Operations Research Letters 44 (2016) 569–574

Contents lists available at ScienceDirect

Operations Research Letters

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

Splitting procedures for the Mixed Capacitated Arc Routing Problem under Time restrictions with Intermediate Facilities

Elias J. Willemse *, Johan W. Joubert

Center of Transport Development, Department of Industrial and Systems Engineering, University of Pretoria, 0002, South Africa

ARTICLE INFO

Article history:
Received 20 August 2014
Received in revised form
7 June 2016
Accepted 7 June 2016
Available online 18 June 2016

Keywords:
Waste collection
Capacitated Arc Routing Problem
Intermediate Facilities
Mixed network
Splitting procedures

ABSTRACT

This paper develops optimal and quick near-optimal splitting procedures for the Mixed Capacitated Arc Routing Problem under Time restrictions with Intermediate Facilities. Splitting procedures are a key component of giant tour-based solution methods for Arc Routing Problems. The optimal and near-optimal splitting procedures are tested within a multi-start constructive heuristic, and a fixed execution-time limit is imposed. Results on benchmark instances show that the constructive-heuristic linked with the new optimal splitting algorithm performs better than the near-optimal versions.

1. Introduction

The Capacitated Arc Routing Problem under Time restrictions with Intermediate Facilities (CARPTIF), first proposed by Ghiani et al. [6], and also referred to as the Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions (CLARTPIF), is a variant of the classical Capacitated Arc Routing Problem (CARP) and models residential waste collection. On a mixed network, with one and two-way streets in the case of waste collection, the problem is termed the Mixed CARPTIF (MCARPTIF), first proposed by Willemse and Joubert [14]. The problem considers a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E} \cup \boldsymbol{A})$, where $\boldsymbol{V}$ represents the set of vertices, $\boldsymbol{E}$ represents the set of undirected edges that may be traversed in both directions, and $\boldsymbol{A}$ represents the set of arcs that can only be traversed in one direction. For waste collection, $\boldsymbol{V}$ corresponds to road intersections and dead-ends, while $\boldsymbol{E}$ and $\boldsymbol{A}$ model road segments between vertices. A subset of required edges and arcs, $\boldsymbol{E}_r \subseteq \boldsymbol{E}$ and $\boldsymbol{A}_r \subseteq \boldsymbol{A}$, must be serviced by a fleet of K homogeneous vehicles with limited capacity, Q, that are based at the depot vertex, v_1. The fleet size K can be either fixed, left as a decision variable or treated as unlimited. Vehicles are allowed to unload their waste at any Intermediate Facility (IF) at a cost of λ, and resume their collection routes. At the end of its route a vehicle must first visit an IF before returning to the depot. The set of IFs is modelled in $\boldsymbol{G}$ as $\boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma} \subset \boldsymbol{V}$. The sum of demand on each sub-trip between IF visits may not exceed Q, and unless $v_1 \in \boldsymbol{\Gamma}$, a vehicle has to visit an IF before returning to the depot. Lastly, a route length or time restriction of L is imposed on each vehicle route. For a comprehensive review of the CARP and other Arc Routing Problems we refer the reader to Corberán and Laporte [3] and Corberán and Prins [4].

Since the CARP and all its extensions are $\mathcal{NP}$-hard the most effective methods for solving the problems are based on heuristics and metaheuristics, many of which employ giant tour approaches that rely on tour splitting procedures [10]. Splitting procedures take as input a giant tour and partitions the tour into feasible vehicle routes. In this paper we present optimal and heuristic splitting procedures for the MCARPTIF that can be used in giant tour approaches for the problem. The structure of our optimal splitting procedure provides a substantial improvement in efficiency over the existing CAPRTIF version that we adapted for mixed networks. Fast, near-optimal splitting procedures are also presented. The optimal and near-optimal procedures were tested in a multi-start Route-First–Cluster-Second heuristic on large MCARPTIF instances. Tight time-limits were imposed to reduce the number of starts of the slower, optimal procedures compared to the faster near-optimal procedures. Even with less starts, the Route-First–Cluster-Second heuristic linked with our efficient optimal splitting procedure performed better than the near-optimal splitting versions.

The following is an outline of the remainder of the paper. In the next section we review current splitting procedures for the CARP

* Corresponding author.
E-mail addresses: ejwillemse@gmail.com (E.J. Willemse), johan.joubert@up.ac.za (J.W. Joubert).

http://dx.doi.org/10.1016/j.orl.2016.06.001
0167-6377/

and a few of its extensions. In Section 3 we present the algorithm notation and detailed descriptions of our splitting procedures. In Section 4 we report on computational experiments, focusing on the execution times of the procedures and the difference in partition costs between optimal and near-optimal splitting. We then compare the performance of the different procedures within a multi-start Route-First–Cluster-Second heuristic. The paper concludes in Section 5 with a summary of our main findings and suggestions for future research opportunities.

2. Current splitting procedures for the CARP and CAPRTIF

The first optimal splitting procedure for the CARP was developed by Ulusoy [12] as part of a Route-First–Cluster-Second constructive heuristic. The heuristic is similar to the one of Beasley [1] for the Vehicle Routing Problem. First, edge demands are ignored and a giant tour is constructed servicing all the required edges in $\boldsymbol{G}$. In the second phase, an auxiliary Directed Acyclic Graph (DAG) is constructed whose arcs represent feasible sub-tours of the giant tour, with respect to demand of the sub-tour and vehicle capacity. The DAG is constructed in such a way that the shortest path through the graph gives the optimal partition of the giant tour into feasible vehicle routes. The shortest path can be calculated using any shortest path algorithm. The splitting procedure consists of constructing the DAG, calculating the shortest path through the graph, and decoding the shortest path to retrieve the optimal giant tour partitions. Lacomme et al. [8] and Belenguer et al. [2] develop multi-start Route-First–Cluster-Second heuristics for the CARP and Mixed CARP (MCARP), respectively, whereby different giant tours are constructed and partitioned, and the best returned as the final solution.

Ghiani et al. [7] develop a splitting procedure, similar to CARP versions, for the Capacitated Arc Routing Problem with Intermediate Facilities (CARPIF). Their procedure calculates the optimal placement of Intermediate Facility (IF) visits within a route. The problem allows inter-route offloads so that collected demand between IF visits never exceeds vehicle capacity, but it does not impose route duration limits. As such, a solution always consists of only one route. When a route duration limit is imposed the problem generalises to the CAPRTIF. To solve the problem Ghiani et al. [6] develop a splitting procedure that constructs two DAGs. The first consists of multiple source and destination vertices, each representing a start- and end-edge of a sub-tour in the giant tour. Shortest paths through the DAG between the sources and destinations represent the optimal placement of IFs in all possible sub-tours. The shortest path costs, calculated using a shortest path algorithm, are then used to construct a second DAG whose shortest path represents the optimal partition of the giant tour into vehicle routes. The optimal placement of IFs in each route is traced back to the shortest paths in the first DAG. The splitting procedure of Ghiani et al. [6] can be applied as-is to giant tours on mixed networks. A solution for the MCARPTIF can thus be obtained by combining the Route-First phase of Belenguer et al. [2] for the MCARP to construct a giant tour on a mixed network, and then applying the CARPTIF splitting procedure of Ghiani et al. [6] for the Cluster-Second phase.

To improve the efficiency of splitting procedures Lacomme et al. [8] develop a compact procedure for the CARP that does not explicitly construct the DAG. Instead, the shortest path through the DAG is directly calculated when scanning sub-tours for their feasibility with respect to vehicle capacity limits. Their version also accounts for a secondary objective of minimising fleet size. The compact version is exclusively used in Memetic Algorithms for the CARP [8,9,11] and MCARP [2], which are currently some of the most effective solution methods for the problems. Memetic Algorithms are metaheuristics based on genetic algorithms enhanced with local search procedures. Chromosomes are encoded as giant tours and an optimal splitting procedure is used to determine chromosome fitness each time a new chromosome is evaluated. An efficient splitting procedure is critical for the applications since chromosome evaluation occurs tens of thousands of times during the MA's execution.

In this paper we extended the compact splitting version of Lacomme et al. [8] to the MCARPIF. We then further extended this version to deal with the MCARPTIF and show that it provides a substantial improvement in efficiency over the version of Ghiani et al. [6]. We also developed two quick heuristic splitting procedures, one that greedily inserts IF visits into sub-tours and then calculates the route partitions and a second that employs a next-fit bin-packing procedure.

3. New splitting procedures

Before presenting our splitting procedures we first describe the graph transformation of $\boldsymbol{G}$ and introduce the basic algorithm notation. Consistent with Belenguer et al. [2] and Lacomme et al. [8], the graph $\boldsymbol{G}$ is transformed into a fully directed graph, $\boldsymbol{G}^* = (\boldsymbol{V}, \boldsymbol{A}^*)$. CARPTIF splitting procedures can then be used as-is on the MCARPTIF, and the other way around. Required arcs, $\boldsymbol{A}_r$, and edges, $\boldsymbol{E}_r$, of $\boldsymbol{G}$ correspond in $\boldsymbol{G}^*$ to a subset $\boldsymbol{R} \subseteq \boldsymbol{A}^*$ of required arcs. Each arc, $u \in \boldsymbol{R}$, has a demand, $q(u)$, a collection cost, $w(u)$, and a pointer, $inv(u)$, to the arc between the same vertices but in the opposite direction. Each required arc in the original graph, $\boldsymbol{G}$, is coded in $\boldsymbol{R}$ by one arc, u, with $inv(u) = 0$, while each required edge is encoded as two opposite arcs, u and v, such that $inv(u) = v$ and $inv(v) = u$. The depot is modelled by including in $\boldsymbol{A}^*$ a fictitious loop, σ, with zero deadheading and service cost. Similarly, the set of IFs are modelled in $\boldsymbol{A}^*$ as a set of dummy arcs, $\boldsymbol{I}$, such that each IF in $\boldsymbol{\Gamma}$ is modelled as a fictitious loop, $\Phi_i \in \boldsymbol{I}$, and Φ_i also have zero deadheading and service cost. The cost of the shortest path from arc u to arc v, which excludes the costs of deadheading u and v, is given by $D(u, v)$, which is pre-calculated for all arcs in $\boldsymbol{A}^*$. Shortest paths can be efficiently calculated using a modified version of Dijkstra's algorithm, and may also incorporate forbidden turns and turn-penalties [8]. The best IF to visit after servicing arc u and before servicing arc v can be pre-calculated using

$$\Phi^*(u, v) = \arg\min\{D(u, k) + D(k, v) : k \in \boldsymbol{I}\}, \tag{1}$$

$$\mu^*(u, v) = D\big(u, \Phi^*(u, v)\big) + D\big(\Phi^*(u, v), v\big) + \lambda, \tag{2}$$

where $\Phi^*(u, v)$ gives the best IF to visit, and $\mu^*(u, v)$ gives the cost of the visit, including the unloading cost, λ, and deadheading costs. We denote by $\boldsymbol{S}$ the giant tour to be partitioned, which consists of a list of tasks, $[S_1, \ldots, S_{|\boldsymbol{S}|}]$. It is assumed that the shortest path is always followed between consecutive tasks and only arcs in $\boldsymbol{R}$ are included in $\boldsymbol{S}$, thus it contains no depot or IF dummy arcs as these are implicitly accounted for by the splitting procedures. Sub-tours in $\boldsymbol{S}$ are denoted as $\boldsymbol{S}_{i\to j} = [S_i, \ldots, S_j]$ where $1 \le i < j \le n$ and $n = |\boldsymbol{S}|$. A single MCARPIF or MCAPRTIF route is a list of tasks that always starts with the dummy depot task, ends with a dummy IF and depot task, and may include inter-route IF visits. The list of tasks between dummy arcs represent subtrips and the load collected on a subtrip may not exceed Q. For the MCAPRTIF, the total cost of a route, including all task service costs, deadheading costs between tasks and IF visit costs, may not exceed L.

3.1. Splitting procedures for the MCARPIF

The first splitting procedure that we present is an MCARPIF adaption of the CARP procedure developed by Lacomme et al. [8]. Recall that splitting procedures use $\boldsymbol{S}$ to construct an auxiliary DAG, $\boldsymbol{H}$, in such a way that its shortest path represents the optimal

giant tour partition. For the CARPIF, Ghiani et al. [7] construct $\boldsymbol{H}$ by including a vertex for each feasible sub-tour $\boldsymbol{S}_{i\to j}$ with respect to vehicle capacity. Vertices representing consecutive sub-tours $\boldsymbol{S}_{i\to j}$ and $\boldsymbol{S}_{j+1\to k}$ are then linked with arcs. A source vertex, linked to vertices representing $\boldsymbol{S}_{1\to j}$, and a sink vertex, linked to vertices representing $\boldsymbol{S}_{k\to|\boldsymbol{S}|}$, are also included in $\boldsymbol{H}$ and the shortest path from the source to the sink represents the optimal partition. Using this approach the DAG consists of at most $\frac{n(n+1)}{2}$ vertices and $\frac{n(n+1)(n-1)}{6}+2n$ arcs. Bellman's algorithm [5] can then compute the shortest path through $\boldsymbol{H}$ in $\mathcal{O}(n^3)$. For the CARP, Lacomme et al. [8] construct $\boldsymbol{H}$ so that it consists of exactly $n+1$ vertices and at most $\frac{n(n+1)}{2}$ arcs where each arc, instead of vertices, represents a feasible sub-tour. Computing the shortest path through $\boldsymbol{H}$ then takes $\mathcal{O}(n^2)$. For a detailed illustration of the method we refer the reader to Appendix A.

To improve the efficiency of split for the CARP, Lacomme et al. [8] develop a compact version that does not explicitly construct **H**. Algorithm 1 shows a similar procedure, called *Efficient-IF-Split*, that we developed for the MCARPIF. It directly calculates the optimal partition and further minimises the number of subtrips as a second objective. Modifications to the CARP procedure of Lacomme et al. [8] are contained in Lines 1, and 9–19, with the CARP calculations shown as comments (//) in Lines 2, 13, 16 and 19. Three labels are used by the procedure. The first, $\boldsymbol{N}_i$, represents the cost of the shortest path from vertex 0 to i in $\boldsymbol{H}$, the second, $\boldsymbol{\Pi}_i$, represents the number of sub-trips in the same shortest path, and the third, $\boldsymbol{P}_i$, represents the predecessor vertex of i on this path and thus stores the resulting optimal placement of IFs in $\boldsymbol{S}$. Note that both $\boldsymbol{N}$ and $\boldsymbol{P}$ are indexed from 0. The depot and final IF and depot visit on a feasible route are implicitly accounted for by the algorithm. By increasing i and j the procedure successively scans sub-tours for capacity violations. When the feasible sub-tour $\boldsymbol{S}_{i\to j}$ is found the optimal partition for the partial giant tour ending at j is updated. In the worst case a total of $\frac{n(n+1)}{2}$ sub-tours can be evaluated, giving the algorithm a running time of $\mathcal{O}(n^2)$. The actual running time of the algorithm is reduced as only feasible sub-tours that meet the capacity constraint are evaluated.

Algorithm 1: *Efficient-IF-Split* for the MCARPIF

Input : $\boldsymbol{S}$
Output: $\boldsymbol{N}, \boldsymbol{P}$

1 $n = |\boldsymbol{S}|$; $\Pi_0 = 0$; $\boldsymbol{P}_0 = 0$; $\boldsymbol{N}_0 = D(\sigma, S_1)$;
2 $//^* \boldsymbol{N}_0 = 0$;
3 **for** $i \leftarrow 1$ **to** n **do** $\boldsymbol{N}_i = \infty$; $\Pi_i = \infty$
4 **for** $i \leftarrow 1$ **to** n **do**
5 $l' = 0$; $c' = 0$; $j = i$;
6 **repeat**
7 $l' = l' + q(S_j)$;
8 **if** $l' \leq Q$ **then**
9 **if** $j = n$ **then**
10 $a = \mu^*(S_j, \sigma)$
11 **else**
12 $a = \mu^*(S_j, S_{j+1})$
13 $//^* a = D(S_j, \sigma)$;
14 **if** $i = j$ **then**
15 $c' = w(S_j) + a$;
16 $//^* c' = D(\sigma, S_j) + w(S_j) + a$;
17 **else**
18 $\Delta c' = D(S_{j-1}, S_j) - \mu^*(S_{j-1}, S_j)$;
19 $//^* \Delta c' = D(S_{j-1}, S_j) - D(S_{j-1}, \sigma)$;
20 $c' = c' + \Delta c' + w(S_j) + a$;
21 **if** $(\boldsymbol{N}_{i-1} + c' < \boldsymbol{N}_j)$ *or* $\left((\boldsymbol{N}_{i-1} + c' = \boldsymbol{N}_j)$ *and* $(\Pi_{i-1} + 1 < \Pi_j)\right)$ **then**
22 $\boldsymbol{N}_j = \boldsymbol{N}_{i-1} + c'$;
23 $\Pi_j = \Pi_{i-1} + 1$;
24 $\boldsymbol{P}_j = i - 1$;
25 $j = j + 1$;
26 **until** $(j > n)$ *or* $(l' > Q)$;
27 **return** $(\boldsymbol{N}, \boldsymbol{P})$

*Original calculations of Lacomme et al. [8] for the CARP.

3.2. Splitting procedures for the MCAPRTIF

For the MCAPRTIF the splitting procedure has to simultaneously determine the optimal IF partitions, resulting from the vehicle capacity limit Q, and the optimal route partitions, resulting from the route time restriction, L. Ghiani et al. [6] extend the method of Ghiani et al. [7] and explicitly construct two DAGs for this purpose. *Efficient-IF-Split* can be used in the same way. First, it calculates the optimal IF partitions for *all* the sub-tours in $\boldsymbol{S}$. The cost of the partitioned sub-tours is then used to construct the second DAG, and the shortest path through it represents the optimal route partitions. For a detailed illustration of the method we refer the reader to Appendix A.

Similar to *Efficient-IF-Split*, $\boldsymbol{H}$ and $\boldsymbol{H}'$ need not be explicitly constructed. Algorithm 2 shows a compact splitting version for the MCAPRTIF that directly calculates the optimal partitions for subtrips and routes. We refer to this version as *Efficient-Split* and to the version that calculates the optimal IF and route partitions in two-phases as *Two-Phase-Split*. Algorithm 2 minimises the total number of routes as the primary objective, and partition cost as secondary. For the IF partitions, two labels are used for each vertex i in $\boldsymbol{H}$. The first, $\boldsymbol{N}_{i,j}$, where $i \geq j$, represents the cost of the shortest path from vertex i to j in $\boldsymbol{H}$, and the second, $\boldsymbol{P}_{i,j}$, where $i \geq j$, represents the predecessor vertex of j on this path back to i and thus stores the resulting optimal placement of IFs in sub-tours of $\boldsymbol{S}$. Two more labels are used for the optimal route partitions. The first, $\boldsymbol{N}'_i$, represents the cost of the shortest path from vertex 0 to i in $\boldsymbol{H}'$, and the second, $\boldsymbol{P}'_i$, represents the predecessor vertex of i on this path and thus stores the resulting optimal route partitions of $\boldsymbol{S}$. Since the MCAPRTIF has multiple routes, the partitioning algorithm is further extended by including $\boldsymbol{\Pi}_i$, which represents the number of routes required from vertex 0 to i in $\boldsymbol{H}'$. By increasing i and j the procedure successively scans sub-tours for capacity violations. When the feasible sub-tour $\boldsymbol{S}_{i\to j}$ is found the optimal partitions for all sub-tours starting at $k \in \{1, \ldots, i\}$ and ending at j are updated. In the worst case, a total of $\frac{n(n+1)(n+2)}{6}$ sub-tours are evaluated for the optimal IF partitions, giving the algorithm a running time of $\mathcal{O}(n^3)$. The actual running time of Algorithm 2 is reduced as only feasible sub-tours, adhering to the route time restriction, are optimally partitioned with IF visits. When $i = j$ the optimal partition of all sub-tours from $k \in \{1, \ldots, j\}$ to j have been calculated. If $\boldsymbol{S}_{k\to j}$ then exceeds L, longer sub-tours starting at k will also exceed L and they need not be updated with IF partitions in subsequent iterations. This significantly reduces the number of sub-tours that have to be updated each time i and j are incremented.

3.3. Heuristic splitting procedures for the MCAPRTIF

The last two splitting procedures that we developed for the MCAPRTIF are heuristic in nature and thus do not guarantee an optimal partition. The first is a straightforward *Next-Fit* bin-packing type procedure that starts with a route consisting of the starting depot task. Starting with the first task in $\boldsymbol{S}$, tasks are progressively added in sequence to the route. If a task cannot be added without exceeding Q, an IF visit is included before the task. If a task cannot be added to the route without exceeding L, including the cost of going from the task to the nearest IF and depot, the route is closed and the task is added to a new route. The procedure, which we call *Simple-Split*, runs in $\mathcal{O}(n)$.

The second heuristic procedure that we developed, called *Heuristic-Split*, is shown in Algorithm 3. It is an extension of

Algorithm 2: *Efficient-Split* for the MCAPRTIF

Input : $\boldsymbol{S}$
Output: $\boldsymbol{N}, \boldsymbol{P}, \boldsymbol{N}', \boldsymbol{P}'$

$n = |\boldsymbol{S}|$;
for $i \leftarrow 0$ **to** $n-1$ **do** $\boldsymbol{N}_{i,i} = \boldsymbol{D}(\sigma, \boldsymbol{S}_i)$; $\boldsymbol{P}_{i,i} = i$ **for** $i \leftarrow 1$ **to** $n-1$ **do**
  for $j \leftarrow i+1$ **to** n **do**
    $\boldsymbol{N}_{i,j} = \infty$;
$\Pi'_0 = 0$; $\boldsymbol{P}'_0 = 0$; $\boldsymbol{N}'_0 = 0$;
for $i \leftarrow 1$ **to** n **do** $\boldsymbol{N}'_i = \infty$; $\Pi_i = \infty$ $k_s = 0$;
for $i \leftarrow 1$ **to** n **do**
  $l' = 0$; $c' = 0$; $c'' = 0$; $j = i$; $k_0 = k_s$;
  repeat
    $l' = l' + q(S_j)$;
    if $l' \leq Q$ **then**
      if $j = n$ **then** $a = \mu^*(S_j, \sigma)$ **else** $a = \mu^*(S_j, S_{j+1})$
      if $i = j$ **then**
        $c' = w(S_j) + a$;
      else
        $\Delta c' = \boldsymbol{D}(S_{j-1}, S_j) - \mu^*(S_{j-1}, S_j)$;
        $c' = c' + \Delta c' + w(S_i) + a$;
      $c'' = c' + \mu^*(S_j, \sigma) - a$;
      if $c'' \leq L$ **then**
        for $k \leftarrow k_0$ **to** $i-1$ **do**
          $N_{temp} = \boldsymbol{N}_{k,i-1} + c'$;
          if $N_{temp} < \boldsymbol{N}_{k,j}$ **then**
            $\boldsymbol{N}_{k,j} = N_{temp}$;
            $\boldsymbol{P}_{k,j} = k$;
          $N'_{temp} = \boldsymbol{N}_{k,i-1} + c''$;
          if $N'_{temp} \leq L$ **then**
            if $(\Pi_i + 1 < \Pi_j)$ *or* $\big((\Pi_i + 1 = \Pi_j)$ *and* $(\boldsymbol{N}'_k + N'_{temp} < \boldsymbol{N}'_j)\big)$ **then**
              $\Pi_j = \Pi_i + 1$;
              $\boldsymbol{N}'_j = \boldsymbol{N}'_k + N'_{temp}$;
              $\boldsymbol{P}'_j = k$;
          if $(i = j)$ *and* $(j < n)$ **then**
            $N'_{best} = \boldsymbol{N}_{k,j} - \mu^*(S_j, S_{j+1}) + D(S_j, S_{j+1}) + w(S_{j+1}) + \mu^*(S_{j+1}, \sigma)$;
            if $N'_{best} > L$ **then** $k_s = k_s + 1$
        $j = j + 1$;
  until $(j > n)$ *or* $(l' > Q)$ *or* $(c'' > L)$;
return $(\boldsymbol{N}, \boldsymbol{P}, \boldsymbol{N}', \boldsymbol{P}')$

Algorithm 3: *Heuristic-Split* for the MCAPRTIF

Input : $\boldsymbol{S}$
Output: $\boldsymbol{N}, \boldsymbol{P}$

$n = |\boldsymbol{S}|$; $\boldsymbol{P}_0 = 0$; $\boldsymbol{N}_0 = 0$;
$\Pi'_0 = 0$; **for** $i \leftarrow 1$ **to** n **do** $\boldsymbol{N}'_i = \infty$; $\Pi_i = \infty$
for $i \leftarrow 1$ **to** n **do**
  $l' = 0$; $c' = 0$; $j = i$;
  repeat
    $l' = l' + q(S_j)$;
    if $i = j$ **then**
      $c' = D(\sigma, S_j) + w(S_j) + \mu^*(S_j, \sigma)$;
    else
      if $l' \leq Q$ **then**
        $\Delta c' = D(S_{j-1}, S_j) - \mu^*(S_{j-1}, \sigma)$
      else
        $\Delta c' = \mu^*(S_{j-1}, S_j) - \mu^*(S_{j-1}, \sigma)$;
        $l' = q(S_j)$;
      $c' = c' + \Delta c' + w(S_j) + \mu^*(S_j, \sigma)$;
    if $c' \leq L$ **then**
      if $(\Pi_i + 1 < \Pi_j)$ *or* $\big((\Pi_i + 1 = \Pi_j)$ *and* $(\boldsymbol{N}_{i-1} + c' < \boldsymbol{N}_j)\big)$ **then**
        $\Pi_j = \Pi_i + 1$;
        $\boldsymbol{N}_j = \boldsymbol{N}_{i-1} + c'$;
        $\boldsymbol{P}_j = i - 1$;
      $j = j + 1$;
  until $(j > n)$ *or* $(c' > L)$;
return $(\boldsymbol{N}, \boldsymbol{P})$

Efficient-IF-Split whereby IF visits are included in routes greedily instead of optimally. When evaluating progressively longer sub-tours, IF visits are inserted the moment that the load of a sub-tour since the last IF insertion exceeds vehicle capacity. The optimal placement of IFs is not calculated for each sub-tour, reducing the computational complexity of the algorithm to $\mathcal{O}(n^2)$ while still ensuring that it produces feasible routes with respect to Q and L. After partitioning $\boldsymbol{S}$, the IF partitions for each of the resulting routes can be improved using *Efficient-IF-Split* while maintaining the overall complexity of $\mathcal{O}(n^2)$.

4. Computational results

For the computational tests we analysed and compared the efficiency of *Two-Phase-Split*, *Efficient-Split*, *Simple-Split* and *Heuristic-Split* for the MCAPRTIF. Efficiency was measured as the time taken by each procedure to partition a giant tour. We then compared the number of routes and cost of the giant tour partitions from each procedure. *Heuristic-Split* was always linked with *Efficient-IF-Split* as a post-partition procedure to improve the IF placements in each route. The aim was to determine if *Simple-Split* or *Heuristic-Split* could be potentially used as a substitute for *Two-Phase-Split* and *Efficient-Split* in giant tour solution methods when solving large scale instances.

Two sets of MCAPRTIF benchmark sets developed in [13] were used for our tests. The first set, *Lpr-IF*, is based on the *lpr* MCARP set of Belenguer et al. [2], and the second set, referred to as *Cen-IF*, is based on actual road networks and contains some of the largest arc routing instances currently available. All heuristics were programmed in Python version 2.7, with critical procedures optimised using Cython version 0.17.1. Computational experiments were run on a PC with a 2.5 GHz Intel Core i5 processor and with 8 GB memory. For more information on the benchmark instances, including download-links, we refer the reader to DATA IN BRIEF.

To compare the efficiency of the procedures thirty giant tours were constructed using Path-Scanning-Random-Link developed by Belenguer et al. [2] for the MCARP and relaxing the capacity limit and route time restriction, as proposed by the authors. The giant tours were then partitioned using one of the four splitting procedures with the primary objective to minimise the number of required routes, and secondly to minimise the partition cost. Fig. 1 shows the CPU time taken by each procedure to partition the giant tours as a function of the size of the tours, n. Giant tours for *Cen-IF* instances are of sizes 1012, 2519 and 2755. All other points in the figure are for the *Lpr-IF* instances. All four splitting procedures show polynomial growth in their execution times as a factor of n. *Simple-Split* was extremely efficient, since it runs in $\mathcal{O}(n)$, splitting the *Cen-IF* tours with over 2500 tasks in less than 0.05 s. *Heuristic-Split* was also efficient, taking less than 0.5 s to partition the *Cen-IF* tours. Owing to their $\mathcal{O}(n^3)$ complexity, *Two-Phase-Split* and *Efficient-Split* took longer to partition tours. On all instances, *Efficient-Split* was substantially quicker than *Two-Phase-Split*, though on small instances with $n \leq 104$ the difference in performance may not be practically significant, with both procedures taking less than 0.5 s per partition. *Two-Phase-Split* took close to 60 s on the largest *Cen-IF* instances, whereas *Efficient-Split* took only 12 s.

Next we compared the solution quality of the partitioning procedures over the thirty Path-Scanning-Random-Link giant tours per instance. Since *Two-Phase-Split* and *Efficient-Split* produce the same optimal partitions their partition costs and resulting number of routes were used as a target. For the primary objective, to minimise the number of routes resulting from the partition, both

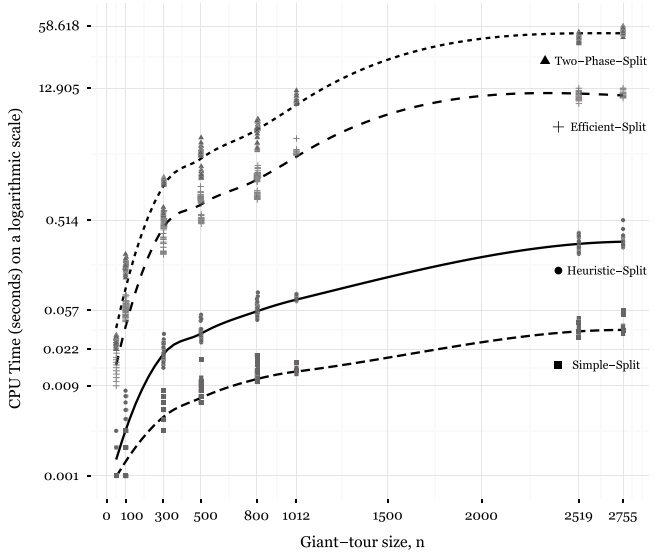

Fig. 1. Scatter plots and trend lines for giant tour size, n, versus partitioning time (in seconds, on a logarithmic scale) of four MCAPRTIF splitting procedures on 30 different giant tours per instance.

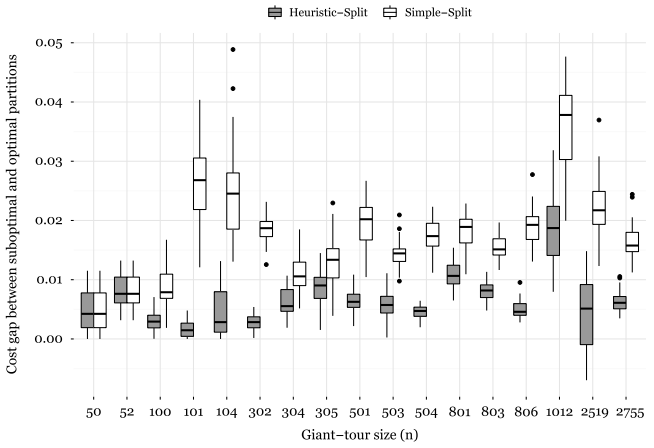

Fig. 2. Box-and-Whisker plot for the cost gap, Z_{gap}, between the partitions of *Simple-Split* and *Heuristic-Split*, and the optimal partitions of *Efficient-Split*.

Simple-Split and *Heuristic-Split* matched *Two-Phase-Split* on more than 79 of the 90 *Cen-IF* and on 431 of the 450 *Lpr-IF* giant tours. On the remaining tours, the near-optimal procedures required at most one additional route. To see how close the suboptimal partitions were to the optimal partitions, cost wise, the cost gap between the partitions was measured as $Z_{gap} = \frac{Z-Z^*}{Z^*}$ where Z is the cost of the final partition of the giant tour, partitioned using *Simple-Split* or *Heuristic-Split*, and Z^* is the cost of the optimal partition, as calculated through *Efficient-Split*. Results per problem instance are shown in Fig. 2 where each instance is labelled according to its giant tour size, n. Except for two instances, the partitions produced by *Heuristic-Split* were very close to optimal with an average cost gap of less than 1%. The cost gap range (maximum gap minus minimum gap) of the procedure is also small, being less than or close to 2%, which shows that it consistently produced near-optimal partitions. Negative cost gaps were observed for the procedure on some of the giant tours for *Cen-IF-c*. These were tours where *Heuristic-Split* partitions required an extra route over the optimal partitions, and the extra route resulted in a lower solution cost. Since minimising the number of routes was our primary objective, these partitions are suboptimal. *Simple-Split* matched *Heuristic-Split* on the two smallest giant tour instances. On all other instances its cost gap was larger, which was expected given its simple structure. The cost gap range of *Simple-Split* was also larger than *Heuristic-Split*, showing that its performance was not as consistent.

Our results show that as a factor of giant tour size, the execution time of *Heuristic-Split* is an order of magnitude lower to that of *Efficient-Split*, whose computation time is an order of magnitude lower than *Two-Phase-Split* (Fig. 1). Furthermore, *Heuristic-Split* consistently produced near-optimal partitions, whereas *Simple-Split* resulted in higher cost partitions and its performance varied per instance (Fig. 2). In time limited applications, methods linked with *Simple-Split* and *Heuristic-Split* may produce better results than *Two-Phase-Split* and *Efficient-Split* since their higher efficiency will allow the methods to evaluate and split more giant tours.

To test the hypothesis the following experiment was conducted using a multi-start Route-First–Cluster-Second constructive heuristic. Path-Scanning-Random-Link (PSRL) was again used for the Route-First phase and different splitting procedures were used for the Cluster-Second phase. Giant tours, representing different starts, were generated with PSRL and partitioned by the splitting procedure until an execution time-limit of 60 s was reached. The best partitioned solution from all the starts was then returned as the final solution. This was repeated for 30 experiments per instance and the mean number of routes, $\overline{K}$, and cost, $\overline{C}$, over the returned solutions were calculated. Summary results over the *Lpr-IF* and *Cen-IF* problem sets are shown in Table 1. Within the multi-start heuristic all the procedures performed the same in minimising the number of solution routes. The only difference in performance was in minimising solution cost. As expected, *Efficient-Split* performed better than *Two-Phase-Split* since it was able to make more starts in 60 s while still producing optimal partitions. On small instances the difference between *Efficient-Split* and *Two-Phase-Split* was less prominent. *Heuristic-Split* outperformed *Two-Phase-Split* on all the *Cen-IF* instances, and five of the fifteen *Lpr-IF* instances. *Simple-Split* had up to 30 and 800 times more starts than *Efficient-Split* and *Heuristic-Split*, respectively, on large *Cen-IF* and *Lpr-IF* instances, but could only match *Efficient-Split* on one and *Heuristic-Split* on three of the smallest *Lpr-IF* instances. Solution costs of *Heuristic-Split* are within one percent of *Efficient-Split* and it performed better on the *Cen-IF-c* instance. On all other instances *Efficient-Split* performed the best, making it the best constructive heuristic implementation on both small and large instances, with and without execution time-limits imposed.

For metaheuristics the effect of using *Simple-Split* or *Heuristic-Split* instead of *Efficient-Split* is difficult to analyse without embedding the procedures within a Memetic Algorithm (MA). The MA of Belenguer et al. [2] terminates after 26 000–40 000 evaluations, or when a one hour execution time limit is reached. If a similar time limit is imposed for the *Cen-IF* instances, and ignoring time required by the MA to perform other functions, such as Local-Search, *Efficient-Split* will be able to evaluate at most approximately 1350, 300, and 350 chromosomes for the three *Cen-IF* instances, respectively. *Heuristic-Split* will be able to evaluate at most 47 000, 12 000 and 12 500 chromosomes for the same problem instance. The impact of more chromosome evaluations based on near-optimal instead of optimal partitions can only be evaluated by implementing the procedures within an MA for the MCAPRTIF.

5. Conclusion

Tests showed that *Efficient-Split*, developed in this paper, was significantly faster than *Two-Phase-Split* that was based on an existing splitting procedure for the CAPRTIF. On small instances the difference in performance between the two procedures was less prominent, highlighting the need for tests to be performed on more realistically sized instances. With this being, to our knowledge, the first formal study on splitting procedures for the MCAPRTIF, the next step will be to extend metaheuristics for the CARP and MCARP

Table 1
Mean results over 30 experiments for different splitting procedures used in a multi-start Route-First–Cluster-Second constructive heuristic. Path-Scanning-Random-Link of Belenguer et al. [2] was used for the Route-First phase and for each experiment the heuristic was executed until a 60 s time-limit was reached.

Instance	n	*Efficient-Split*			*Two-Phase-Split*			*Heuristic-Split*			*Simple-Split*		
		$\overline{Z}$	$\overline{K}$	# starts	$\overline{Z}$	$\overline{K}$	# starts	$\overline{Z}$	$\overline{K}$	# starts	$\overline{Z}$	$\overline{K}$	# starts
Cen-IF-a	1012	240 696	9	25	241 511	9	6	241 100	9	934	244 564	9	16 701
Cen-IF-b	2755	602 695	22	6	612 302	22	1	603 543	22	155	606 484	22	5 602
Cen-IF-c	2519	533 331	19	7	544 645	19	1	532 083	19	181	538 188	19	5 506
Lpr-IF-a-01	52	13 594	1	6040	13 594	1	2128	13 684	1	82 554	13 684	1	296 150
Lpr-IF-a-02	104	28 392	1	1084	28 480	1	323	28 569	1	24 108	28 569	1	150 489
Lpr-IF-a-03	304	78 835	3	155	78 988	3	45	78 842	3	4 768	79 409	3	56 248
Lpr-IF-a-04	503	133 130	5	82	133 312	5	21	133 630	5	2 449	134 652	5	34 372
Lpr-IF-a-05	806	211 765	8	46	212 093	8	12	212 337	8	1 381	214 510	8	21 467
Lpr-IF-b-01	50	14 835	1	6877	14 835	1	2442	14 871	1	85 118	14 871	1	298 656
Lpr-IF-b-02	101	28 897	2	1193	28 928	2	370	28 926	2	25 053	29 412	2	149 217
Lpr-IF-b-03	305	80 222	3	151	80 290	3	46	80 419	3	4 735	80 678	3	55 991
Lpr-IF-b-04	501	132 254	5	80	132 485	5	23	132 366	5	2 494	134 318	5	34 532
Lpr-IF-b-05	801	220 638	8	49	221 079	8	13	221 175	8	1 442	223 750	8	21 274
Lpr-IF-c-01	50	18 734	1	7361	18 734	1	2386	18 765	1	85 118	18 765	1	298 211
Lpr-IF-c-02	100	36 596	2	1469	36 618	2	366	36 596	2	25 184	36 596	2	152 323
Lpr-IF-c-03	302	114 012	4	299	114 097	4	100	114 093	5	6 057	115 535	5	54 044
Lpr-IF-c-04	504	173 817	7	129	174 087	7	39	174 214	7	2 691	176 197	7	29 681
Lpr-IF-c-05	803	273 427	10	73	273 675	10	19	274 325	10	1 560	276 953	10	19 132

n: Giant route size $|\boldsymbol{S}|$; $\overline{Z}$: Mean solution cost over 30 experiments; $\overline{K}$: Mean number of solution routes over 30 experiments; # starts: mean number of starts completed within a 60 s time-limit.

that rely on splitting procedures, such as Memetic Algorithms, at which point the effect of using *Heuristic-Split* instead of *Efficient-Split* under time-limits can be formally tested.

Acknowledgements

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Number 87749) and by the South African Department of Trade and Industry (Grant Number 96415).

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.orl.2016.06.001.

References

[1] J.E. Beasley, Route first—cluster second methods for vehicle routing, Omega 11 (4) (1983) 403–408.
[2] J. Belenguer, E. Benavent, P. Lacomme, C. Prins, Lower and upper bounds for the mixed capacitated arc routing problem, Comput. Oper. Res. 33 (12) (2006) 3363–3383.
[3] Á. Corberán, G. Laporte, Arc Routing: Problems, Methods, and Applications, Vol. 20, SIAM, 2015.
[4] Á. Corberán, C. Prins, Recent results on arc routing problems: An annotated bibliography, Networks 56 (1) (2010) 50–69.
[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT-Press, 1990.
[6] G. Ghiani, F. Guerriero, G. Laporte, R. Musmanno, Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions, J. Math. Model. Algorithms 3 (3) (2004) 209–223.
[7] G. Ghiani, G. Improta, G. Laporte, The capacitated arc routing problem with intermediate facilities, Networks 37 (3) (2001) 134–143.
[8] P. Lacomme, C. Prins, W. Ramdane-Chérif, Competitive memetic algorithms for arc routing problems, Ann. Oper. Res. 131 (4) (2004) 159–185.
[9] T. Liu, Z. Jiang, N. Geng, A memetic algorithm with iterated local search for the capacitated arc routing problem, Int. J. Prod. Res. 51 (10) (2013) 3075–3084.
[10] C. Prins, P. Lacomme, C. Prodhon, Order-first split-second methods for vehicle routing problems: A review, Transp. Res. C 40 (2014) 179–200.
[11] K. Tang, Y. Mei, X. Yao, Memetic algorithm with extended neighborhood search for capacitated arc routing problems, IEEE Trans. Evol. Comput. 13 (5) (2009) 1151–1166.
[12] G. Ulusoy, The fleet size and mix problem for capacitated arc routing, European J. Oper. Res. 22 (3) (1985) 329–337.
[13] E. Willemse, W. Joubert, Benchmark dataset for undirected and mixed capacitated arc routing problems under time restrictions with intermediate facilities, Data Brief, in press.
[14] E. Willemse, W. Joubert, Constructive heuristics for the mixed capacity arc routing problem under time restrictions with intermediate facilities, Comput. Oper. Res. 68 (2016) 30–62.