the order in which edges are considered and the way we choose day combinations) we see that there are more cases where a simple Path Scanning cannot improve the solution. Still, for both algorithms, running the post-optimization on individual days does improve the cost of the solution and can thereby be beneficial.

Note that each version of our algorithms can generate one solution in approximately $\frac{1}{10}$ of the time spent in our trials as they all have 10 runs. This would be the time used when generating one solution for a population of solutions in a meta heuristic.

A remaining point of interest regarding the runtime is whether the runtime for each algorithm depends on the size of the instance. A natural way to measure size is by using the number of nodes in the graph. Plots of number of nodes in the graphs against the runtime are shown in Figure 7. For five out of six algorithms we see that an increase in the number of nodes increases the runtime. Furthermore we see that the variety in runtime increases as the graphs increase in size.

For the last version, EO3, we see that a set of instances have a relatively high runtime even though the graphs are small (note that the runtime for this algorithm is generally very low compared to the other algorithms). A closer look at the numbers shows us that they are F area instances which are the least rural instances we have. This might imply that non-rural areas are more difficult to handle for this algorithm.

Other ways of measuring the size of an instance could be the number of required edges or the length of the planning horizon. Using the number of required edges gave the same result as the number of nodes, whereas no clear result was found when the length of the planning horizon was used. See Appendix B.1.

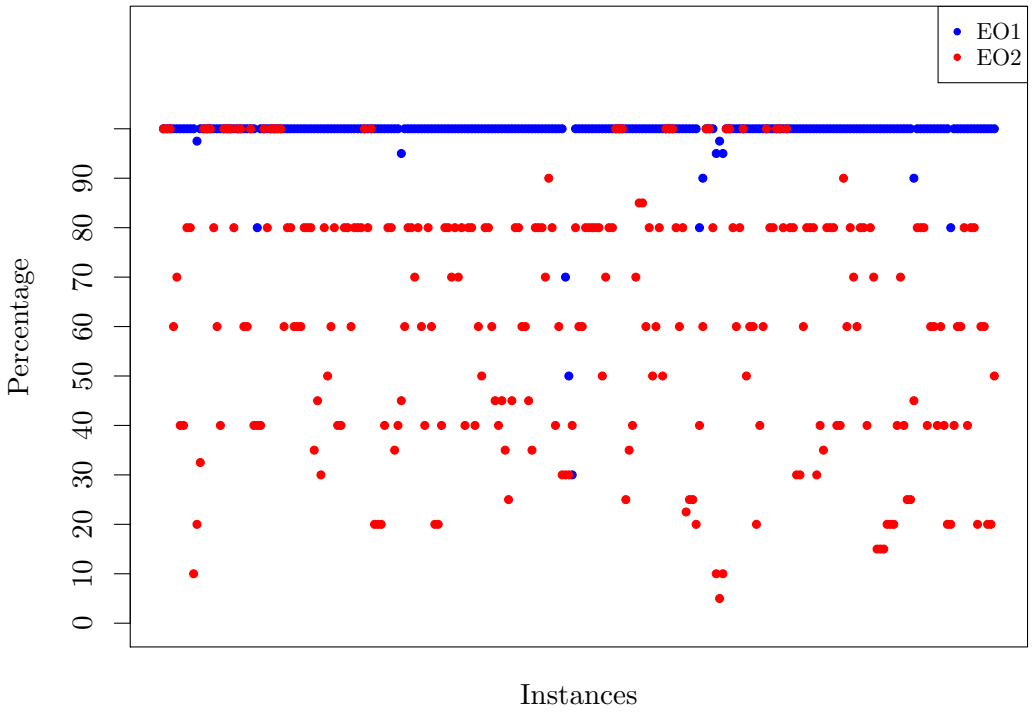

Figure 6: Percentage of service days in solution where cost was improved using PS

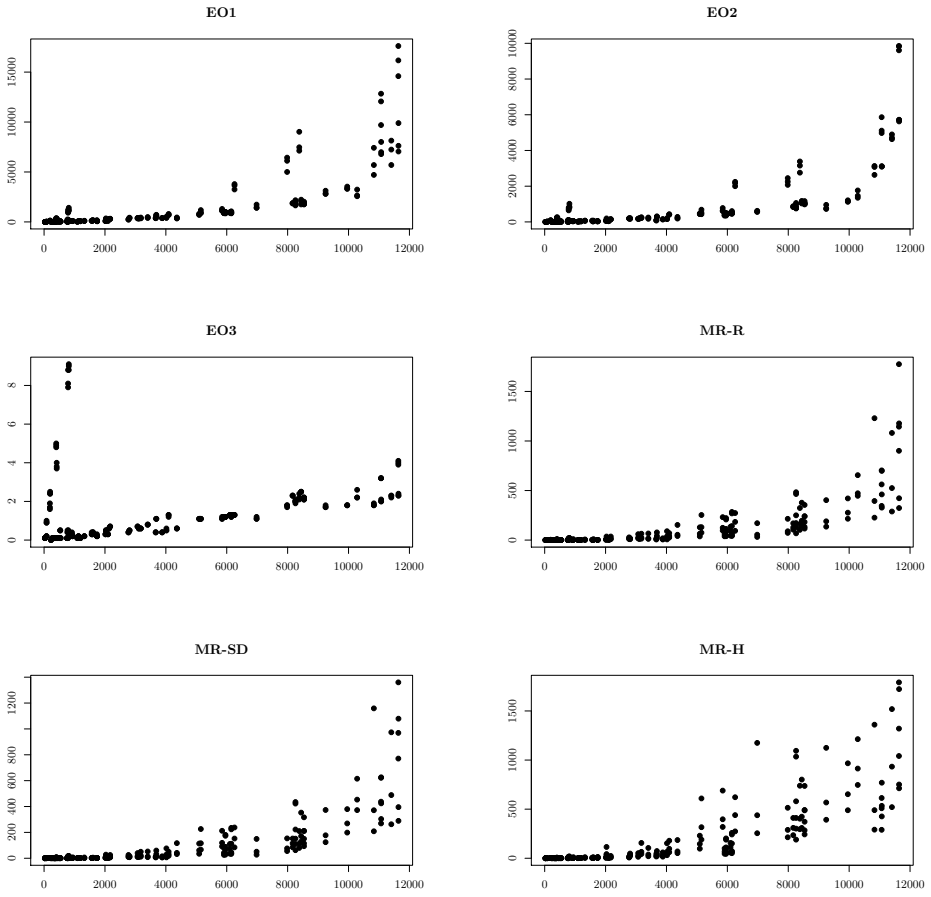

Figure 7: Runtime in seconds for each algorithm plotted against number of nodes in graph.

6 Concluding remarks

Two different types of construction heuristics for the SP-CARP have been presented. Both of them contain several components that can be implemented in different ways, which resulted in a set of algorithms that was tested on 15 instances. These initial tests showed that some combinations of the components performed poorly regarding runtime, number of vehicles used for routing, and total routing costs. Six algorithms were chosen that performed well on at least one of the three parameters mentioned above. These six algorithms were used to find solutions to an additional 243 instances. The solutions were analyzed in order to compare the performance of these six algorithms.

References

[1] José-Manuel Belenguer, Enrique Benavent, Philippe Lacomme, and Christian Prins. Lower and upper bounds for the mixed capacitated arc routing problem.

Computers and Operations Research, 33(12):3363–3383, 2006.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[3] Cleverson Gonçalves Dos Santos, Cassius Tadeu Scarpin, and Fausto Pinheiro Da Silva. Mathematical modeling for periodic capacitated arc routing problem. *International Journal of Engineering*, 8(02):8269, 2016.

[4] Bruce L Golden, James S DeArmon, and Edward K Baker. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10(1):47–59, 1983.

[5] Lone Kiilerich and Sanne Whlk. New large-scale data instances for carp and new variations of carp. *INFOR: Information Systems and Operational Research*, 56(1):1–32, 2018.

[6] Leon Y. O. Li and Richard W. Eglese. An interactive algorithm for vehicle routeing for winter-gritting. *Journal of the Operational Research Society*, 47(2):217–228, Feb 1996.

[7] Luís Santos, João Coutinho-Rodrigues, and John R Current. An improved heuristic for the capacitated arc routing problem. *Computers and Operations Research*, 36(9):2632–2637, 2009.

[8] Sanne Wøhlk. *Contributions to arc routing*. Citeseer, 2006.

A Runtime analysis

Runtime plotted against required edges and number of service days to see if any relation exists.

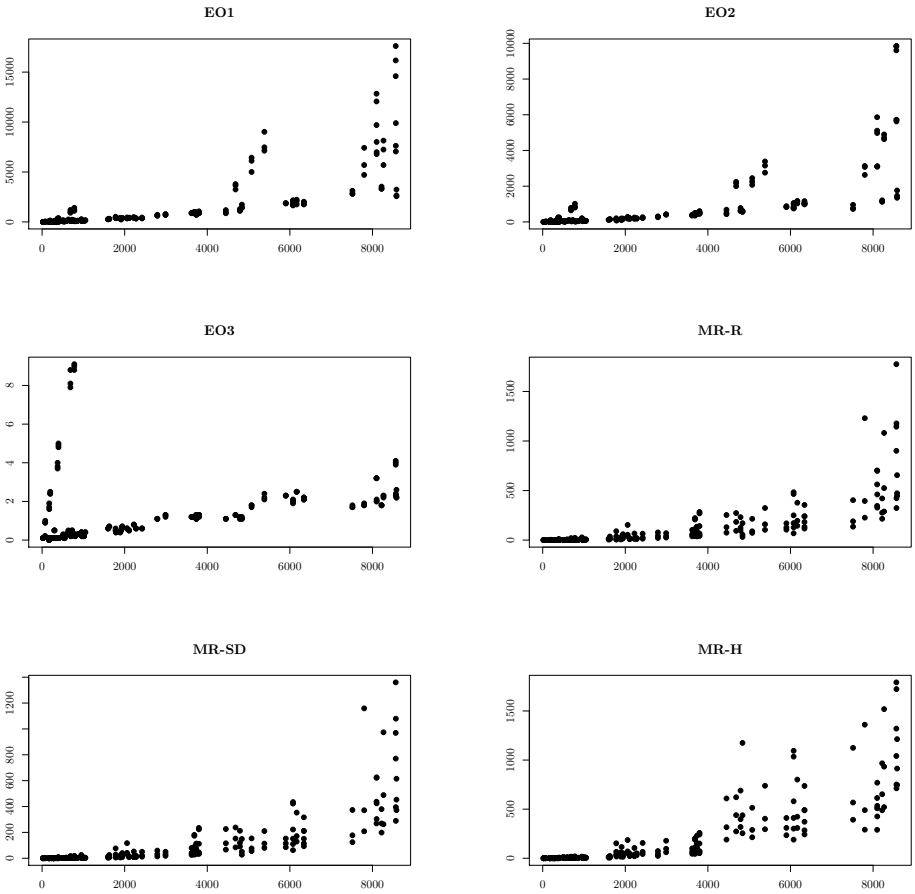

Figure 8: Runtime in seconds for each algorithm plotted against number of required edges in graph.

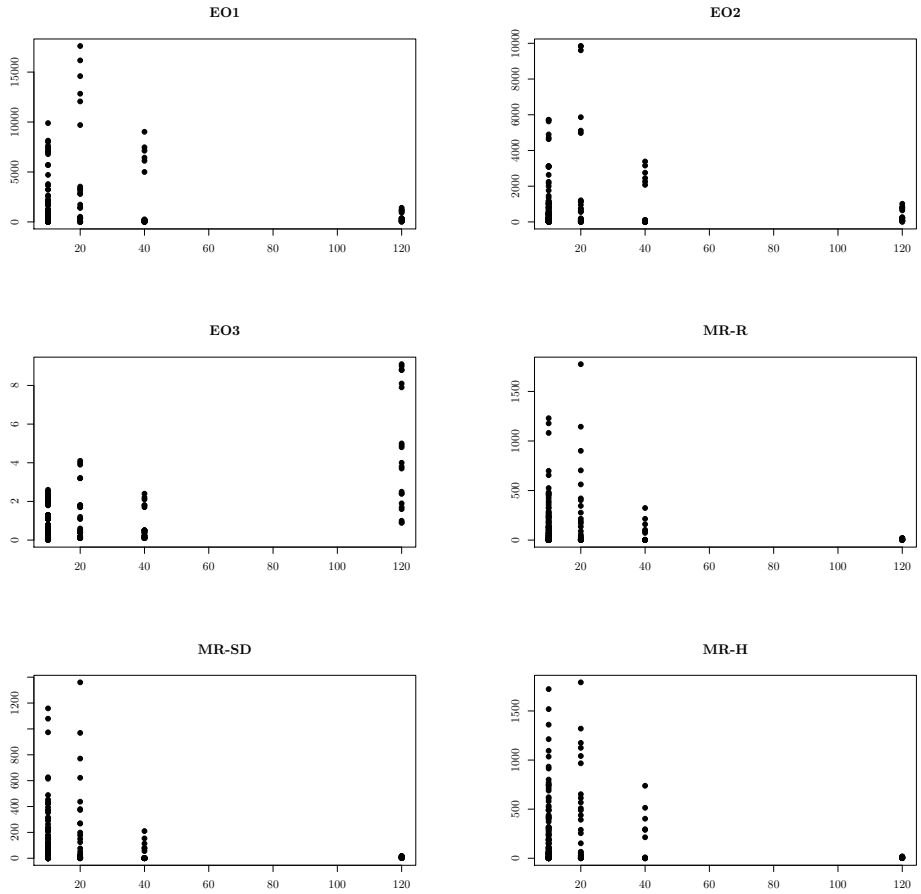

Figure 9: Runtime in seconds for each algorithm plotted against number of service days.

B Performance of the six algorithms with regard to runtime, number of vehicles, and cost

This appendix includes the performance of each of the six final algorithms regarding runtime, number vehicles needed to do the routing, and the total cost of routing. The results are listed for each of the 249 instances that we used to test our algorithms. The first subsections shows the runtime, the second section shows the number of vehicles, and the last section shows the total cost.

B.1 Runtime

The tables in this section show the runtime for the algorithms E1, E2, E3, MR-R, MR-SD, and MH-H. All of the algorithms were set to make 10 iterations and return the best solution.

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
F1_g	6-24	1,107.8	801.6	8.8	17.3	14.4	14.7
F1_g	5-24	1,208.5	852.0	9.0	18.9	16.0	16.1
F1_g	4-24	1,416.4	1,017.0	9.1	21.6	18.4	18.1
F1_p	6-8	174.4	82.1	0.5	0.9	0.8	2.0
F1_p	4-8	196.5	89.1	0.4	1.5	1.2	3.0
F1_p	2-8	224.9	102.0	0.5	2.6	2.3	5.3
F6_g	6-8	65.3	30.1	0.5	0.6	0.6	0.8
F6_g	4-8	65.4	39.8	0.5	0.6	0.6	0.9
F6_g	2-8	80.3	36.0	0.5	1.0	0.8	1.3
F6_p	3-8	26.2	7.2	0.1	0.2	0.2	0.4
F6_p	2-8	26.5	9.0	0.1	0.2	0.2	0.4
F6_p	1-8	26.5	10.1	0.1	0.2	0.2	0.6
F9_g	6-24	919.7	645.4	7.9	15.0	13.0	13.5
F9_g	5-24	997.1	709.4	8.1	16.4	14.0	14.6
F9_g	4-24	1,195.5	790.3	8.8	18.5	16.2	16.5
F9_p	5-8	154.5	79.4	0.4	0.8	0.8	2.0
F9_p	4-8	171.4	77.2	0.4	1.2	1.1	2.5
F9_p	1-8	249.3	109.7	0.5	4.5	3.2	8.2
F10_g	6-24	305.4	208.1	3.7	4.9	4.5	4.1
F10_g	5-24	322.8	220.9	3.8	5.1	4.7	4.2
F10_g	4-24	342.3	257.3	4.0	5.7	5.1	4.3
F11_g	6-24	81.9	56.2	1.6	1.8	1.7	1.7
F11_g	4-24	86.0	63.3	1.7	2.0	1.7	1.8
F11_g	2-24	102.0	73.8	1.9	2.3	2.1	1.9
F12_g	6-8	8.3	5.2	0.2	0.2	0.2	0.2
F12_g	4-8	8.3	5.2	0.2	0.2	0.2	0.2
F12_g	2-8	8.8	6.0	0.2	0.3	0.3	0.3
F13_g	4-8	1.3	0.5	0.1	0.1	0.1	0.1
F13_g	3-8	1.3	0.6	0.1	0.1	0.1	0.1
F13_g	2-8	1.3	0.7	0.1	0.1	0.1	0.1
F14_g	6-24	319.9	197.8	4.8	6.9	6.1	5.3
F14_g	5-24	341.7	219.6	4.9	7.2	6.4	5.4
F14_g	4-24	371.8	254.8	5.0	8.0	7.0	5.8
F15_g	6-24	123.1	84.4	2.4	2.8	2.6	2.6
F15_g	5-24	124.4	81.6	2.4	2.9	2.7	2.7
F15_g	4-24	131.9	88.2	2.5	3.1	2.8	2.6
F16_g	6-24	29.7	19.0	0.9	0.9	0.9	0.9
F16_g	5-24	30.7	19.0	0.9	0.9	0.9	0.9
F16_g	4-24	31.3	18.0	1.0	1.0	1.0	1.0
F17_g	5-4	1.0	0.5	0.1	0.1	0.1	0.1
F17_g	3-4	1.0	0.7	0.1	0.1	0.1	0.1
F17_g	2-4	1.1	0.7	0.1	0.1	0.1	0.1

Table 10: Runtime for the six algorithms on area F

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
N1_g	6-2	1,757.3	1,064.2	2.1	116.0	93.2	242.6
N1_g	3-2	1,872.4	1,016.5	2.1	240.2	211.6	487.7
N1_g	2-2	2,025.7	1,164.9	2.1	354.6	316.5	736.6
N1_p	5-4	1,427.1	566.9	1.1	31.7	27.4	254.9
N1_p	3-4	1,410.3	568.5	1.2	54.5	50.8	438.0
N1_p	1-4	1,732.7	622.3	1.1	170.5	149.5	1,174.8
N2_g	5-2	1,759.6	989.9	2.2	132.7	114.9	284.6
N2_g	4-2	1,855.6	982.3	2.2	182.4	150.8	372.0
N2_g	3-2	1,898.0	1,047.6	2.2	239.8	210.8	491.3
N3_g	3-2	1,814.9	782.2	2.0	171.6	151.7	410.7
N3_g	2-2	1,856.5	836.4	2.1	250.0	222.4	580.0
N3_g	1-2	2,169.5	983.2	2.0	482.9	434.5	1,094.5
N4_g	6-2	1,650.3	751.0	2.0	68.2	62.4	189.6
N4_g	4-2	1,715.9	775.1	1.9	127.8	110.3	301.7
N4_g	1-2	2,110.3	1,059.9	2.0	464.9	422.7	1,034.8
N5_g	5-2	1,749.6	1,034.4	2.5	144.4	129.1	308.7
N5_g	4-2	2,063.2	1,167.7	2.5	194.8	171.3	423.4
N5_g	2-2	2,225.6	1,152.7	2.5	377.3	352.9	801.3
N6_g	5-2	1,863.1	879.6	2.3	104.9	85.1	234.8
N6_g	4-2	1,853.6	830.2	2.3	127.6	115.5	309.6
N6_g	3-2	1,878.1	849.2	2.3	169.3	152.6	410.6
N7_g	3-2	132.9	30.7	0.3	2.0	2.0	3.8
N7_g	2-2	131.1	31.1	0.3	2.0	2.0	5.5
N7_g	1-2	138.5	31.9	0.3	3.2	3.0	9.8
N9_g	6-2	85.7	55.0	0.3	2.5	2.5	4.0
N9_g	3-2	89.3	59.2	0.3	4.8	3.8	7.5
N9_g	2-2	91.3	66.6	0.3	6.8	5.7	11.4
N9_p	5-4	67.5	37.9	0.2	1.9	1.9	3.0
N9_p	2-4	67.8	36.1	0.2	1.9	1.9	8.2
N9_p	1-4	68.3	38.0	0.2	2.2	1.9	13.3
N10_g	5-2	610.9	260.1	1.1	21.1	14.0	23.7
N10_g	2-2	648.2	297.8	1.1	42.4	31.7	43.8
N10_g	1-2	713.1	306.5	1.1	77.6	59.6	73.4
N11_g	4-2	253.4	123.7	0.6	4.5	4.1	7.2
N11_g	2-2	267.7	119.2	0.6	8.8	7.3	10.1
N11_g	1-2	283.5	121.3	0.6	16.3	13.7	15.7
N12_g	6-2	77.0	30.3	0.2	1.1	1.1	1.3
N12_g	2-2	76.0	39.2	0.2	1.3	1.1	1.9
N12_g	1-2	80.8	39.9	0.2	2.3	1.8	3.0
N13_g	6-2	26.4	6.3	0.1	0.4	0.3	0.3
N13_g	3-2	25.7	7.8	0.1	0.4	0.3	0.4
N13_g	1-2	26.4	10.8	0.1	0.5	0.4	0.8

Table 11: Runtime for the six algorithms on area N

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
N14_g	6-2	691.7	406.7	1.3	25.3	18.7	61.2
N14_g	4-2	722.1	419.4	1.3	39.1	30.9	96.4
N14_g	2-2	793.0	423.4	1.2	69.9	50.5	178.7
N15_g	5-2	284.5	152.5	0.7	9.9	6.3	6.5
N15_g	3-2	299.2	142.1	0.7	13.7	9.3	11.7
N15_g	1-2	323.6	140.7	0.7	36.1	25.2	23.5
N16_g	6-2	98.4	45.0	0.3	1.1	1.1	1.5
N16_g	4-2	94.9	43.3	0.4	1.2	1.2	1.3
N16_g	1-2	104.5	45.6	0.3	5.9	3.5	3.6
N17_g	5-2	13.0	5.0	0.1	0.2	0.2	0.3
N17_g	3-2	13.3	5.8	0.1	0.2	0.2	0.2
N17_g	1-2	13.0	4.9	0.1	0.4	0.3	0.3

Table 12: Runtime for the six algorithms on instances from area N (continued)

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
O1_g	6-2	3,238.3	1,762.1	2.6	446.7	372.3	746.5
O1_g	5-2	2,569.3	1,351.1	2.2	472.4	453.3	913.5
O1_g	4-2	2,648.3	1,461.9	2.2	655.1	615.0	1,212.9
O1_p	4-4	3,318.8	1,129.1	1.8	215.3	198.5	488.9
O1_p	3-4	3,328.4	1,144.2	1.8	276.7	269.2	652.1
O1_p	2-4	3,536.3	1,209.4	1.8	420.4	379.8	967.0
O6_p	3-4	2,798.4	728.3	1.7	135.4	124.2	392.0
O6_p	2-4	2,843.1	753.3	1.7	189.6	177.5	567.9
O6_p	1-4	3,123.6	955.5	1.8	403.0	373.4	1,124.5
O9_g	6-2	352.4	191.8	0.6	41.0	32.3	52.0
O9_g	5-2	356.8	201.4	0.6	54.0	41.6	69.8
O9_g	2-2	446.4	280.9	0.6	152.8	116.8	185.4
O9_p	6-4	412.1	159.1	0.5	6.9	6.9	31.0
O9_p	3-4	422.1	158.7	0.5	37.5	28.7	66.0
O9_p	1-4	519.7	202.9	0.6	89.5	76.3	152.5
O10_g	6-2	1,102.0	605.9	1.1	96.3	91.0	318.9
O10_g	5-2	1,156.0	630.3	1.1	123.9	119.3	396.9
O10_g	3-2	1,298.3	776.7	1.2	231.3	212.4	688.9
O11_g	6-2	376.9	168.0	0.5	8.7	8.0	22.0
O11_g	4-2	401.2	193.9	0.5	13.3	12.2	35.4
O11_g	3-2	424.7	211.1	0.5	18.3	15.7	48.0
O12_g	6-2	44.4	17.1	0.1	0.6	0.5	0.8
O12_g	2-2	45.8	20.1	0.1	0.8	0.7	1.5
O12_g	1-2	47.1	19.8	0.1	1.5	1.2	3.0
O13_g	5-2	6.0	1.6	0.0	0.1	0.1	0.1
O13_g	2-2	6.0	1.8	0.0	0.1	0.1	0.2
O13_g	1-2	6.1	2.3	0.0	0.1	0.1	0.3
O14_g	6-2	867.6	431.2	1.1	74.6	66.7	189.9
O14_g	4-2	947.6	517.4	1.1	130.3	115.9	316.2
O14_g	2-2	1,183.0	685.1	1.1	253.1	226.0	609.2
O15_g	6-2	246.6	110.5	0.5	10.4	8.6	33.3
O15_g	5-2	255.7	123.9	0.5	12.6	10.9	42.4
O15_g	2-2	351.9	195.1	0.5	35.9	27.9	115.0
O16_g	6-2	86.2	38.4	0.2	2.4	2.1	3.2
O16_g	4-2	90.8	47.6	0.2	4.2	3.7	5.5
O16_g	2-2	107.1	53.6	0.2	8.5	6.7	10.4
O17_g	2-2	18.2	8.4	0.1	0.7	0.6	0.8
O17_g	3-2	17.3	7.3	0.1	0.5	0.4	0.6
O17_g	2-2	18.1	7.4	0.1	0.7	0.6	0.8

Table 13: Runtime for the six algorithms on instances from area O

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
K1_g	6-2	7,048.1	5,721.9	2.4	323.5	289.3	712.3
K1_g	5-2	7,620.7	5,712.8	2.3	422.2	395.3	751.0
K1_g	2-2	9,892.7	5,626.1	2.3	1,177.5	1,079.2	1,722.6
K1_p	3-8	7,127.5	2,757.5	2.2	103.3	80.0	295.6
K1_p	2-8	7,472.4	3,153.5	2.1	160.0	113.6	402.8
K1_p	1-8	9,023.4	3,385.0	2.4	323.7	210.2	738.4
K2_g	3-4	16,170.1	9,607.8	3.9	1,144.2	969.3	1,320.3
K2_g	4-4	14,593.4	9,849.0	4.1	900.9	770.7	1,041.3
K2_g	2-4	17,610.8	9,835.4	4.0	1,775.1	1,360.0	1,791.3
K3_g	3-2	6,983.6	3,106.9	2.1	460.9	424.4	533.6
K3_g	4-2	6,787.4	3,116.9	2.0	327.6	303.5	425.9
K3_g	2-2	8,006.6	3,095.7	2.0	698.4	625.9	769.5
K4_g	6-4	9,697.6	5,863.0	3.2	344.9	269.2	289.7
K4_g	4-4	12,063.1	4,976.9	3.2	561.9	437.4	509.0
K4_g	3-4	12,836.4	5,110.9	3.2	703.4	622.0	613.8
K5_g	6-2	5,697.7	4,633.5	2.2	288.4	262.8	520.6
K5_g	4-2	7,242.8	4,711.7	2.3	524.9	488.5	932.9
K5_g	2-2	8,138.9	4,898.8	2.3	1,081.3	973.7	1,519.0
K6_g	5-2	4,706.4	2,633.4	1.8	226.4	208.6	291.2
K6_g	3-2	5,695.2	3,077.3	1.8	394.4	371.2	489.8
K6_g	1-2	7,419.3	3,131.6	1.9	1,229.8	1,158.9	1,360.3
K6_p	3-8	5,001.3	2,074.0	1.7	72.8	55.6	213.5
K6_p	2-8	6,118.0	2,255.9	1.8	89.0	76.2	289.0
K6_p	1-8	6,435.1	2,452.2	1.8	214.7	153.7	514.0
K7_g	5-2	70.1	28.8	0.1	1.0	1.0	1.4
K7_g	2-2	69.7	32.4	0.1	1.0	1.0	2.3
K7_g	1-2	68.8	34.9	0.1	1.9	1.8	3.6
K9_g	5-2	244.5	193.5	0.4	10.8	9.3	8.0
K9_g	4-2	233.8	193.0	0.4	15.9	13.7	9.6
K9_g	2-2	260.0	192.9	0.4	27.5	25.4	17.2
K9_p	3-8	151.7	31.4	0.2	1.9	2.0	3.8
K9_p	2-8	149.8	29.8	0.2	1.9	2.1	4.5
K9_p	1-8	138.1	40.3	0.3	2.0	2.1	6.4
K10_g	6-2	701.4	440.6	1.1	37.1	34.6	97.5
K10_g	4-2	746.0	443.7	1.1	64.2	61.1	145.9
K10_g	2-2	823.0	476.4	1.1	129.0	113.7	229.9
K11_g	6-2	344.8	191.6	0.6	11.1	9.9	31.1
K11_g	4-2	347.5	189.8	0.6	15.7	13.5	45.6
K11_g	3-2	353.3	206.0	0.6	20.2	18.6	50.3
K12_g	4-2	68.6	29.5	0.2	1.3	1.2	1.7
K12_g	2-2	69.7	33.0	0.2	1.3	1.3	2.1
K12_g	1-2	70.3	29.5	0.2	2.4	2.1	3.2

Table 14: Runtime for the six algorithms on instances from area K

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
K13_g	3-2	13.7	4.7	0.1	0.2	0.2	0.2
K13_g	2-2	13.9	4.2	0.1	0.2	0.2	0.3
K13_g	1-2	14.1	6.0	0.1	0.2	0.2	0.4
K14_g	5-2	3,247.7	2,172.2	1.3	93.6	84.3	273.6
K14_g	3-2	3,649.7	2,244.5	1.3	183.7	152.5	439.3
K14_g	2-2	3,784.9	1,999.2	1.3	272.9	237.9	621.5
K15_g	5-2	364.7	243.9	0.6	14.0	12.1	43.1
K15_g	3-2	363.5	213.0	0.6	23.9	20.7	65.6
K15_g	1-2	437.3	258.3	0.6	64.5	50.4	155.9
K16_g	6-2	100.8	67.9	0.2	1.6	1.6	4.3
K16_g	4-2	96.4	71.1	0.2	2.0	1.7	4.2
K16_g	1-2	116.4	60.6	0.2	6.6	6.0	10.8
K17_g	3-2	14.1	4.5	0.1	0.2	0.2	0.2
K17_g	2-2	13.9	5.1	0.1	0.2	0.2	0.3
K17_g	1-2	13.6	6.1	0.1	0.2	0.2	0.4

Table 15: Runtime for the six algorithms on instances from area K (continued)

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
S1_g	6-2	856.6	474.8	1.3	42.2	36.2	65.2
S1_g	5-2	885.5	454.4	1.2	66.9	46.0	72.6
S1_g	1-2	1,033.8	526.3	1.2	284.0	234.8	258.4
S1_p	4-4	386.9	146.9	0.4	6.8	6.7	16.4
S1_p	2-4	373.3	150.4	0.4	8.2	7.0	29.2
S1_p	1-4	390.9	135.5	0.4	18.0	15.1	59.7
S2_g	6-2	902.0	531.6	1.3	45.9	34.5	51.7
S2_g	2-2	938.1	475.4	1.3	141.5	112.2	150.1
S2_g	1-2	1,048.9	605.8	1.3	269.1	223.6	242.0
S3_g	5-2	850.9	352.7	1.2	43.6	35.0	55.5
S3_g	3-2	894.8	377.6	1.2	82.3	64.3	86.7
S3_g	1-2	998.5	472.2	1.2	221.8	180.6	201.9
S4_g	6-2	881.8	436.9	1.2	39.1	27.7	44.6
S4_g	2-2	909.1	506.1	1.2	112.7	88.3	110.6
S4_g	1-2	997.6	457.8	1.2	207.6	173.9	189.2
S5_g	6-2	917.6	506.3	1.3	44.7	35.3	58.5
S5_g	3-2	942.4	492.9	1.3	93.9	74.7	113.2
S5_g	2-2	963.3	482.1	1.3	135.2	110.5	155.8
S6_g	6-2	874.0	352.4	1.2	38.1	26.0	45.8
S6_g	4-2	870.5	376.0	1.2	58.4	41.8	67.7
S6_g	2-2	906.2	397.6	1.2	102.7	79.4	100.3
S6_p	3-4	373.3	99.9	0.4	7.3	7.3	17.8
S6_p	2-4	370.0	82.3	0.4	7.4	7.2	25.6
S6_p	1-4	386.7	83.7	0.4	15.2	13.2	45.4
S7_g	4-2	118.6	40.1	0.3	2.6	2.6	3.8
S7_g	2-2	118.1	40.4	0.3	2.7	2.6	4.1
S7_g	1-2	118.9	41.0	0.3	3.9	3.8	5.9
S9_g	4-2	19.2	11.8	0.1	0.6	0.6	0.6
S9_g	2-2	18.9	12.3	0.1	0.7	0.6	0.9
S9_g	1-2	20.5	12.4	0.1	1.7	1.4	1.5
S9_p	3-4	10.7	5.6	0.1	0.3	0.3	0.4
S9_p	2-4	10.6	6.0	0.1	0.3	0.3	0.4
S9_p	1-4	10.7	5.6	0.1	0.3	0.3	0.6
S10_g	6-2	412.5	173.8	0.8	12.9	8.5	20.3
S10_g	4-2	411.8	221.6	0.8	16.2	13.7	37.8
S10_g	1-2	488.8	246.3	0.8	66.8	52.5	104.9
S11_g	5-2	118.7	75.9	0.4	2.1	2.1	2.9
S11_g	3-2	115.2	57.4	0.3	2.0	1.9	3.6
S11_g	2-2	116.6	57.7	0.3	4.0	2.6	4.8
S12_g	5-2	33.7	20.8	0.1	0.6	0.6	0.6
S12_g	3-2	34.1	20.9	0.1	0.6	0.6	0.6
S12_g	1-2	34.9	17.7	0.1	0.7	0.6	1.1

Table 16: Runtime for the siz algorithms on instances from area S

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
S13_g	3-2	9.0	2.0	0.1	0.2	0.1	0.1
S13_g	2-2	9.0	2.7	0.1	0.2	0.1	0.2
S13_g	1-2	9.0	2.5	0.1	0.2	0.2	0.2
S14_g	5-2	353.8	168.1	0.7	16.4	9.4	14.9
S14_g	3-2	364.1	172.1	0.7	24.1	15.2	25.9
S14_g	1-2	400.0	183.8	0.7	59.6	40.0	51.7
S15_g	6-2	152.4	68.0	0.4	2.1	2.1	2.8
S15_g	3-2	155.7	58.3	0.4	5.9	3.3	5.0
S15_g	2-2	154.0	60.9	0.4	7.6	4.8	7.6
S16_g	5-2	32.0	7.5	0.1	0.4	0.4	0.4
S16_g	2-2	32.1	10.3	0.1	0.4	0.4	0.6
S16_g	1-2	31.9	11.7	0.1	0.7	0.5	0.8
S17_g	3-2	13.6	3.6	0.1	0.2	0.2	0.2
S17_g	2-2	13.5	2.4	0.1	0.2	0.2	0.2
S17_g	1-2	13.5	3.4	0.1	0.2	0.2	0.3

Table 17: Runtime for the six algorithms on instances from area S (continued)

B.2 Vehicles

The tables in this section show the number of vehicles used in the best found solution for the algorithms E1, E2, E3, MR-R, MR-SD, and MH-H.

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
F1_g	6-24	7	11	12	9	7	14
F1_g	5-24	8	17	19	12	9	17
F1_g	4-24	11	24	21	16	12	23
F1_p	6-8	1	2	2	1	1	2
F1_p	4-8	2	4	3	2	2	2
F1_p	2-8	3	9	7	3	3	4
F6_g	6-8	1	2	2	1	1	1
F6_g	4-8	1	2	2	1	1	3
F6_g	2-8	2	4	4	2	2	5
F6_p	3-8	1	2	2	1	1	1
F6_p	2-8	1	2	2	1	1	2
F6_p	1-8	1	2	3	1	1	1
F9_g	6-24	6	10	12	9	6	12
F9_g	5-24	8	14	14	12	8	16
F9_g	4-24	11	20	22	16	11	22
F9_p	5-8	2	2	2	1	1	2
F9_p	4-8	2	3	4	2	2	3
F9_p	1-8	5	18	17	5	5	10
F10_g	6-24	3	5	5	4	3	5
F10_g	5-24	4	7	7	5	4	7
F10_g	4-24	5	9	10	7	5	9
F11_g	6-24	2	3	3	2	2	3
F11_g	4-24	3	4	4	3	2	5
F11_g	2-24	5	7	9	6	5	8
F12_g	6-8	1	2	1	1	1	2
F12_g	4-8	1	2	2	2	1	2
F12_g	2-8	2	3	3	3	2	4
F13_g	4-8	1	1	1	1	1	1
F13_g	3-8	1	1	1	1	1	1
F13_g	2-8	1	2	2	1	1	1
F14_g	6-24	4	8	7	6	4	8
F14_g	5-24	5	10	8	7	5	10
F14_g	4-24	7	12	12	10	7	14
F15_g	6-24	3	4	4	3	2	4
F15_g	5-24	3	5	5	4	3	6
F15_g	4-24	4	6	7	5	4	8
F16_g	6-24	1	2	2	2	1	2
F16_g	5-24	2	2	2	2	1	2
F16_g	4-24	2	3	2	2	2	3
F17_g	5-4	1	1	1	1	1	1
F17_g	3-4	1	2	1	1	1	2
F17_g	2-4	2	2	2	2	2	2

Table 18: Number of vehicles needed in the solutions found by the six algorithms on instances from area F

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
N1_g	6-2	3	5	5	3	3	6
N1_g	3-2	6	13	11	6	6	14
N1_g	2-2	9	16	32	9	9	18
N1_p	5-4	2	9	6	2	2	5
N1_p	3-4	3	12	14	2	2	7
N1_p	1-4	6	44	45	6	6	23
N2_g	5-2	3	9	9	4	3	7
N2_g	4-2	5	15	13	5	5	7
N2_g	3-2	6	14	17	6	6	10
N3_g	3-2	5	9	13	5	5	8
N3_g	2-2	7	21	16	7	7	8
N3_g	1-2	14	43	31	14	14	19
N4_g	6-2	2	5	6	2	2	2
N4_g	4-2	4	8	7	4	4	4
N4_g	1-2	13	33	37	13	13	17
N5_g	5-2	3	6	5	3	3	5
N5_g	4-2	4	8	8	4	4	10
N5_g	2-2	8	20	22	8	8	19
N6_g	5-2	3	4	5	3	3	4
N6_g	4-2	3	8	7	3	3	5
N6_g	3-2	4	11	12	4	4	6
N7_g	3-2	1	3	3	1	1	1
N7_g	2-2	1	5	5	1	1	2
N7_g	1-2	2	9	9	2	2	3
N9_g	6-2	1	2	2	1	1	2
N9_g	3-2	2	3	3	2	2	4
N9_g	2-2	2	4	4	2	2	4
N9_p	5-4	1	1	1	1	1	1
N9_p	2-4	1	3	2	1	1	2
N9_p	1-4	2	5	5	1	1	4
N10_g	5-2	2	4	4	2	2	3
N10_g	2-2	4	6	14	4	4	7
N10_g	1-2	7	13	17	7	7	21
N11_g	4-2	2	2	3	1	1	2
N11_g	2-2	3	4	5	2	2	5
N11_g	1-2	4	15	7	4	4	7
N12_g	6-2	1	2	1	1	1	1
N12_g	2-2	2	2	3	1	1	2
N12_g	1-2	3	5	5	2	2	5
N13_g	6-2	1	1	1	1	1	1
N13_g	3-2	1	2	2	1	1	1
N13_g	1-2	2	4	3	2	2	2

Table 19: Number of vehicles needed in the solutions found by the six algorithms on instances from area N

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
N14_g	6-2	2	3	3	2	2	4
N14_g	4-2	3	4	5	3	3	4
N14_g	2-2	5	14	11	5	5	12
N15_g	5-2	2	3	3	2	2	3
N15_g	3-2	3	6	5	3	3	5
N15_g	1-2	7	12	14	7	7	13
N16_g	6-2	1	2	1	1	1	2
N16_g	4-2	2	2	2	1	1	2
N16_g	1-2	4	8	7	4	4	6
N17_g	5-2	1	1	1	1	1	1
N17_g	3-2	1	2	2	1	1	2
N17_g	1-2	2	4	3	2	2	4

Table 20: Number of vehicles needed in the solutions found by the six algorithms on instances from area N (continued)

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
O1_g	6-2	8	17	24	8	8	19
O1_g	5-2	10	36	31	10	10	25
O1_g	4-2	13	43	49	13	13	39
O1_p	4-4	4	16	21	4	4	10
O1_p	3-4	5	24	26	5	5	11
O1_p	2-4	7	34	32	7	7	20
O6_p	3-4	3	27	15	3	3	7
O6_p	2-4	5	34	21	4	4	9
O6_p	1-4	8	76	70	8	8	30
O9_g	6-2	4	9	6	4	4	7
O9_g	5-2	5	10	11	4	4	9
O9_g	2-2	11	37	31	11	11	26
O9_p	6-4	1	3	4	1	1	2
O9_p	3-4	2	10	11	2	2	5
O9_p	1-4	6	25	30	6	6	18
O10_g	6-2	6	9	12	5	5	14
O10_g	5-2	7	20	22	7	7	15
O10_g	3-2	12	33	37	12	12	23
O11_g	6-2	2	5	4	2	2	4
O11_g	4-2	4	8	11	4	4	7
O11_g	3-2	5	10	16	5	5	10
O12_g	6-2	1	2	2	1	1	2
O12_g	2-2	2	4	3	2	2	2
O12_g	1-2	3	12	7	3	3	4
O13_g	5-2	1	1	1	1	1	1
O13_g	2-2	1	3	2	1	1	2
O13_g	1-2	1	5	5	1	1	2
O14_g	6-2	6	9	9	5	5	15
O14_g	4-2	9	26	34	9	9	25
O14_g	2-2	18	45	75	18	18	39
O15_g	6-2	3	8	8	3	3	7
O15_g	5-2	4	10	8	4	4	7
O15_g	2-2	10	20	21	10	10	28
O16_g	6-2	2	5	4	2	2	6
O16_g	4-2	4	7	7	4	4	13
O16_g	2-2	7	23	18	7	7	11
O17_g	2-2	3	6	7	3	3	5
O17_g	3-2	2	5	4	2	2	4
O17_g	2-2	3	7	9	3	3	6

Table 21: Number of vehicles needed in the solutions found by the six algorithms on instances from area O

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
K1_g	6-2	5	13	10	5	5	12
K1_g	5-2	6	14	16	6	6	16
K1_g	2-2	15	34	35	15	15	37
K1_p	3-8	2	11	10	2	2	5
K1_p	2-8	3	13	14	3	3	10
K1_p	1-8	5	30	38	5	5	19
K2_g	3-4	10	27	25	10	10	29
K2_g	4-4	8	19	18	7	7	23
K2_g	2-4	14	39	67	14	14	37
K3_g	3-2	7	21	18	7	7	11
K3_g	4-2	5	15	13	5	5	8
K3_g	2-2	10	36	25	10	10	16
K4_g	6-4	3	10	6	3	3	4
K4_g	4-4	5	17	18	5	5	7
K4_g	3-4	7	25	20	7	7	10
K5_g	6-2	4	9	11	4	4	13
K5_g	4-2	7	18	21	7	7	20
K5_g	2-2	14	35	38	14	14	36
K6_g	5-2	4	10	10	4	4	5
K6_g	3-2	7	18	23	6	6	10
K6_g	1-2	19	46	58	18	18	30
K6_p	3-8	2	12	7	2	2	4
K6_p	2-8	3	12	11	2	2	8
K6_p	1-8	4	22	25	4	4	15
K7_g	5-2	1	2	2	1	1	1
K7_g	2-2	1	2	3	1	1	1
K7_g	1-2	2	5	4	2	2	2
K9_g	5-2	2	4	4	2	2	5
K9_g	4-2	3	5	4	3	3	6
K9_g	2-2	5	9	10	5	5	10
K9_p	3-8	1	5	7	1	1	2
K9_p	2-8	1	9	6	1	1	2
K9_p	1-8	2	11	15	1	1	5
K10_g	6-2	3	4	7	3	3	7
K10_g	4-2	4	7	15	4	4	11
K10_g	2-2	8	22	23	8	8	18
K11_g	6-2	2	4	3	2	2	3
K11_g	4-2	3	5	4	3	3	6
K11_g	3-2	3	7	7	3	3	7
K12_g	4-2	1	2	2	1	1	2
K12_g	2-2	1	2	3	1	1	2
K12_g	1-2	2	6	4	2	2	2

Table 22: Number of vehicles needed in the solutions found by the six algorithms on instances from area K

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
K13_g	3-2	1	2	2	1	1	1
K13_g	2-2	1	2	2	1	1	1
K13_g	1-2	1	2	3	1	1	2
K14_g	5-2	4	9	7	4	4	10
K14_g	3-2	6	17	20	6	6	16
K14_g	2-2	9	28	27	9	9	21
K15_g	5-2	2	3	3	2	2	6
K15_g	3-2	3	10	7	3	3	9
K15_g	1-2	9	25	25	9	9	21
K16_g	6-2	1	2	1	1	1	3
K16_g	4-2	2	2	2	2	2	3
K16_g	1-2	5	13	12	5	5	11
K17_g	3-2	1	1	2	1	1	1
K17_g	2-2	1	2	2	1	1	1
K17_g	1-2	1	2	2	1	1	2

Table 23: Number of vehicles needed in the solutions found by the six algorithms on instances from area K (continued)

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
S1_g	6-2	3	5	6	2	2	4
S1_g	5-2	3	6	6	3	3	5
S1_g	1-2	13	29	25	13	13	19
S1_p	4-4	1	3	4	1	1	2
S1_p	2-4	2	8	8	2	2	4
S1_p	1-4	3	11	16	3	3	12
S2_g	6-2	2	3	4	2	2	4
S2_g	2-2	7	13	17	7	7	11
S2_g	1-2	13	28	26	13	13	22
S3_g	5-2	3	6	5	2	2	4
S3_g	3-2	4	10	8	4	4	5
S3_g	1-2	11	29	28	11	11	13
S4_g	6-2	2	4	4	2	2	3
S4_g	2-2	6	9	15	6	5	8
S4_g	1-2	11	30	29	11	10	13
S5_g	6-2	2	3	4	2	2	4
S5_g	3-2	4	11	10	4	4	9
S5_g	2-2	6	14	15	6	6	9
S6_g	6-2	2	4	4	2	2	3
S6_g	4-2	3	5	6	3	3	3
S6_g	2-2	5	11	11	5	5	6
S6_p	3-4	1	10	10	1	1	2
S6_p	2-4	2	13	14	1	1	2
S6_p	1-4	2	30	28	2	2	5
S7_g	4-2	1	2	2	1	1	1
S7_g	2-2	1	4	4	1	1	1
S7_g	1-2	2	8	8	2	2	3
S9_g	4-2	1	2	2	1	1	2
S9_g	2-2	2	2	2	2	2	3
S9_g	1-2	3	4	4	3	3	4
S9_p	3-4	1	1	1	1	1	1
S9_p	2-4	1	2	1	1	1	1
S9_p	1-4	1	2	2	1	1	1
S10_g	6-2	2	3	3	2	2	3
S10_g	4-2	2	3	6	2	2	4
S10_g	1-2	8	18	15	8	8	11
S11_g	5-2	1	2	2	1	1	2
S11_g	3-2	2	3	3	1	1	2
S11_g	2-2	2	4	3	2	2	3
S12_g	5-2	1	1	1	1	1	1
S12_g	3-2	1	1	1	1	1	2
S12_g	1-2	2	3	3	2	2	2

Table 24: Number of vehicles needed in the solutions found by the six algorithms on instances from area S

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
S13_g	3-2	1	1	1	1	1	1
S13_g	2-2	1	2	2	1	1	1
S13_g	1-2	1	2	2	1	1	1
S14_g	5-2	2	3	4	2	2	3
S14_g	3-2	3	6	6	3	3	7
S14_g	1-2	8	16	16	8	8	12
S15_g	6-2	1	2	2	1	1	2
S15_g	3-2	2	3	3	2	2	4
S15_g	2-2	3	6	6	3	3	4
S16_g	5-2	1	2	2	1	1	1
S16_g	2-2	1	2	3	1	1	1
S16_g	1-2	2	5	4	2	2	2
S17_g	3-2	1	2	2	1	1	1
S17_g	2-2	1	2	2	1	1	1
S17_g	1-2	1	2	3	1	1	1

Table 25: Number of vehicles needed in the solutions found by the six algorithms on instances from area S (continued)

B.3 Cost

The tables in this section show the total routing cost for the best found solution for the algorithms E1, E2, E3, MR-R, MR-SD, and MH-H.

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
F1_g	6-24	12,872,700	12,163,100	17,218,600	19,286,800	16,800,900	16,272,000
F1_g	5-24	14,190,600	13,793,700	18,243,300	20,535,500	17,918,200	17,737,400
F1_g	4-24	16,619,600	15,610,100	20,972,700	24,726,100	20,250,500	21,561,100
F1_p	6-8	1,466,180	931,506	1,281,960	2,042,800	2,144,700	869,701
F1_p	4-8	1,652,150	1,108,360	1,426,130	2,195,780	2,297,300	1,008,110
F1_p	2-8	1,932,270	1,338,560	1,827,600	2,413,990	2,560,830	1,318,330
F6_g	6-8	1,225,410	857,242	1,443,190	1,494,430	1,486,890	780,036
F6_g	4-8	1,228,720	973,243	1,399,660	1,558,600	1,501,310	818,463
F6_g	2-8	1,474,630	1,077,370	1,467,830	1,734,260	1,763,360	1,091,170
F6_p	3-8	518,349	158,380	268,940	574,374	571,976	150,462
F6_p	2-8	515,602	185,124	274,501	566,206	575,362	171,088
F6_p	1-8	519,953	205,933	297,546	558,619	573,861	227,253
F9_g	6-24	12,261,500	11,657,800	16,268,700	17,328,500	15,561,300	15,840,500
F9_g	5-24	13,352,500	13,251,800	17,983,300	19,359,900	16,974,600	17,247,900
F9_g	4-24	15,701,400	15,060,100	19,898,000	23,263,700	19,745,700	20,914,400
F9_p	5-8	1,460,770	954,039	1,368,350	2,002,460	2,008,450	859,166
F9_p	4-8	1,623,690	991,220	1,401,760	2,096,930	2,305,060	947,221
F9_p	1-8	2,463,340	1,885,230	2,332,250	2,935,670	3,047,150	2,044,800
F10_g	6-24	4,694,970	4,200,770	6,272,080	6,642,430	6,353,500	4,827,490
F10_g	5-24	4,954,740	4,415,100	6,575,940	7,081,040	6,390,860	5,167,240
F10_g	4-24	5,263,170	4,872,180	6,871,640	7,504,150	6,973,340	5,551,180
F11_g	6-24	1,633,050	1,369,200	2,336,580	2,138,420	2,162,470	1,600,790
F11_g	4-24	1,674,420	1,565,990	2,481,360	2,365,600	2,260,830	1,792,700
F11_g	2-24	1,892,840	1,820,430	2,649,310	2,702,230	2,721,480	2,300,810
F12_g	6-8	233,609	191,537	292,989	284,720	288,344	206,231
F12_g	4-8	229,394	191,346	296,016	291,934	281,592	221,060
F12_g	2-8	241,288	213,384	314,580	316,026	309,610	230,720
F13_g	4-8	59,550	37,214	59,640	52,360	65,024	39,792
F13_g	3-8	65,344	37,184	60,382	60,192	62,264	39,792
F13_g	2-8	56,350	39,896	62,518	57,448	65,024	46,074
F14_g	6-24	5,060,780	4,759,440	7,518,880	7,473,010	7,086,700	5,555,980
F14_g	5-24	5,291,500	4,733,940	7,603,770	7,919,620	7,227,820	6,006,480
F14_g	4-24	5,743,750	5,395,940	8,074,600	8,389,700	7,859,560	6,779,960
F15_g	6-24	2,270,330	2,080,750	3,307,790	3,123,210	3,107,340	2,589,760
F15_g	5-24	2,353,890	2,117,160	3,279,610	3,229,340	3,211,330	2,487,130
F15_g	4-24	2,480,750	2,277,040	3,438,330	3,599,780	3,433,820	2,639,130
F16_g	6-24	799,891	638,231	1,000,990	918,456	961,584	678,594
F16_g	5-24	809,866	731,069	1,031,320	1,044,830	1,003,910	749,964
F16_g	4-24	847,287	704,768	1,065,330	954,696	996,990	691,398
F17_g	5-4	41,680	31,948	49,592	40,112	45,220	39,844
F17_g	3-4	39,920	35,903	56,205	45,280	47,104	42,576
F17_g	2-4	40,504	37,225	51,832	48,336	49,576	42,392

Table 26: Total routing cost in the solutions found by the six algorithms on instances from area F

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
N1_g	6-2	9,759,430	5,849,560	7,566,150	12,341,000	12,455,900	5,884,690
N1_g	3-2	11,174,400	7,465,500	9,255,440	13,512,800	13,373,800	7,761,180
N1_g	2-2	12,722,800	9,131,010	10,647,000	15,080,400	15,129,100	9,137,620
N1_p	5-4	10,902,100	3,440,990	4,925,420	13,395,700	14,354,300	3,316,740
N1_p	3-4	10,822,000	4,519,460	5,473,540	13,335,100	14,291,100	4,193,110
N1_p	1-4	15,095,600	8,283,130	9,315,250	17,176,600	18,370,800	8,184,790
N2_g	5-2	9,606,830	5,942,540	7,910,080	12,295,900	12,062,200	6,009,740
N2_g	4-2	10,748,100	6,380,890	8,146,270	12,697,500	13,068,300	6,485,400
N2_g	3-2	11,130,100	7,129,690	8,915,220	13,486,800	13,606,700	7,313,650
N3_g	3-2	10,720,600	6,369,250	8,132,350	12,880,700	13,162,300	6,116,640
N3_g	2-2	11,725,900	7,414,390	9,488,090	13,854,600	14,205,000	7,345,360
N3_g	1-2	15,731,400	11,285,400	13,638,100	17,669,500	17,907,300	11,006,000
N4_g	6-2	9,055,320	4,972,030	6,279,150	11,414,100	11,622,800	4,678,580
N4_g	4-2	10,161,300	5,976,820	7,288,530	12,326,400	12,656,300	5,233,340
N4_g	1-2	14,859,400	10,903,800	12,827,500	17,056,400	17,095,500	10,573,100
N5_g	5-2	9,495,940	6,104,400	7,934,180	12,094,900	11,963,900	5,817,600
N5_g	4-2	9,915,480	6,422,590	7,999,310	12,168,100	12,357,600	6,736,490
N5_g	2-2	11,975,300	8,140,450	10,460,100	14,047,300	14,506,800	8,789,660
N6_g	5-2	9,481,860	5,058,530	6,812,150	11,469,200	11,921,300	4,654,420
N6_g	4-2	9,444,230	5,675,700	7,066,310	11,621,300	12,055,600	5,054,810
N6_g	3-2	9,857,990	6,126,160	7,844,510	11,889,000	12,093,300	5,562,550
N7_g	3-2	2,424,870	1,045,210	1,371,810	2,458,540	2,555,020	1,074,480
N7_g	2-2	2,430,360	1,352,820	1,561,570	2,460,060	2,559,670	1,421,140
N7_g	1-2	3,059,130	1,916,380	2,082,800	3,146,840	3,281,240	1,948,130
N9_g	6-2	2,517,530	1,447,040	2,027,750	2,723,230	2,799,650	1,341,560
N9_g	3-2	2,991,120	1,933,520	2,704,660	3,319,310	3,261,690	1,770,300
N9_g	2-2	2,823,090	2,113,670	2,852,120	3,206,820	3,174,630	2,203,880
N9_p	5-4	3,238,070	922,176	1,385,370	3,442,890	3,546,180	729,129
N9_p	2-4	3,238,140	1,239,060	1,308,890	3,432,090	3,531,800	1,184,420
N9_p	1-4	3,282,900	1,542,250	1,866,940	3,024,430	3,540,590	1,644,970
N10_g	5-2	4,018,490	2,027,590	2,747,350	5,079,630	5,268,900	2,098,780
N10_g	2-2	4,413,060	2,591,010	3,241,830	5,463,110	5,637,800	2,638,360
N10_g	1-2	5,025,870	3,257,900	4,100,200	6,035,320	6,014,740	3,412,140
N11_g	4-2	2,389,660	1,315,240	1,620,310	2,862,350	2,954,550	1,121,660
N11_g	2-2	2,472,890	1,395,730	1,818,030	2,968,070	3,035,580	1,339,820
N11_g	1-2	2,721,050	1,535,280	2,026,530	3,055,570	3,160,870	1,642,570
N12_g	6-2	859,732	403,555	598,241	1,040,190	1,030,040	398,096
N12_g	2-2	900,274	547,005	676,217	1,019,060	1,039,180	456,508
N12_g	1-2	954,236	610,520	753,750	1,065,600	1,082,700	595,095
N13_g	6-2	324,528	142,043	195,621	393,010	379,421	141,156
N13_g	3-2	321,641	152,070	209,954	381,499	359,483	156,058
N13_g	1-2	344,104	196,288	273,437	398,816	406,950	195,929

Table 27: Total routing cost in the solutions found by the six algorithms on instances from area N

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
N14_g	6-2	4,339,660	2,786,910	3,692,010	5,486,990	5,494,190	2,293,660
N14_g	4-2	4,610,270	2,840,040	3,814,650	5,744,110	5,704,520	2,764,640
N14_g	2-2	5,180,470	3,493,050	4,261,530	6,277,770	6,271,490	3,895,150
N15_g	5-2	1,740,480	1,000,880	1,339,720	2,151,070	2,198,440	816,423
N15_g	3-2	1,764,810	1,016,580	1,270,030	2,195,410	2,144,850	945,595
N15_g	1-2	1,930,920	1,158,400	1,401,680	2,384,090	2,411,750	1,178,840
N16_g	6-2	486,751	286,643	402,764	571,556	558,108	229,096
N16_g	4-2	500,284	341,531	395,298	558,729	558,399	233,996
N16_g	1-2	550,331	322,760	414,276	662,933	649,019	320,002
N17_g	5-2	98,553	54,410	70,235	119,718	128,519	53,983
N17_g	3-2	102,687	55,983	83,558	119,671	128,120	49,288
N17_g	1-2	107,030	63,495	84,077	126,811	141,673	66,140

Table 28: Total routing cost in the solutions found by the six algorithms on instances from area N (continued)

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
O1_g	6-2	7,723,130	4,024,450	4,776,850	9,599,880	9,904,680	3,536,730
O1_g	5-2	7,929,410	4,045,140	4,984,190	9,780,480	10,168,600	3,707,030
O1_g	4-2	8,405,990	4,566,870	5,125,700	10,090,200	10,414,900	4,062,540
O1_p	4-4	9,820,950	3,843,820	4,334,310	13,503,300	13,530,700	3,245,130
O1_p	3-4	10,250,500	3,982,650	4,576,910	13,507,200	13,856,100	3,602,040
O1_p	2-4	10,914,900	4,566,810	4,922,370	14,281,300	14,848,200	4,019,150
O6_p	3-4	9,162,730	2,934,750	3,404,660	12,025,400	12,115,400	2,710,670
O6_p	2-4	9,429,160	3,231,240	3,688,640	12,114,300	12,701,700	3,030,690
O6_p	1-4	10,995,800	4,400,100	4,598,400	13,482,900	14,400,000	4,281,810
O9_g	6-2	3,144,820	1,985,170	2,244,800	3,841,360	4,006,670	1,648,590
O9_g	5-2	3,167,470	1,980,000	2,373,700	3,895,120	4,040,420	1,776,210
O9_g	2-2	3,959,010	2,670,730	2,930,450	4,848,700	4,998,730	2,462,660
O9_p	6-4	3,543,200	1,558,220	1,802,500	4,499,250	4,520,020	1,241,510
O9_p	3-4	3,691,710	1,714,280	1,996,430	4,929,740	5,040,620	1,490,430
O9_p	1-4	4,571,100	2,475,570	2,605,440	5,710,290	6,007,470	2,169,480
O10_g	6-2	3,964,050	2,248,370	2,675,680	4,855,290	5,050,350	2,020,550
O10_g	5-2	4,226,340	2,383,050	2,828,510	4,980,550	5,258,870	2,176,700
O10_g	3-2	4,906,400	3,042,180	3,313,410	5,690,740	5,849,780	2,768,110
O11_g	6-2	1,859,300	1,063,960	1,238,250	2,238,530	2,319,040	855,621
O11_g	4-2	2,043,860	1,168,750	1,296,670	2,285,220	2,471,030	1,075,170
O11_g	3-2	2,185,990	1,267,520	1,467,080	2,509,820	2,536,390	1,171,600
O12_g	6-2	457,165	238,400	267,352	507,412	505,654	169,848
O12_g	2-2	517,317	249,509	289,873	550,731	558,277	234,458
O12_g	1-2	569,298	294,940	327,347	626,147	649,664	291,561
O13_g	5-2	130,219	52,859	59,666	131,321	129,019	48,567
O13_g	2-2	130,108	56,306	62,028	126,671	128,762	53,574
O13_g	1-2	130,043	67,533	70,216	129,094	131,931	69,081
O14_g	6-2	3,236,990	1,814,100	2,200,510	3,933,910	4,171,500	1,635,790
O14_g	4-2	3,579,470	2,081,170	2,412,920	4,283,270	4,513,550	1,997,060
O14_g	2-2	4,516,120	2,916,390	3,107,160	5,155,340	5,383,210	2,740,300
O15_g	6-2	1,251,110	780,346	956,922	1,621,720	1,734,480	695,687
O15_g	5-2	1,309,850	810,164	1,020,560	1,699,760	1,807,300	767,785
O15_g	2-2	1,837,190	1,334,520	1,393,390	2,197,870	2,270,230	1,236,700
O16_g	6-2	519,358	324,315	445,589	717,535	752,729	296,257
O16_g	4-2	540,354	364,115	443,307	772,149	841,894	331,014
O16_g	2-2	640,511	439,444	521,570	828,355	939,897	439,674
O17_g	2-2	181,068	113,161	133,899	215,683	245,924	113,259
O17_g	3-2	166,156	101,411	130,226	197,163	220,762	96,923
O17_g	2-2	177,867	114,375	137,642	210,623	236,386	116,969

Table 29: Total routing cost in the solutions found by the six algorithms on instances from area O

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
K1_g	6-2	11,788,500	6,685,590	8,652,810	14,305,700	14,940,100	6,634,350
K1_g	5-2	12,063,200	7,473,350	8,939,610	14,652,700	15,139,600	7,125,180
K1_g	2-2	14,502,300	9,817,210	10,898,500	16,985,700	17,546,300	10,059,300
K1_p	3-8	19,175,500	5,109,530	6,274,760	23,023,800	25,182,300	4,780,200
K1_p	2-8	20,598,900	5,934,880	6,912,020	24,584,400	26,514,300	5,436,670
K1_p	1-8	22,095,200	8,112,590	9,122,880	25,630,800	28,501,300	7,659,610
K2_g	3-4	25,850,400	16,339,900	19,145,300	31,709,200	33,093,500	16,487,400
K2_g	4-4	24,269,800	15,109,300	18,458,300	30,232,800	31,120,200	15,584,400
K2_g	2-4	27,690,600	18,587,200	20,246,400	33,947,700	34,492,100	19,598,100
K3_g	3-2	11,971,500	6,619,430	7,557,150	14,345,900	15,104,400	5,765,110
K3_g	4-2	11,410,000	6,021,370	7,065,530	13,644,900	14,539,100	5,432,870
K3_g	2-2	12,826,500	7,203,550	8,781,020	14,980,500	15,732,500	6,889,480
K4_g	6-4	21,567,400	9,879,760	12,746,800	27,223,200	27,837,400	8,987,410
K4_g	4-4	22,491,100	11,285,600	13,390,900	27,901,100	28,544,300	10,457,500
K4_g	3-4	23,716,400	11,851,700	14,782,400	29,258,400	30,157,000	11,210,300
K5_g	6-2	11,411,500	6,951,880	8,187,110	13,880,200	14,290,600	6,674,710
K5_g	4-2	11,889,000	7,569,200	8,726,440	14,688,500	15,007,500	7,675,860
K5_g	2-2	13,825,200	9,075,210	10,560,300	16,131,700	16,745,300	9,254,980
K6_g	5-2	10,858,400	5,402,930	6,444,270	13,537,600	13,969,600	4,632,990
K6_g	3-2	11,498,700	6,061,120	7,152,160	13,432,500	14,028,200	5,396,410
K6_g	1-2	14,553,800	9,792,290	10,505,600	16,410,500	17,411,700	9,263,020
K6_p	3-8	19,497,000	5,000,890	5,780,960	23,594,000	23,437,100	4,264,450
K6_p	2-8	18,998,800	5,285,080	6,211,350	23,091,700	25,143,900	4,794,960
K6_p	1-8	21,202,600	7,148,030	7,951,310	23,917,100	27,358,500	6,509,700
K7_g	5-2	2,112,840	823,421	933,708	2,112,380	2,188,270	834,086
K7_g	2-2	2,126,580	1,255,630	1,238,860	2,187,920	2,157,480	1,032,110
K7_g	1-2	2,654,730	1,499,760	1,609,960	2,669,930	2,722,990	1,422,420
K9_g	5-2	3,956,070	2,757,950	3,816,180	4,621,620	4,730,120	2,583,280
K9_g	4-2	4,285,290	3,108,160	3,593,350	4,861,360	4,854,490	2,714,760
K9_g	2-2	4,618,740	3,549,190	4,161,640	5,560,820	5,633,670	3,697,230
K9_p	3-8	5,443,760	1,346,490	1,633,360	5,782,730	6,048,570	1,191,930
K9_p	2-8	5,483,540	1,326,690	1,797,800	5,874,200	6,081,120	1,353,690
K9_p	1-8	5,485,490	1,917,840	2,323,260	5,832,200	6,052,530	1,806,980
K10_g	6-2	4,174,200	2,581,410	3,040,040	4,994,810	5,077,080	2,315,040
K10_g	4-2	4,313,100	2,684,150	3,284,820	5,056,550	5,345,040	2,557,150
K10_g	2-2	5,103,170	3,223,400	3,720,140	5,715,760	5,869,530	3,273,050
K11_g	6-2	2,807,670	1,463,880	1,900,590	3,479,870	3,515,610	1,494,500
K11_g	4-2	2,987,460	1,685,490	2,183,490	3,487,580	3,645,410	1,575,890
K11_g	3-2	2,958,700	1,839,730	2,149,810	3,489,500	3,681,150	1,815,250
K12_g	4-2	1,338,820	656,221	785,264	1,635,180	1,698,120	520,326
K12_g	2-2	1,339,230	715,697	806,394	1,687,800	1,648,580	583,527
K12_g	1-2	1,395,740	749,855	941,629	1,641,940	1,796,790	650,787

Table 30: Total routing cost in the solutions found by the six algorithms on instances from area K

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
K13_g	3-2	494,762	194,657	265,953	573,344	576,177	179,974
K13_g	2-2	497,093	211,817	304,641	571,956	559,686	189,522
K13_g	1-2	495,964	246,441	313,263	585,653	559,174	211,508
K14_g	5-2	6,677,490	4,241,040	5,161,280	8,035,780	7,930,200	4,256,080
K14_g	3-2	7,064,080	4,942,390	5,518,420	8,426,560	8,805,750	4,862,380
K14_g	2-2	7,863,490	5,436,990	6,227,590	8,994,320	9,334,800	5,591,350
K15_g	5-2	3,003,370	2,047,990	2,539,060	3,831,860	3,986,820	1,771,520
K15_g	3-2	3,110,090	1,979,240	2,721,920	3,976,940	3,994,450	1,989,860
K15_g	1-2	4,029,490	2,789,250	3,314,110	4,679,500	4,856,580	2,896,030
K16_g	6-2	1,310,950	855,813	1,384,750	1,701,770	1,703,940	730,578
K16_g	4-2	1,362,470	1,055,560	1,170,110	1,752,080	1,822,350	846,148
K16_g	1-2	1,781,170	1,213,240	1,429,350	2,035,690	2,165,170	1,274,930
K17_g	3-2	461,282	185,172	272,287	544,606	554,681	172,631
K17_g	2-2	450,853	191,584	277,708	537,654	541,805	178,305
K17_g	1-2	455,503	231,462	292,164	524,614	546,186	200,812

Table 31: Total routing cost in the solutions found by the six algorithms on instances from area K (continued)

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
S1_g	6-2	5,800,530	3,222,750	3,851,670	6,817,450	6,874,090	3,024,840
S1_g	5-2	5,912,290	3,380,410	4,004,700	7,084,430	7,055,350	3,256,780
S1_g	1-2	7,828,250	5,179,330	6,099,900	9,160,180	8,711,030	5,376,340
S1_p	4-4	4,485,440	1,527,340	1,715,250	5,297,200	5,476,570	1,144,820
S1_p	2-4	4,578,860	1,601,350	1,846,720	5,091,390	5,526,870	1,325,790
S1_p	1-4	4,907,970	1,834,140	2,051,050	5,253,370	6,132,370	1,733,310
S2_g	6-2	5,713,120	3,471,360	3,824,670	6,664,380	7,007,170	3,089,060
S2_g	2-2	6,558,740	3,756,540	4,710,450	7,855,730	7,571,330	3,995,290
S2_g	1-2	7,876,050	5,159,440	5,833,830	9,066,820	8,532,670	5,161,130
S3_g	5-2	5,785,170	2,884,380	3,676,390	6,870,690	6,887,170	2,812,460
S3_g	3-2	6,083,520	3,124,710	3,972,040	7,119,400	7,185,990	3,109,780
S3_g	1-2	7,511,430	4,957,930	5,546,930	8,420,740	8,551,700	4,723,210
S4_g	6-2	5,646,540	2,867,780	3,494,380	6,868,540	6,985,290	2,735,900
S4_g	2-2	6,276,390	3,491,430	4,427,680	7,532,480	7,229,660	3,438,550
S4_g	1-2	7,313,400	4,818,700	5,203,380	8,368,720	8,242,490	4,511,650
S5_g	6-2	5,666,140	2,935,720	3,740,520	6,847,510	6,958,780	2,904,550
S5_g	3-2	5,972,660	3,319,550	4,173,130	7,023,170	6,908,840	3,520,950
S5_g	2-2	6,323,750	3,747,940	4,537,210	7,562,550	7,530,540	3,778,180
S6_g	6-2	5,669,520	2,688,700	3,376,290	6,677,820	6,753,290	2,604,240
S6_g	4-2	5,690,610	3,006,560	3,628,740	6,840,940	6,977,460	2,910,470
S6_g	2-2	6,157,800	3,425,150	4,155,220	7,138,440	7,231,920	3,327,240
S6_p	3-4	4,346,270	1,141,280	1,319,940	5,276,950	5,530,380	992,960
S6_p	2-4	4,393,970	1,167,010	1,416,030	5,315,360	5,702,810	1,096,060
S6_p	1-4	4,760,170	1,444,320	1,710,490	5,456,980	5,989,100	1,413,160
S7_g	4-2	2,654,700	1,032,890	1,436,720	2,683,170	2,813,890	1,072,750
S7_g	2-2	2,639,300	1,150,310	1,479,660	2,684,510	2,811,130	1,237,430
S7_g	1-2	2,654,960	1,287,550	1,624,450	2,808,390	3,083,850	1,387,960
S9_g	4-2	1,347,280	700,612	929,289	1,414,840	1,427,770	575,074
S9_g	2-2	1,408,030	834,658	997,701	1,378,750	1,503,290	681,740
S9_g	1-2	1,485,510	986,575	1,001,620	1,719,680	1,700,340	904,494
S9_p	3-4	1,285,460	521,071	592,096	1,291,560	1,247,370	354,358
S9_p	2-4	1,308,690	530,197	606,380	1,314,810	1,266,840	353,521
S9_p	1-4	1,264,790	554,602	575,515	1,246,170	1,245,170	409,096
S10_g	6-2	3,442,520	1,895,040	2,342,270	4,002,630	4,208,130	1,696,560
S10_g	4-2	3,397,220	2,104,490	2,566,260	3,976,750	4,126,930	2,003,790
S10_g	1-2	4,687,400	3,218,330	3,706,370	5,360,110	5,193,400	3,346,870
S11_g	5-2	1,411,190	806,719	1,225,140	1,737,240	1,752,840	714,630
S11_g	3-2	1,443,720	849,160	1,087,250	1,772,300	1,740,340	732,529
S11_g	2-2	1,505,540	929,119	1,110,800	1,783,090	1,833,970	918,898
S12_g	5-2	731,142	393,887	560,230	820,481	821,571	335,557
S12_g	3-2	720,849	441,813	570,856	833,760	820,367	346,021
S12_g	1-2	783,958	417,109	519,258	869,804	890,029	425,570

Table 32: Total routing cost in the solutions found by the six algorithms on instances from area S

Graph	Vehicle	EO1	EO2	EO3	MR-R	MR-SD	MR-H
S13_g	3-2	305,304	126,782	172,445	334,222	333,507	128,878
S13_g	2-2	304,718	135,252	177,799	338,286	335,145	129,838
S13_g	1-2	302,312	150,167	201,685	338,648	337,548	142,284
S14_g	5-2	2,541,110	1,406,780	1,716,500	3,054,320	3,067,310	1,266,830
S14_g	3-2	2,597,880	1,456,700	1,745,200	3,102,710	3,162,340	1,425,480
S14_g	1-2	3,118,340	2,021,350	2,278,520	3,641,910	3,782,570	2,024,350
S15_g	6-2	1,373,590	712,464	846,574	1,594,090	1,654,270	651,879
S15_g	3-2	1,412,430	749,405	903,436	1,667,320	1,720,080	805,797
S15_g	2-2	1,463,090	766,944	941,938	1,693,620	1,740,630	844,966
S16_g	5-2	621,073	255,678	305,663	663,875	679,908	278,467
S16_g	2-2	614,075	279,027	321,780	652,286	683,313	298,178
S16_g	1-2	657,219	302,989	388,389	716,417	744,646	335,223
S17_g	3-2	351,702	154,093	193,478	391,373	416,940	152,190
S17_g	2-2	351,702	158,639	203,664	379,315	419,191	161,757
S17_g	1-2	351,702	220,663	235,520	383,545	420,148	167,516

Table 33: Total routing cost in the solutions found by the six algorithms on instances from area S (continued)