Operations Research Letters 37 (2009) 392–398

Contents lists available at ScienceDirect

Operations Research Letters

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

A Branch-and-Price Algorithm for the Capacitated Arc Routing Problem with Stochastic Demands

Christian H. Christiansen, Jens Lysgaard[*], Sanne Wøhlk

CORAL - Centre for OR Applications in Logistics, Department of Business Studies, Aarhus School of Business, University of Aarhus, Fuglesangs Allé 4, 8210 Aarhus V, Denmark

ARTICLE INFO

Article history:
Received 6 July 2007
Accepted 27 May 2009
Available online 10 June 2009

Keywords:
Arc Routing
Stochastic programming
Logistics

ABSTRACT

We address the Capacitated Arc Routing Problem with Stochastic Demands (CARPSD), which we formulate as a Set Partitioning Problem. The CARPSD is solved by a Branch-and-Price algorithm, which we apply without graph transformation. The demand's stochastic nature is incorporated into the pricing problem. Computational results are reported.

1. Introduction and problem description

The Capacitated Arc Routing Problem (CARP) arises in several practical situations e.g. the planning of winter gritting, delivery of mail, or refuse collection. The CARP can be defined on a network in which a demand, either zero or positive, is associated with each edge. If the demand is positive, it is distributed along the edge. An unlimited fleet of identical vehicles is available for the operation, and all vehicles are situated at a single depot.

The CARP is then to find a collection of routes with minimum costs. The collection of routes must comply with the following constraints: (i) Each edge with positive demand is serviced exactly once; (ii) The vehicle capacity is not exceeded on any route; and (iii) All routes originate from and terminate at the depot.

However, for some practical situations the demand of each edge is better described by a random variable. An example of this could be stochastic snow depths due to drifting or stochastic amount of refuse from households and industries. Therefore we consider a variant of the CARP called Capacitated Arc Routing Problem with Stochastic Demands (CARPSD) in which all positive demands are described by random variables. All remaining aspects of the problem are assumed deterministic. The stochastic demands make the CARPSD different from the CARP with respect to two issues:

- At some point along a route, the actual accumulated demand might exceed the vehicle capacity. This event is denoted as a *failure*.
- The objective is to find a collection of routes with minimum *expected* cost.

If a failure occurs on some route, an action is taken in order to complete the originally planned route. The actual recourse action is based on a recourse strategy, which is decided upon *a priori* to the construction of the routes.

For the Capacitated Vehicle Routing Problem with Stochastic Demands (CVRPSD), which is the node routing counterpart of the CARPSD, two recourse strategies have been proposed in [1]. The first strategy (A) assumes that a customer's actual demand becomes known upon arrival at the customer. If the customer's actual demand then exceeds the remaining vehicle capacity, the vehicle is depleted and makes a back and forth trip to the depot to replenish. After the replenishment the originally planned route is continued. The second strategy (B) differs from strategy (A) by adding information. This added information means that at any point along a route, the next customer's actual demand is known. This allows the vehicle to skip customers with a zero actual demand.

However, when adapting these strategies to the CARPSD one important issue must be addressed, namely that a failure may occur anywhere on an edge, not only at the endpoints. This complicates the calculation of the recourse cost in two ways. Firstly, the serviced fraction of the edge (when the failure occurs) needs to be incorporated into the expected cost, and secondly, we need to decide via which endpoint the vehicle returns to the depot for replenishment. For each service edge, we assume that its demand is met according to a Poisson process. This implies that when partitioning a service edge into a number of segments, the demand of a segment follows a Poisson distribution. Indeed, the demand of the entire edge follows a Poisson distribution, and can be computed by adding the demands of the edge segments.

[*] Corresponding author.
E-mail addresses: chc@asb.dk (C.H. Christiansen), lys@asb.dk (J. Lysgaard), sanw@asb.dk (S. Wøhlk).

Furthermore, we adopt recourse strategy (A) from the CVRPSD and assume that the total actual demand of an edge is revealed gradually as the vehicle progresses along the edge. In case of failure we assume that the vehicle always makes a back and forth trip from the point of failure to the depot via the endpoint resulting in the least cost path to the depot. Note that this might incur that the vehicle is required to make a U-turn on some edge. We assume that these U-turns are possible, and that they can be done at zero cost.

In this article we develop an exact method for solving the CARPSD. In particular, we formulate the CARPSD as a Set Partitioning Problem and develop a Branch-and-Price algorithm, in which the pricing is done by dynamic programming.

Our work is motivated by the fact that no exact algorithms have yet been developed for the CARPSD, and our problem definition is more generic than the ones previously studied. Most important is that we do not impose restrictions regarding the point of failure.

Further motivation is found in the literature regarding the CARP. Exact methods for the CARP, which address the CARP directly are generally capable of solving instances with up to 30 service edges which is less than the best algorithms based on graph transformation [2]. However, little research has yet considered the opportunities which lie in modifying the Branch-and-Cut-and-Price algorithm developed for the Capacitated Vehicle Routing Problem (CVRP) in [3] and applying it directly to the CARP or CARPSD.

This article delivers the first step in this process by formulating the CARPSD as a Set Partitioning Problem and applying pricing directly on the CARPSD, without graph transformation. This modeling approach is further motivated by the promising development of a Branch-and-Price algorithm for the CVRPSD in [4].

The remainder of the article is organized as follows. Section 2 contains a literature review. Section 3 introduces the needed notation and provides a formal model description. Section 4 gives a description of the proposed solution procedure. Sections 5 and 6 contain a detailed explanation of the pricing procedure and the branching rule, respectively. Computational results are reported in Section 7.

2. Related literature

The CARP was first suggested in [5]. Since then, many heuristics have been suggested for solving the problem, and both combinatorial and LP based lower bounding procedures have been presented. For recent reviews of the CARP and its variations, see [6–9].

The first exact algorithm for the CARP is based on Branch-and-Bound [10] and the authors are able to solve a special set of CARP instances containing from 15 to 50 demand edges to optimality. Both [11] and [2] propose an exact solution strategy for the CARP which is based on transforming the problem into a CVRP, which is in turn solved by state-of-the-art algorithms. Both papers report computational results that are highly competitive to the existing ones. Branch-and-Cut-and-Price based algorithms, which address the CARP directly, are suggested for the CARP in [12] and [13]. The latter utilizes the fact that CARP networks are very sparse since they often represent street networks. Two versions of the related pricing problem are explored. The first one allows non-elementary routes and is solved by dynamic programming, whereas the second one only allows elementary routes and is solved with a Cut-and-Branch algorithm.

Many extensions of the CARP have been suggested, most of which arise from real world applications. These include multiple depots, time windows for allowed service, alternative objective functions, and stochastic demands. The latter will be further explored in this paper.

The stochastic versions of the CARP have received very little attention. To the best of our knowledge CARPSD was first considered in [14] and further extended in [15]. The CARPSD was not approached directly. Instead the scope was to evaluate the robustness of solutions to the CARP if the demands were in fact stochastic, and how this robustness could be improved. In particular, [15] contains an application of a hybrid genetic heuristic to the CARP. Different solutions were obtained by varying the vehicle capacity in each run of the heuristic. The solutions obtained were then evaluated by means of simulation studies.

In [16] the CARPSD with Normal distributed demands was approached directly. For each edge, the Normal distribution describing the demand was truncated to avoid negative demands and demands that exceeded the vehicle capacity. The problem was further restricted, since a route could only fail once, and if a failure occurred on a route it would always be immediately before serving the last edge. The solution method was an application of a modified version of the memetic algorithm originally developed in [17].

Because the research body concerning the CARPSD is very sparse, we turn our attention to the modeling approaches found in the literature concerning the CVRPSD.

The CVRPSD has received some attention since its first appearance in [18]. Since then research on the CVRPSD has developed three different modeling approaches concerning the incorporation of stochastic demands:

(1) Constraint modeling. This approach incorporates the cost of failing indirectly by imposing the constraint that the risk of failure on any route should not exceed some predetermined threshold. This constraint can be incorporated by adjusting the vehicle capacity, and the CVRPSD can then be solved as a standard CVRP [19–21].

(2) Recourse modeling. These models incorporate a cost of failure explicitly into the objective function, and the CVRPSD is then modeled as a two-stage stochastic program with simple recourse (see e.g. [4,22,23]). Two exact algorithms have been developed for the CVRPSD modeled as a two-stage recourse program. The first is the L-shaped algorithm originally developed in [24], and the second is the Branch-and-Price algorithm developed in [4].

(3) Decision modeling approach. This modeling approach considers the single vehicle version of the CVRPSD. The approach takes into account the new information that becomes available as the vehicle progresses along a route. The purpose of including this information into the design of the recourse strategy is to enable more advanced recourse strategies than the previously proposed strategies (A) and (B). However, the complexity of the modeling approach has so far prohibited the implementation of exact solution procedures [25–27].

In this article we model the CARPSD as a two-stage stochastic program with simple recourse.

3. Notation and model formulation

We assume that the input data of a particular CARPSD instance is given by an undirected connected graph $G^0(V, E^0)$ with vertex set $V = \{0, \dots, n\}$ and edge set E^0. Vertex 0 denotes the depot. Each edge $e \in E^0$ has a positive length. A particular subset $E_s \subseteq E^0$ of the edges must be serviced; these are called *service edges*. Each service edge $e \in E_s$ has a non-negative stochastic demand represented by the random variable q_e, and we assume that q_e follows a Poisson distribution with an expected value of $E[q_e]$. Edges in $E^0 \setminus E_s$ have no demand and may or may not be traversed. The requirement for servicing each service edge means that the edge must be traversed at least once, and exactly one of these traversals satisfies the demand of the edge; any additional traversals of the edge is done as if the edge had no demand.

Without loss of generality we assume that $|V|$ is minimal in the sense that each vertex in $V \setminus \{0\}$ is the end vertex of at least one service edge. In most applications, G^0 will be relatively sparse.

Based on the input graph G^0, we construct a multigraph $G(V, E)$ which forms the basis for our formulation of the CARPSD. The vertex set in G is the same as that in G^0. The set $E_s \subset E$ of service edges is also the same as in G^0. In addition, we add a set E_t of *travel edges* to E, in the following way.

Between each pair of vertices in V, we construct a *travel edge* which represents the shortest path in G^0 between the two end vertices of the edge. We note that such a shortest path may contain one or more service edges in G^0; in any case, traversing the edge $e \in E_t$ in G represents the traversal of the corresponding path in G^0 *without* servicing any of its edges.

With each $e \in E$, we associate a length of ℓ_e of the edge. For each $e \in E_t$ the parameter ℓ_e represents the length of the shortest path in G^0 between the two end vertices of the edge, whereas for each $e \in E_s$, the parameter ℓ_e is equal to the length of the corresponding service edge in G^0.

As a result of this construction, we have that between any pair of vertices in G, there is either a travel edge or both a travel edge and a service edge. In the latter case, the service edge is at least as long as the travel edge. By construction, the subgraph induced by E_t satisfies the triangle inequality.

For notational convenience we introduce an artificial variable $q_e = 0$ (implying $E[q_e] = 0$) for each travel edge e. We denote the vehicle capacity Q.

By convention, we write any edge $\{i, j\} \in E$ such that $i < j$. To avoid ambiguities we will, where necessary, use an extended notation for the individual edges. Specifically, we will let $\{i, j\}^s$ denote the *service* edge between i and j, and let $\{i, j\}^t$ denote the *travel* edge between i and j. For notational simplicity we let d_i denote the length of the shortest path in G^0 between 0 and $i, \forall i \in V$, with $d_0 = 0$. For any $S \subset V$ we denote by $E(S)$ the set of edges with both endpoints in S and by $\delta(S)$ the set of edges with exactly one endpoint in S.

A route is defined as a sequence of edges $(z_1, \dots, z_k)$ where $z_2, \dots, z_{k-1} \in E \setminus \delta(\{0\})$ and $z_1, z_k \in \delta(\{0\})$. If the route satisfies $\sum_{e=z_1,\dots,z_k} E[q_e] \leq Q$ it is said to be *feasible*, and if all $e \in \{z_1, \dots, z_k\} \cap E_s$ are different, the route is defined as *elementary*.

The expected cost of a route r is denoted as c_r. Furthermore, let $\mathfrak{R}$ denote the set of all feasible and elementary routes, and let α_{er} be a parameter taking the value 1, if edge e is included on route r and 0 otherwise. Finally, let x_r be a binary variable taking the value 1, if route r is included in the selected route collection and 0 otherwise. This leads to the following Set Partitioning formulation of the CARPSD:

(P_{org})

$$\text{min:} \quad \sum_{r \in \mathfrak{R}} c_r x_r \tag{1}$$

$$\text{s.t.:} \quad \sum_{r \in \mathfrak{R}} \alpha_{er} x_r = 1 \quad \forall e \in E_s \tag{2}$$

$$x_r \in \{0, 1\} \; \forall r \in \mathfrak{R}. \tag{3}$$

Objective (1) is to minimize the total expected cost of the route collection chosen. The constraint set (2) ensures that all $e \in E_s$ are serviced, and (3) is the binary constraint on the x_r variables.

The key to incorporating the stochastic demands into P_{org} is through the computation of c_r. Because recourses are conducted in *addition* to traveling, c_r decomposes into two elements. These are denoted as $C_r^D = \sum_{e=z_1,\dots,z_k} \ell_e$ which is the traveled distance along route r, and C_r^F which is the expected failure cost of route r. The complicating part is C_r^F, which we will now discuss in further detail. To find the expected failure cost of an edge we need to compute two elements.

The first element is the cost of failing on edge $e = \{i, j\}^s$. Let $0 \leq f \leq 1$ be a number such that $f\ell_e$ is the distance between i and the point of failure, implying that $(1-f)\ell_e$ is the distance between the point of failure and j. We then compute the recourse cost for any $e \in E_s$ and any point of failure on the edge as follows:

$$\text{REC}(e, f) = 2 \min\{f\ell_e + d_i;\, (1-f)\ell_e + d_j\}. \tag{4}$$

The second element is the probability of failing on edge $e = \{i, j\}^s$. However, before discussing this in further detail, we need to establish an important property regarding the stochastic demands.

Proposition 1. *For computing the probability of failing on a given edge, we need only to consider the total expected demand on the path from the depot to that edge, and not how this total expected demand is divided between the edges on the path.*

Proof. The Poisson distributed demands q_e possess an accumulative property, i.e. if $q_e \sim \text{Poisson}(E[q_e])$ and $q_k \sim \text{Poisson}(E[q_k])$ then $(q_e + q_k) \sim \text{Poisson}(E[q_e] + E[q_k])$. Because only the *total* expected demand is relevant for determining the probability distribution, we can disregard how the demands are distributed among the service edges on the path. □

The result obtained in Proposition 1 is analogous to the result in Proposition 1 in [4] for the corresponding node routing problem.

We partition each service edge into edge segments of equal lengths. We let L_e denote the number of segments that e has been partitioned into. Note that because of our assumption regarding the nature of the demand on an edge, the demand of each edge segment is indeed a Poisson distributed random variable with an expected value of $E[q_e]/L_e$. We number the segments $v = 1, \dots, L_e$ starting from the point of service (either i or j) to the ending point of service, i.e. when the vehicle services segment v it has already serviced $v - 1$ segments on e. For any $\lambda > 0$, let $\xi(\lambda)$ be a Poisson distributed random variable with an expected value of λ. Further, we let $\xi(0) = 0$.

The expected number of failures in the vth segment on edge $e \in E_s$, on a path from the depot up to and including edge e with an accumulated expected demand of λ, can be computed as follows, where $u \in Z^+$ represents the failure number in segment v:

$$\text{FAIL}(\lambda, e, v) = \sum_{u=1}^{\infty} \left[P\left(\xi\left(\lambda - (L_e - v + 1)\frac{E[q_e]}{L_e}\right) \leq uQ\right) - P\left(\xi\left(\lambda - (L_e - v)\frac{E[q_e]}{L_e}\right) \leq uQ\right)\right]. \tag{5}$$

For practical computations ∞ is replaced by a sufficiently large positive integer. By letting $L_e \to \infty$, we can find the exact contribution $\text{EFC}(\lambda, e)$ from $e \in E_s$ to the expected failure cost on a path from the depot up to and including e with an expected accumulated demand of λ. If the edge is traversed from i to j, we obtain:

$$\text{EFC}(\lambda, e) = \sum_{v=1}^{L_e} \text{FAIL}(\lambda, e, v)\,\text{REC}\left(e, \frac{v}{L_e}\right). \tag{6}$$

If, instead, the edge is traversed from j to i, we use $\text{REC}(e, 1 - v/L_e)$ instead of $\text{REC}(e, v/L_e)$ in (6). For notational convenience, we define $\text{EFC}(0, e) = 0 \; \forall e \in E_t$.

The final step of finding the exact expected cost of the route $(z_1, \dots, z_k)$ is as follows:

$$c_r = \sum_{h=1}^{k} \left(\ell_{z_h} + \text{EFC}\left(\sum_{p=z_1,\dots,z_h} E[q_p], z_h \right) \right). \tag{7}$$

We note that the direction of traversal is taken into account when calculating the individual EFC-values.

The results obtained in (6) and (7) are encouraging, because they give us the possibility to approximate $\mathrm{EFC}(\lambda, e)$ arbitrarily closely, by choosing an appropriate value of L_e. This in turn also means that we can approximate c_r arbitrarily close. We use this result to establish lower and upper bounds for c_r. We use these bounds to ensure sufficient quality of the reported solution.

In particular, we compute an upper bound $\overline{\mathrm{REC}}(e, v/L_e)$ and a lower bound $\underline{\mathrm{REC}}(e, v/L_e)$ on $\mathrm{REC}(e, v/L_e)$. If an edge $e = \{i, j\}^s$ is traversed from i to j, we obtain:

$$\underline{\mathrm{REC}}\left(e, \frac{v}{L_e}\right) = 2 \min\left\{d_i + (v-1)\frac{\ell_e}{L_e}; d_j + (L_e - v)\frac{\ell_e}{L_e}\right\}. \tag{8}$$

If, instead, the edge is traversed from j to i, we obtain:

$$\underline{\mathrm{REC}}\left(e, \frac{v}{L_e}\right) = 2 \min\left\{d_i + (L_e - v)\frac{\ell_e}{L_e}; d_j + (v-1)\frac{\ell_e}{L_e}\right\}. \tag{9}$$

In both cases an upper bound can be calculated as follows:

$$\overline{\mathrm{REC}}\left(e, \frac{v}{L_e}\right) = \underline{\mathrm{REC}}\left(e, \frac{v}{L_e}\right) + 2\frac{\ell_e}{L_e}. \tag{10}$$

Using these bounds it is straightforward to compute a lower bound $\underline{\mathrm{EFC}}(\lambda, e)$ on $\mathrm{EFC}(\lambda, e)$, by replacing REC in (6) with its lower bound. In the same way an upper bound $\overline{\mathrm{EFC}}(\lambda, e)$ is established.

The final step is to find a lower bound $\underline{c}_r$ (upper bound $\overline{c}_r$), by simply replacing EFC in (7) with $\underline{\mathrm{EFC}}$ ($\overline{\mathrm{EFC}}$). The tightness of $\overline{c}_r$ and $\underline{c}_r$ depends on the number of segments each edge is divided into, and the tightness comes at the expense of the effort needed to compute these bounds.

4. Solution procedure

Complete enumeration of the set $\mathfrak{R}$ is generally prohibitively time consuming for any reasonable sized problem. Therefore we enumerate $\mathfrak{R}$ implicitly and solve P_{org} by Branch-and-Price.

The master problem is constructed from P_{org} by relaxing the integrality constraints (3) and changing the constraints (2) from '=' to '$\geq$'. Furthermore, we expand the set $\mathfrak{R}$ by adding all feasible non-elementary routes without *1-cycles*. That is, we permit a service edge $e \in E_s$ to be serviced more than once on a route, but only if at least one other edge is serviced in between any two traversals of edge e. The resulting expanded set, $\mathfrak{R}'$, thereby consists of all feasible routes without 1-cycles. We denote by a_{er} the number of times edge e is serviced on route r, and we replace the exact expected cost c_r by its lower bound $\underline{c}_r$. These modifications result in the following master problem:

(P_M)

$$\text{min:} \sum_{r \in \mathfrak{R}'} \underline{c}_r x_r \tag{11}$$

$$\text{s.t.:} \sum_{r \in \mathfrak{R}'} a_{er} x_r \geq 1 \quad \forall e \in E_s \tag{12}$$

$$x_r \geq 0 \; \forall r \in \mathfrak{R}'. \tag{13}$$

We solve P_M for a set of columns $COL \subset \mathfrak{R}'$, which is expanded at each iteration of our algorithm. Initially COL contains $|E_s|$ columns each representing a route which services exactly one service edge. By solving P_M we obtain a dual vector π related to constraints (12). We use this vector to find new columns with negative expected reduced cost. In particular, we seek a route that traverses service edges $(z_1, \ldots, z_k)$ such that:

$$\underline{c}_r - \sum_{i=1}^{i=k} \pi_{z_i} < 0. \tag{14}$$

The identified columns satisfying (14) are then added to COL, and P_M is resolved. This pricing is continued until no more columns with a negative expected reduced cost can be found. If the solution to P_M is integer and no more columns satisfying (14) can be found, the solution is indeed optimal for P_M. If the solution is fractional we resort to branching, and we continue branching and pricing until a globally optimal integer solution to P_M is reached.

An optimal integer solution to P_M is also optimal for P_{org} within the precision level determined by the number of segments that each edge $e \in E_s$ is divided into. Indeed, the permission to service an edge more than once in P_M will not be an advantage in any integer solution. The reason for this is that any service edge $\{i, j\}^s$ can be replaced by the parallel travel edge $\{i, j\}^t$ without increasing the cost of the route.

To ensure that the number of branching nodes solved is minimal, we apply the "best bound" rule when selecting the next subproblem during the tree search.

5. The pricing procedure

Before going into details regarding the pricing procedure, we need to establish an important proposition regarding an optimal route:

Proposition 2. *At least one optimal solution to P_{org} is characterized by only having routes that do not contain consecutive travel edges.*

Proof. If some route r on $G(V, E)$ with expected cost of c_r contains two consecutive travel edges $\{i, j\}^t$ and $\{j, k\}^t$, a new route with an expected cost of c'_r can be constructed, without changing the set of service edges, by replacing $\{i, j\}^t$ and $\{j, k\}^t$ by the single travel edge $\{i, k\}^t$. Because of our construction of $G'(V, E_t)$ it holds that $c'_r \leq c_r$. □

Proposition 2 is essential for our Branch-and-Price algorithm, because we can disregard all paths containing consecutive travel edges, making our pricing procedure run in $O(n^2 Q)$ time.

We emphasize the importance of knowing the direction in which a service edge is traversed when calculating the associated expected failure cost, and therefore we extend our notation slightly. We denote the expected failure cost of service edge $e = \{i, j\}^s$ as $\mathrm{EFC}^{i,j}(\lambda, e)$ if e is traversed from i to j, and we denote the expected failure cost of e as $\mathrm{EFC}^{j,i}(\lambda, e)$ if e is traversed from j to i.

We formulate the pricing problem as a shortest path problem on a specially constructed acyclic graph $G_p(V_p, A_p)$. For the purpose of applying dynamic programming we assume that $E[q_e] \in Z^+ \forall e \in E_s$. The vertex set V_p is the union of the disjoint subsets $V_p = V_s \cup V_t \cup \{v(0, 0)\}$. For $\lambda = 1, \ldots, Q$ and $i = 0, \ldots, n$, vertex $v^s(\lambda, i) \in V_s$ represents all paths with an accumulated expected demand of λ ending at vertex $i \in V$ immediately after serving some edge in E_s. For $\lambda = 0, \ldots, Q$ and $i = 0, \ldots, n$ (where $\lambda > 0 \vee i > 0$), vertex $v^t(\lambda, i) \in V_t$ represents all paths with an accumulated expected demand of λ ending at vertex $i \in V$ immediately after traversing an edge in E_t. Vertex $v(0, 0)$ is the starting point of all paths.

For each $i = 1, \ldots, n$ we add an arc from $v(0, 0)$ to $v^t(0, i)$ and we set the cost of this arc equal to d_i. For each $e = \{0, i\}^s$ we add an arc from $v(0, 0)$ to $v^s(E[q_e], i)$ and we set the cost equal to $\ell_e + \underline{\mathrm{EFC}}^{0,i}(E[q_e], e) - \pi_e$. For each $\lambda = 0, \ldots, Q$ and for each $e = \{i, j\}^t$ we add two arcs $(v^s(\lambda, i), v^t(\lambda, j))$ and $(v^s(\lambda, j), v^t(\lambda, i))$ and we set the cost of these arcs equal to ℓ_e. Furthermore, for each $\lambda = 1, \ldots, Q$ and for each $e = \{i, j\}^s$ we add arc $(v^t(\lambda, i), v^s(\lambda + E[q_e], j))$ with cost $\ell_e + \underline{\mathrm{EFC}}^{i,j}(\lambda + E[q_e], e) - \pi_e$, and arc $(v^t(\lambda, j), v^s(\lambda + E[q_e], i))$ with cost $\ell_e + \underline{\mathrm{EFC}}^{j,i}(\lambda + E[q_e], e) - \pi_e$.

Finally, for each $\lambda = 1, \ldots, Q$ and for each $i = 1, \ldots, n$ we add an arc $(v^s(\lambda, i), v^t(\lambda, i))$ of length 0. These extra arcs are added to allow routes that only contain service edges.

Finding the shortest path from $v(0, 0)$ to each vertex in V_s effectively solves the pricing problem. This can be done in $O(n^2 Q)$ time, also when applying 1-cycle elimination.

Table 1
Computational results.

Inst.	Vertices	Edges	Root LB	Time root	Obj. Val.	Total time	Tree size
A10A	10	11	103.51	0.03	103.93*	0.04	11
A10B	10	11	72.31	0.02	74.89*	0.60	247
A10C	10	11	53.43	0.05	56.12*	1.32	131
A10D	10	11	47.00	0.1	50*	4.75	103
A13A	13	22	204.56	0.03	207.32*	0.38	159
A13B	13	22	126.98	0.05	132.52		
A13C	13	22	93.96	0.16	96.19		
A13D	13	22	91.00	0.45	91.30		
A15A	15	22	180.62	0.03	184.82*	0.29	107
A15B	15	22	122.93	0.06	127.38*	13.47	1 403
A15C	15	22	98.43	0.16	104.11		
A15D	15	22	92.00	0.35	96.45		
A20A	20	29	296.80	0.04	298.66*	0.57	147
A20B	20	29	196.49	0.08	202.69		
A20C	20	29	129.24	0.35	133.94		
A20D	20	29	119.56	0.91	121.62		
A24A	24	37	320.29	0.05	327.25*	926.91	79 179
A24B	24	37	212.16	0.15	216.62		
A24C	24	37	159.19	0.74	160.78		
A24D	24	37	150.03	1.57	150.09		
A27A	27	35	452.31	0.05	460.64*	32.18	3 995
A27B	27	35	299.36	0.13	310.86		
A27C	27	35	222.13	0.49	229.56		
A27D	27	35	204.36	1.6	207.09		
A31A	31	52	861.50	0.05	869.38		
A31B	31	52	520.90	0.19	527.77		
A31C	31	52	336.98	0.9	340.66		
A31D	31	52	290.29	2.34	291.34		
A40A	40	63	1 105.52	0.09	1 106.1*	0.44	39
A40B	40	63	624.84	0.31	630.02		
A40C	40	63	417.28	1.76	418.40		
A40D	40	63	365.54	8.2	365.83		
gdb1	12	22	327.61	0.03	341.28		
gdb2	13	23	359.96	0.04	368.25		
gdb3	8	28	287.95	0.02	301.99		
gdb4	9	36	308.61	0.02	319.82*	34.59	7 659
gdb5	8	11	414.30	0.04	420.32*	126.15	19 143
gdb6	11	22	325	0.02	331.04*	18.46	4 063
gdb7	11	33	339.87	0.03	352.24		
gdb8	11	44	374.91	0.17	378.73		
gdb9	11	55	332.37	0.27	333.53		
gdb10	12	26	273.82	0.05	278.80		
gdb11	12	22	375.12	0.77	375.56		
gdb12	11	19	487.64	0.04	493.47*	0.50	95
gdb13	13	26	536.16	0.06	539.59		
gdb14	12	22	102.78	0.04	102.81*	0.06	7
gdb15	12	22	58.12	0.07	58.12*	0.47	39
gdb16	27	46	125.72	0.12	125.87		
gdb17	27	51	87.39	0.14	87.40		
gdb18	12	25	166.54	0.59	166.54*	2.79	27
gdb19	22	45	56.90	0.01	57.1*	0.02	7
gdb20	10	28	116.98	0.06	118.86		
gdb21	7	21	155.29	0.14	155.46		
gdb22	7	21	201.24	0.26	201.26		
gdb23	8	28	239.24	0.65	239.24		
kshs1	8	15	14 566.22	0.05	14 912.13*	15.39	2 147
kshs2	10	15	9 173.85	0.07	9 842.52		
kshs3	6	15	8 569.52	0.04	9 020.03		
kshs4	8	15	11 760.59	0.31	12 283.24*	45.87	6 563
kshs5	8	15	10 627.21	0.04	11 394.01		
kshs6	9	15	9 491.31	0.09	10 164.08		

6. Branching rule

In [28] branching on time windows for the Capacitated Vehicle Routing Problem with Time Windows was introduced and applied with success. A similar branching based on the capacity resource was suggested for and applied to the CVRPSD in [4]. We adapt this rule of branching on the capacity resource to the CARPSD.

If an optimal solution of P_M contains decision variables with fractional values, at least one service edge is serviced more than once with different accumulated expected demands, possibly on the same route.

Based on this observation we apply the following branching rule: Consider a fractional solution in which edge $e \in E_s$ is serviced on two paths with an accumulated expected demand of λ_1 and λ_2 respectively, where $\lambda_1 < \lambda_2$. We then choose a threshold value $\lambda_1 < \kappa_e \leq \lambda_2$ and we construct two new subproblems. In the first subproblem we prohibit paths serving e with an accumulated expected demand of $\lambda \geq \kappa_e$. In the second subproblem we prohibit all paths serving e with an accumulated expected demand of $\lambda < \kappa_e$. Clearly, this branching renders the original fractional solution infeasible in both the constructed subproblems.

Table 2
Computational results (continued).

Inst.	Vertices	Edges	Root LB	Time root	Obj. Val.	Total time	Tree size
egl-e4-A	77	98	6 626.21	60.37	6628.37		
egl-e4-B	77	98	9 526.68	17.45	9532.59		
egl-e4-C	77	98	12 909.10	6.78	12916.35		
bccm1A	24	39	146.00	3.06	146.00		
bccm1B	24	39	152.19	1.00	153.80		
bccm1C	24	39	250.54	0.14	254.74		
bccm2A	24	34	206.24	2.41	207.34		
bccm2B	24	34	245.49	0.89	247.42		
bccm2C	24	34	532.73	0.08	535.79*	24.14	1387
bccm3A	24	35	70.42	2.69	70.53		
bccm3B	24	35	82.96	0.62	84.03		
bccm3C	24	35	158.84	0.09	160.63		
bccm4A	41	69	361.17	66.51	361.17		
bccm4B	41	69	378.69	37.18	378.70		
bccm4C	41	69	410.15	18.20	410.67		
bccm4D	41	69	551.64	1.96	552.68		
bccm5A	34	65	388.16	27.59	388.16		
bccm5B	34	65	377.41	10.92	377.41		
bccm5C	34	65	452.84	7.28	452.84		
bccm5D	34	65	657.77	1.08	659.77		
bccm6A	31	50	195.59	5.88	195.64		
bccm6B	31	50	210.97	1.41	211.84		
bccm6C	31	50	330.60	0.21	333.28		
bccm7A	40	66	249.71	25.50	249.72		
bccm7B	40	66	253.55	15.26	253.60		
bccm7C	40	66	323.08	0.81	324.09		
bccm8A	30	63	357.12	18.66	357.12		
bccm8B	30	63	374.82	9.04	374.82		
bccm8C	30	63	558.64	0.66	559.74		
bccm9A	50	92	282.08	245.03	282.08		
bccm9B	50	92	289.00	135.99	289.00		
bccm9C	50	92	297.71	97.81	297.71		
bccm10A	50	97	383.89	199.49	383.89		
bccm10B	50	97	394.21	135.17	394.21		
bccm10C	50	97	410.21	95.89	410.22		
bccm10D	50	97	527.43	12.21	527.64		

7. Implementation and computational results

To evaluate our Branch-and-Price algorithm we have run several test instances from the literature. We define an ϵ-optimal solution as a feasible integer solution x' satisfying that $\sum_{r\in\mathfrak{R}} \bar{c}_r x' \leq (1+\epsilon)Z^*$, where Z^* denotes the true optimal objective value. To ensure that we reach an ϵ-optimal solution, we set for each $e = \{i,j\}^s$ the value of L_e such that $\ell_e + \overline{EFC}^{i,j} \leq (1+\epsilon)(\ell_e + \underline{EFC}^{i,j})$. For ease of implementation we use the same L_e for each service edge. In our experiments we set $\epsilon = 0.01$.

We have tested our algorithm on instances which are constructed by modifying test instances for the CARP. From http://www.uv.es/~belengue/carp.html we chose all ‘gdb’, ‘kshs’, ‘bccm’ instances, and those ‘eglese’ instances with $|V| < 80$ in which each $i \in V$ is an endpoint of at least one service edge. Moreover, from http://www.hha.dk/~sanw we chose all ‘A’ instances [8].

For each service edge we set the expected demand equal to the original demand of that edge in the CARP instance. All other aspects of the test instances are unchanged. All instances were run on a Pentium Centrino 1500 MHz computer with 480 MB of RAM, using a time limit of 30 min.

The test results are presented in Tables 1 and 2. Columns 1, 2 and 3 show the instance name, the number of vertices, and the number of service edges. Columns 4 and 5 contain the lower bound obtained at the root node, and the time in seconds spent in the root node, respectively. Column 6 contains the best lower bound obtained within the time limit. If the result is marked by ‘*’ then it is an 1%-optimal solution. This means that the value reported is a lower bound, which is less than 1% from the optimal objective value. For instance, the true optimal objective value of the instance ‘A10D’ is contained in the interval [50; 50.5]. Results in columns 7 and 8 are only reported for those instances where a 1%-optimal solution is found. Column 7 shows the total time in seconds, and column 8 shows the number of nodes in the search tree.

The largest instance we have solved is the ‘A40A’ which includes 40 vertices and 69 service edges. Our pricing procedure has proven most successful on instances that are characterized by tight capacity constraints. This is well in line with the general experience regarding Branch-and-Price algorithms for the CVRP and CVRPSD [4].

The bounds found at the root node are generally very tight. Even so, the branching trees can become very large. Good examples of this are the instances ‘A24A’ and ‘gdb5’.

Acknowledgments

We are grateful to two anonymous referees whose helpful comments led to an improved paper.

References

[1] D.J. Bertsimas, A vehicle routing problem with stochastic demand, Operations Research 40 (1992) 574–585.
[2] H. Longo, M. Poggi de Aragão, E. Uchoa, Solving capacitated arc routing problems using a transformation to the CVRP, Computers & Operations Research 33 (2006) 1823–1837.
[3] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R.F. Werneck, Robust branch-and-cut-and-price for the capacitated vehicle routing problem, Mathematical Programming 106 (2006) 491–511.
[4] C.H. Christiansen, J. Lysgaard, A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands, Operations Research Letters 35 (2007) 773–781.
[5] B.L. Golden, R.T. Wong, Capacitated arc routing problems, Networks 11 (1981) 305–315.
[6] D. Ahr, Contributions to multiple postmen problems, Ph.D. Thesis, University of Heidelberg, Germany, 2004.

[7] M. Dror (Ed.), Arc Routing: Theory, Solutions and Applications, Kluwer Academic Publishers, Boston, 2000.

[8] S. Wøhlk, Contributions to arc routing, Ph.D. Thesis, University Press of Southern Denmark, Odense, 2006.

[9] S. Wøhlk, A decade of capacitated arc routing, in: B.L. Golden, S. Raghavan, E.A. Wasil (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, 2008.

[10] R. Hirabayashi, N. Nishida, Y. Saruwatari, Tour construction algorithm for the capacitated arc routing problem, Asia-Pacific Journal of Operational Research 9 (1992) 155–175.

[11] R. Baldacci, V. Maniezzo, Exact methods based on node-routing formulations for undirected arc-routing problems, Networks 47 (2006) 52–60.

[12] J.M. Belenguer, E. Benavent, D. Gómez-Cabrero, Cutting plane and column generation for the capacitated arc routing problem, in: ORP3 Meeting, Valencia, 2005.

[13] A.N. Letchford, A. Oukil, Exploiting sparsity in pricing routines for the capacitated arc routing problem, Computers & Operations Research 36 (2009) 2320–2327.

[14] G. Fleury, P. Lacomme, C. Prins, W. Ramdane-Chérif, Robustness evaluation of solutions for the capacitated arc routing problem, in: Conference, AI Simulation and Planning in High Autonomy Systems, ISBN: 1-56555242-3, 2002, pp. 290–295.

[15] G. Fleury, P. Lacomme, C. Prins, W. Ramdane-Chérif, Improving robustness of solutions to arc routing problems, Journal of the Operational Research Society 56 (2005) 526–538.

[16] G. Fleury, P. Lacomme, C. Prins, Evolutionary algorithms for stochastic arc routing problems, in: G.R. Raidl, F. Rothlauf, G.D. Smith, G. Squillero, S. Cagnoni, J. Branke, D.W. Corne, R. Drechsler, Y. Jin, C.G. Johnson (Eds.), Applications of Evolutionary Computing, Springer-Verlag, Berlin and Heidelberg, 2004, pp. 501–512.

[17] P. Lacomme, C. Prins, W. Ramdane-Chérif, Competitive memetic algorithms for arc routing problems, Parallel Computing 30 (2004) 377–387.

[18] F.A. Tillman, The multiple terminal delivery problem with probabilistic demands, Transportation Science 3 (1969) 192–204.

[19] C. Bastian, A.H.G. Rinnooy Kan, The stochastic vehicle routing problem revisited, European Journal of Operational Research 56 (1992) 407–412.

[20] M. Dror, P. Trudeau, Stochastic vehicle routing with modified savings algorithm, European Journal of Operational Research 23 (1986) 228–235.

[21] W.R. Stewart, B.L. Golden, Stochastic vehicle routing: A comprehensive approach, European Journal of Operational Research 14 (1983) 371–385.

[22] M. Dror, G. Laporte, P. Trudeau, Vehicle routing with stochastic demands: Properties and solution frameworks, Transportation Science 23 (1989) 166–176.

[23] G. Laporte, F.V. Louveaux, L. Van Hamme, An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands, Operations Research 50 (2002) 415–423.

[24] G. Laporte, F.V. Louveaux, The integer L-shaped method for stochastic integer programs with complete recourse, Operations Research Letters 13 (1993) 133–142.

[25] N. Secomandi, A rollout policy for the vehicle routing problem with stochastic demands, Operations Research 49 (2001) 796–802.

[26] N. Secomandi, Analysis of a rollout approach to sequencing problems with stochastic routing applications, Journal of Heuristics 9 (2003) 321–352.

[27] W.H. Yang, K. Mathur, R.H. Ballou, Stochastic vehicle routing problem with restocking, Transportation Science 34 (2000) 99–112.

[28] S. Gélinas, M. Desrochers, J. Desrosiers, M.M. Solomon, A new branching strategy for time constrained routing problems with application to backhauling, Annals of Operations Research 61 (1995) 91–109.