European Journal of Operational Research 244 (2015) 100–109

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

An approach to the asymmetric multi-depot capacitated arc routing problem

Dmitry Krushinsky[*], Tom Van Woensel

Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 30 December 2013
Accepted 7 January 2015
Available online 19 January 2015

Keywords:
Arc routing
Valid inequalities
Branch-and-cut

ABSTRACT

Despite the fact that the Capacitated Arc Routing Problems (CARPs) received substantial attention in the literature, most of the research concentrates on the symmetric and single-depot version of the problem. In this paper, we fill this gap by proposing an approach to solving a more general version of the problem and analysing its properties. We present an MILP formulation that accommodates asymmetric multi-depot case and consider valid inequalities that may be used to tighten its LP relaxation. A symmetry breaking scheme for a single-depot case is also proposed. An extensive numerical study is carried to investigate the properties of the problem and the proposed solution approach.

1. Introduction

The Capacitated Arc Routing Problem (CARP) originally defined by Golden and Wong (1981) has been extensively studied during the recent decades due to its numerous potential applications, including post delivery, waste collection, winter services (e.g. salt gritting or snow plowing), public transport routing, etc. To fulfil the needs of applications, a number of extensions and modifications of the CARP were considered. These can be roughly classified based on the following features:

- symmetric (undirected) vs. asymmetric (including a particular case of directed) network;
- single vs. multiple vehicles;
- single vs. multiple depots;
- single vs. multiple objectives (see, e.g., Grandinetti, Guerriero, Laganá, & Pisacane, 2003);
- additional constraints: time windows, priorities, etc.

For an overview of the modifications and solution methods we refer the reader to Corberan and Prins (2010); Hertz (2005). The available solution approaches can be divided into two broad categories: (a) those based on a reduction of the ARP to the Vehicle (Node) Routing Problem (VRP) (Longo, de Aragão, & Uchoa, 2006), and (b) those designed specifically for the ARP. The success of the first group of approaches is partially based on the fact that the VRP is substantially better studied than the ARP. At the same time, the transformation of the ARP to the VRP ignores sparsity of the underlying graph and creates computational problems for larger networks. On the contrary, most of the original ARP approaches utilise the sparse nature of realistic road networks (see Fig. 1), which allows large instances at least to be handled, if not solved to optimality (see, e.g., Bartolini, Cordeau, & Laporte, 2013; Bode & Irnich, 2012, 2014; Corberan, Oswald, Plana, Reinelt, & Sanchis, 2012 for a single vehicle case).

As seen from the literature, most research concentrates on the undirected version of the problem (see, e.g., Belenguer & Benavent, 2003; Bode & Irnich, 2014 and references within). At the same time, one way streets are not uncommon, which implies that a realistic approach should be able to handle this case. Furthermore, we could not find an exact approach dealing with the multi-depot CARP – another natural extension of the original problem. These observations motivated us to consider a more general version of the CARP with multiple depots and directed arcs.

The major goal of this paper is to study the properties of the CARP and to consider the possibility of finding exact solutions to the multi-depot CARP for realistic (large, sparse, asymmetric) road networks.

This paper is organised as follows. The next section presents the basic MILP formulation utilised throughout the paper. Section 3 focuses on the inequalities that can be used to tighten the formulation, followed by Section 4 describing our branch-and-cut solution approach. Section 5 provides a computational study of our approach. Finally, Section 6 concludes the paper with a summary of main results and future research directions.

2. The basic MILP model

In this section, we propose a two-index MILP formulation for the asymmetric multi-depot CARP. First, we introduce some notions and notation.

[*] Corresponding author. Tel.: +31402472069.
E-mail address: d.krushinsky@gmail.com (D. Krushinsky).

http://dx.doi.org/10.1016/j.ejor.2015.01.005
0377-2217/© 2015 Elsevier B.V. All rights reserved.

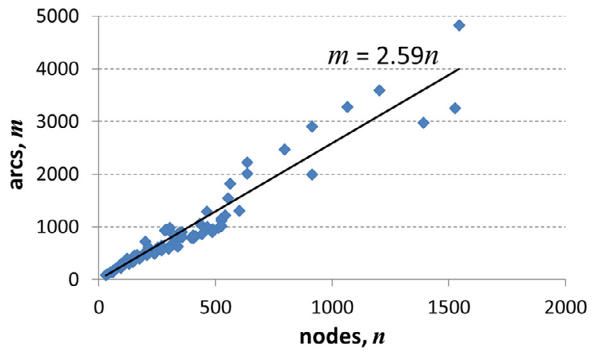

Fig. 1. Sparsity of real road networks for several urban areas in Europe and USA. Clearly, the number of arcs is of order n, i.e. much smaller than the potential limit of n^2 (n is the number of nodes).

2.1. Notions and notations

Through the rest of the paper, let $G(V, A, w)$ denote a directed weighted graph with the set of vertices V ($|V| = n$), the set of arcs A ($|A| = m$) and a weight function $w(\cdot) : A \to \mathbb{R}_+$. Given some set $S \subset V$, $G(V \setminus S)$ denotes a graph obtained from G by deleting each vertex from S together with all incident arcs.

A *walk* of length L in G is a sequence $i_1 - i_2 - \cdots - i_L$ of vertices such that there is an arc between each consecutive pair, i.e. $(i_l, i_{l+1}) \in A$ for all $l = 1, \ldots, L - 1$. Without any ambiguity one may think of a walk in terms of the corresponding arcs, rather then vertices. Under a *tour*, we understand a walk for which holds $i_1 = i_L$, and by the weight of a walk we understand the sum of the weights of all arcs in it. Note that the multiplicity of arcs in a walk does matter, i.e. if some arc is traversed three times then its weight contributes to the weight of the walk with a factor of 3. An *elementary cycle* is a tour traversing each vertex and arc at most once. Given a directed cycle $i_1 - i_2 - \cdots - i_j - \cdots - i_k - \cdots - i_L$, arc (i_j, i_k) is a *forward chord* if $i_1 - i_2 - \cdots - i_j - i_k - \cdots - i_L$ is also a directed cycle. Respectively, arc (i_k, i_j) is a *backward chord*.

We say that vertex j is *reachable* from vertex i, if there exists a directed $i - j$ path; arc (j, l) is reachable from i, if j is reachable from i.

Further, for any subset $S \in V$ let us denote by $\delta^+(S) = \{(i, j) \in A \mid i \in S, j \in V \setminus S\}$ and $\delta^-(S) = \{(i, j) \in A \mid i \in V \setminus S, j \in S\}$ the sets of arcs having one of the endpoints in S; for singletons we use shortcuts $\delta^+(i) = \delta^+(\{i\})$ and $\delta^-(i) = \delta^-(\{i\})$.

2.2. The formulation

The problem under consideration can be formalised as follows. Given a directed weighted graph $G(V, A, w)$, a demand function $d(\cdot) : A \to \mathbb{R}_+$, selected vertices (depot vertices) $v^d_k \in V$ ($k \in K$) and a number $Q \in \mathbb{R}_+$ (capacity), the goal is to find:

- $|K|$ tours of the minimum total weight, each tour traversing a corresponding vertex v^d_k ($k \in K$);
- an assignment of arcs with positive demand to the tours, such that each arc is assigned to one of the tours and the sum of demands assigned to a tour does not exceed Q.

Let us denote $d_a = d(a)$, $A^d = \{a \in A \mid d(a) > 0\}$ and introduce two sets of variables: $z^k_a \in \mathbb{Z}_+$ ($a \in A$) denote how many times arc $a \in A$ is traversed by tour k, and $x^k_a \in \{0, 1\}$ ($a \in A^d$) reflect the assignment of arcs to the tours. Now, the problem can be formulated as follows.

$$\sum_{k\in K}\sum_{a\in A} w_a z^k_a \longrightarrow \min \tag{1}$$

s.t.

$$\sum_{a\in A} d_a x^k_a \le Q, \quad k \in K \tag{2}$$

$$\sum_{k\in K} x^k_a = 1, \quad a \in A^d \tag{3}$$

$$\sum_{a\in\delta^+(i)} z^k_a = \sum_{a\in\delta^-(i)} z^k_a, \quad i \in V, k \in K \tag{4}$$

$$x^k_a \le z^k_a, \quad a \in A^d, k \in K \tag{5}$$

$$\sum_{a\in\delta^-(i)} u^k_a \le 1, \quad i \in V, k \in K \tag{6}$$

$$u^k_a \le z^k_a, \quad a \in A, k \in K \tag{7}$$

$$\sum_{a\in\delta^-(v^d_k)} u^k_a = 0, \quad k \in K \tag{8}$$

$$y^k_{v^d_k} = 0, \quad k \in K \tag{9}$$

$$y^k_j \ge y^k_i + 1 + M(u^k_{(i,j)} - 1), \quad (i, j) \in A, k \in K \tag{10}$$

$$\sum_{a\in\delta^-(i)} u^k_a \ge \frac{1}{M}\sum_{a\in\delta^+(i)} z^k_a, \quad i \in V \setminus \{v^d_k\}, k \in K \tag{11}$$

$$z^k_a \in \mathbb{Z}_+, \quad a \in A, k \in K \tag{12}$$

$$x^k_a \in \{0, 1\}, \quad a \in A^d, k \in K \tag{13}$$

$$y^k_i \in \mathbb{R}_+, \quad i \in V, k \in K \tag{14}$$

$$u^k_a \in \{0, 1\}, \quad a \in A, k \in K \tag{15}$$

Objective (1) explicitly minimises the total weight of all the tours. The capacity of each tour is limited by constraints (2), while constraints (3) ensure that each demand arc is served. Constraints (4) are flow conservation constraints. Further, constraints (5) ensure that only traversed demand arcs can be assigned to a tour. Finally, constraints (6)–(11) ensure connectedness of each tour by constructing a tree rooted at the depot vertex v^d_k and spanning all vertices traversed by a tour. The tree is oriented so that there is a directed path from the root to each leaf. Variables u^k_a reflect the arcs in such a tree, while variables y^k_i denote the level of vertex i in a tree. The root has level 0 (as assigned by constraints (9)), its neighbours have level 1, etc. Constraints (6) and (7) ensure that each vertex in the tree has at most one incoming arc and that the tree contains only the traversed arcs, respectively. Constraints (10) prohibit cycles. Constraints (11) ensure that each traversed vertex, except the depot, has an incoming arc in the tree. Parameter M in (10) and (11) is a large enough positive number. Note that constraints (6)–(11) are given here purely on the purpose of making the formulation complete and are not used further in this paper.

It is not hard to understand that the proposed formulation can be adjusted for the case of an undirected graph by assuming that each undirected edge is represented by two arcs and replacing constraints (3) by:

$$\sum_{k\in K} x^k_{(i,j)} + x^k_{(j,i)} = 1, \quad (i, j) \in A^d. \tag{16}$$

Though this formulation has a polynomial size, it is quite extensive, both in terms of constraints ($|A^d| + (4 + 3n + 2|A^d| + 5m)|K|$) and variables ($m|K| + |A^d| \cdot |K| + n|K| + m|K|$), $2m|K| + |A^d| \cdot |K|$ of which are integer. In case all arcs have positive demands, $A^d = A$ holds and these quantities become $n + (4 + 3n + 7m)|K|$, $(n + 3m)|K|$ and $3m|K|$, respectively. However, it can be seen that as much as $(3 + 3n + 5m)|K|$ constraints and $(n + m)|K|$ variables ($m|K|$ of which are integer) are used purely to guarantee connectedness of the tours. In fact, constraints (6)–(11) can be replaced by the following subtour eliminating constraints:

$$\sum_{a\in\delta^+(S)} z^k_a \ge \frac{1}{M}\sum_{a\in A'} z^k_a, \quad k \in K, \tag{17}$$

where $v_d^k \in S \subset V, A' = \{(i,j) \mid (i,j) \in A,\ i,j \in V \setminus S\}$, M is a large enough positive constant ensuring that the right hand side does not exceed 1. Similar constraints are considered in Maniezzo (2004), but we will show how to strengthen them.

Though formulation ((1–5), (12)–(13), (17)) can have exponentially many constraints, it can be utilised within a row generating framework to realise computational benefits. The further analysis is based on its LP relaxation. Note that this formulation does not explicitly contain information about locations of the depots, this information is implicitly present only via the definition of sets S in constraints (17).

3. Tightening the formulation

In this section, we consider several families of valid inequalities, divided into two major groups: (a) *structural* inequalities enforcing a correct structure of feasible solutions, and (b) *capacity-related* inequalities derived from the capacity constraints. Note that some of these valid inequalities were also used in other CARP formulations, but we describe them for the sake of completeness (there are certain peculiarities w.r.t. our formulation). In addition, we consider symmetry breaking constraints that are not valid inequalities, as they cut off some feasible solutions (but preserve at least one optimal solution).

3.1. Structural valid inequalities

We first show how the subtour elimination constraints can be tightened and propose other valid inequalities based purely on the structure of the tours.

3.1.1. Subtour elimination

Observe that any optimal solution to our relaxation does not contain a subtour not serving any demand arc, because eliminating such a subtour improves the objective value (assuming that all arc weights are positive) and does not violate any constraints. Taking this into account allows revising inequality (17) as:

$$\sum_{a \in \delta^-(S)} z_a^k \geq \frac{1}{|A'|} \sum_{a \in A'} x_a^k, \tag{18}$$

where $A' = \{(i,j) \mid (i,j) \in A^d,\ i,j \in V \setminus S\}$. Though one constraint (18) per cut (equivalently, per set S) suffices, it is beneficial to split it as follows:

$$\sum_{a \in \delta^-(S)} z_a^k \geq x_a^k, \quad \forall a \in A', \tag{19}$$

because the latter leads to a tighter LP relaxation.

Constraints (19) can be trivially tightened in case $G(V \setminus S)$ is disconnected: each left hand side sum can be limited to only arcs joining S with the connected component of $G(V \setminus S)$ containing particular $a \in A^d$ (see Fig. 2a).

However, for a directed graph this case can be generalised and constraints (19) can be tightened even if $G(V \setminus S)$ is connected but not strongly connected (see Fig. 2b). In particular, it is enough that for each demand arc $a' = (i', j') \in A'$ the left hand side sum is taken only over all arcs $a = (i,j) \in \delta^+(S)$ for which i' is reachable from j in $G(V \setminus S)$.

Besides the case of disconnected subtours, inequalities (19) can be used to cut off some fractional solutions corresponding to connected tours. However, in this case the separation problem becomes more difficult: instead of just identifying connected components, it involves finding a cut with a weight smaller than the weight of some arc separated from the depot by the cut (weights are values of z-variables in the solution). We formulate this separation problem as an MILP (see Appendix B.1).

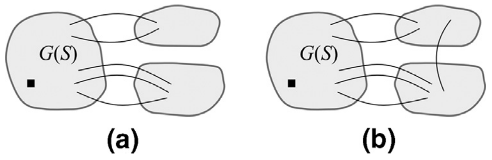

Fig. 2. Two cases when constraints (19) can be tightened: (a) a disconnected case; (b) a weakly connected case. The black square denotes the depot vertex.

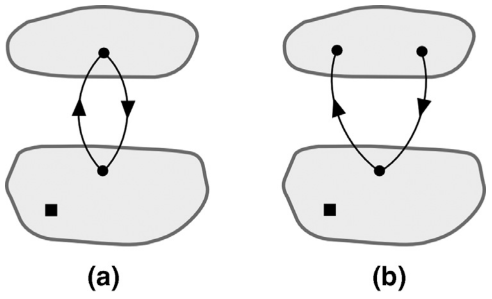

Fig. 3. Two types of bridges: (a) O-bridge; (b) V-bridge. Grey ovals denote connected components, the black square denotes the depot vertex.

3.1.2. Bridge inequalities

Besides the subtour elimination (necessary for a valid ARP solution), there are a number of constraints that tighten the LP relaxation and may be useful in practice. Here we would like to consider a family of inequalities that rely on the presence of bridges in the input graph.

Let us define a *bridge* as a pair of opposite arcs whose removal increases the number of connected components in G by 1. This notion of a bridge is quite general, but we consider only two types of bridges that we call O-bridges (two arcs share two vertices) and V-bridges (two arcs have one common vertex), see Fig. 3.

The following simple observation shows that O-bridges have a very general property.

Observation 1. *In each optimal solution to ARP, each arc in any O-bridge is traversed by each tour at most once.*

The properties of V-bridges are less general. Let us consider some V-bridge $B = \{(i, j_1), (j_2, i)\}$ that decomposes G into $G_1(V_1, A_1)$ and $G_2(V_2, A_2)$ so that $i \in V_1$ and $j_1, j_2 \in V_2$. If one defines G' as $G'(V_2 \cup \{i\}, A_2 \cup B)$, then the following lemma takes place.

Lemma 1. *If G' defined above is a directed cycle without forward chords or G' contains an arc (j_2, j_1) such that $w_{(j_2,j_1)} < w_{(i,j_1)} + w_{(j_2,i)}$, then each arc from B is traversed by each tour at most once in any optimal solution.*

Proof. Let us label vertices of G' as $i_1, i_2, \dots, i_n$ so that G' contains all arcs of the type (i_j, i_{j+1}) $(j = 1, \dots, n-1)$ and arc (i_n, i_1). Suppose, i_1 corresponds to i in the statement of the lemma, then i_2 and i_n correspond to j_1 and j_2, respectively. Note that G' does not contain any chord having i_1 as one of its endpoints, due to the fact that $\{(i_1, i_2), (i_n, i_1)\}$ corresponds to a bridge in G.

Clearly, if G' is a chordless cycle, each arc needs only to be traversed at most once. If G' contains a backward arc, say (i_8, i_5), than the route $i_1 - \dots - i_8 - i_5 - \dots - i_8 - \dots - i_n$ is the shortest route traversing each arc in G' and it does not traverse (i_1, i_2) and (i_n, i_1) more than once. It is straighforward to check that the latter holds also in case of an arbitrary number of backward arcs. Presence of a single forward arc, say (i_5, i_8), leads to an optimal route $i_1 - i_2 - \dots - i_5 - i_8 - \dots - i_n - i_1 - i_2 - \dots - i_5 - \dots - i_8 - \dots - i_n$ that traverses (i_1, i_2) and (i_n, i_1) twice. However, if G' contains a shortcut (i_n, i_2) satisfying the condition of the lemma, any route traversing (i_1, i_2) and (i_n, i_1) at most once will be optimal, e.g. $i_1 - i_2 - \dots - i_5 - i_8 - \dots - i_n - i_2 - \dots - i_5 - \dots - i_8 - \dots - i_n$. Again, it can be easily verified that the result extends to the case of arbitrarily many forward arcs in G'. □

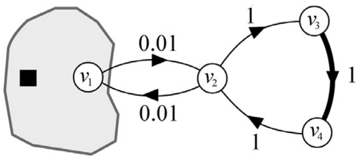

Fig. 4. A fractional solution that can be eliminated by bridge inequalities. Numbers at arcs are the values of the corresponding z-variables, a sole demand arc is shown in bold, the black square denotes the depot vertex.

Corollary 1. *If G' defined above contains an arc (j_2, j_1) such that $w_{(j_2,j_1)} < w_{(i,j_1)} + w_{(j_2,i)}$, then each arc from B is traversed by each tour at most once in any optimal solution.*

Proof. Note that the flows through (i, j_1) and (j_2, i) are equal for any feasible solution. Suppose, there is an optimal solution violating the assertion of the corollary. By replacing a unit of flow through $j_2 - i - j_1$ by a unit of flow through (j_2, j_1) one improves the objective value while preserving feasibility. □

These properties of bridges allow introducing the following constraints that exclude fractional solutions similar to the one in Fig. 4.

Lemma 2. *Suppose, $\{(i, j_1), (j_2, i)\}$ is an O-bridge (in this case $j_1 = j_2$) or a V-bridge satisfying Lemma 1. Then the following constraint is valid:*

$$\sum_{a\in\delta^+(i)\setminus(i,j_1)} z_a^k \geq z_{(i,j_1)}^k. \tag{20}$$

Proof. Let us consider the following two cases: $z_{(i,j_1)}^k = 0$ and $z_{(i,j_1)}^k \geq 1$. In the first case, the inequality trivially holds. The second case can be reduced to $z_{(i,j_1)}^k = 1$ due to Observation 1 or Lemma 1, depending on the type of the bridge under consideration. Suppose, that the inequality (20) is not valid. This implies existence of a feasible route traversing (i, j_1) but not traversing any arc $a \in \delta^+(i) \setminus (i, j_1)$ and, thus, not containing any path from i to the depot. This, in turn implies that the route contains a directed path from j_1 to the depot that does not include arc (j_2, i), which is a contradiction because $\{(i, j_1), (j_2, i)\}$ is a bridge. □

Fig. 4 illustrates bridge inequalities. It can be seen in the figure that $\{(v_2, v_3), (v_4, v_2)\}$ is a V-bridge, and inequality (20) becomes $z_{(v_2,v_1)}^k \geq z_{(v_2,v_3)}^k$, which is violated by the given solution.

Clearly, the number of bridges is at most $m/2$ as each arc can belong to only one bridge. This bound is tight for O-bridges (consider a graph obtained from an undirected tree by replacing each edge by two opposite arcs). Taking into account that the number of possible bridge inequalities is relatively small, it makes sense to include all of them into the formulation.

3.2. Capacity-related valid inequalities

Taking into account that our formulation contains the so-called knapsack constraints (2) one may apply to each tour any of the valid inequalities for the well studied 0–1 knapsack problem, e.g. cover inequalities for which separation and lifting procedures are known (see, e.g., Balas, 1975). However, such inequalities may not work well in practice, because the projection of the ARP feasible set onto the subspace of x-variables for each tour has a nonempty intersection with the interior of convex hull of 0–1 knapsack feasible solutions. This is due to the fact that normally $\sum_{a\in A} d_a < Q|K|$ holds as a strict inequality. Thus, pure exploitation of the knapsack constraints in the space of x-variables seems not very promising.

Observing the fact that constraints (2) and (3) describe a polytope of the Generalised Assignment problem (GAP), one may try to separate some of the GAP facet defining inequalities. For an overview of the GAP and corresponding inequalities, see, for example, Gottlieb and Rao (1990). However, according to our experience, these are not efficient: in a branch-and-cut tree with hundreds of nodes we could find only a couple of violated GAP inequalities using both specific GAP cutting planes and general techniques, like lift-and-project procedure (see, e.g., Balas, Ceria, & Cornuéjols, 1993). To sum up, any cutting planes defined only in the space of x-variables are unlikely to be violated by solutions to our LP-relaxation.

This motivated us to consider possibilities of utilising knapsack constraints in the space of both x- and z-variables exploiting relations between the two classes of variables. As we show in the next sections, this can be done in different ways.

3.2.1. Aggregated capacity inequalities (CI1)

Consider some subset $S \subset V \setminus \{v_k^d\}_{k\in K}$ not containing *any* of the depot vertices. Clearly, if the total demand of all arcs having at least one endpoint in S exceeds Q then at least two tours cross $\delta^+(S)$ (or, equivalently, $\delta^-(S)$). This idea may be generalised into the following inequality:

$$\sum_{k\in K}\sum_{a\in\delta^+(S)} z_a^k \geq \left\lceil \frac{1}{Q} \sum_{a\in A'\cup\delta(S)} d_a \right\rceil, \tag{21}$$

where $A' = \{(i, j) \mid (i, j) \in A^d, \ |\{i, j\} \cap S| \geq 1\}$. Equivalent inequalities were proposed by Belenguer and Benavent (2003) for their aggregated formulation.

The separation problem for inequalities (21) is equivalent to the problem of finding a cut (in a graph of the same size as the input graph) of a weight exceeding a certain threshold that depends on the cut itself. For the numerical experiments, we formulated this separation problem as an MILP (see Appendix B.2). As the structure of the latter closely resembles the structure of an MILP formulation for a polynomially solvable mincut problem, it can be solved reasonably fast.

Note that inequalities (21) are aggregated in a sense that they impose certain constraints on all the tours. At the same time, they induce inequalities involving only a subset of all the tours. For some fixed $S \subset V \setminus \{v_k^d\}_{k\in K}$, if one denotes by q the value of the right hand side in (21), then the following inequalities are valid:

$$\sum_{k\in K'}\sum_{a\in\delta^+(S)} z_a^k \geq 1, \quad \forall K' \in K : |K'| = |K| - q + 1. \tag{22}$$

3.2.2. Capacity inequalities (CI2)

These inequalities make sense only if the demand cannot be served by less than $|K|$ vehicles. Let us denote by Q' the minimum demand assigned to a route. This can be approximated as $Q' \geq D - Q \cdot (|K| - 1)$, where D is the total demand, or computed via a variant of the bin packing problem. It is not hard to understand that the inequality

$$\sum_{a\in\delta^+(S)} z_a^k \geq 1, \quad k \in K \tag{23}$$

is valid for any S satisfying $\sum_{a\in A^d\cap S\times S} d_a < Q'$. For a particular tour k', it can be lifted by introducing the information from other tours as follows:

$$\sum_{a\in\delta^+(S)} z_a^{k'} + \sum_{k\in K\setminus\{k'\}}\sum_{a\in A'} x_a^k \geq 1, \tag{24}$$

where sets $S \subset V$ and $A' \subset A^d \cap S \times S$ must satisfy $\sum_{a\in(A^d\cap S\times S)\setminus A'} d_a < Q'$. In particular, if $A' = \emptyset$, inequality (24) transforms to (23), and S must be small enough to satisfy $\sum_{a\in A^d\cap S\times S} d_a < Q'$). Otherwise, S can be an arbitrary large subset of V. Note that each such valid inequality is described by a triple (k', S, A').

The separation problem for (24) can be solved in two ways: either by formulating it as an MILP (see Appendix B.3) or by the following heuristic algorithm (we use the notation $\bar{x}$ to denote the value of variable x):

```
Input: graph $G(V, A)$, solution to cut off $(\bar{z}^k_a, \bar{x}^k_a)$, $k' \in K$
Initialise: $S^* := V$, $A'^* := A^d$ – parameters of the cut (24); define $viol(k', S, A')$ as an amount by which inequality (24) is violated for the given parameters
Repeat: $L$ times
1. generate a random cut $\delta(S)$ in $G$ by running the contraction algorithm (see, e.g., Karger & Stein, 1996) while treating values $\bar{z}^{k'}_a$ as weights of arcs in $G$.
2. if $\sum_{a\in\delta^+(S)} \bar{z}^{k'}_a \geq 1 - viol(k', S^*, A'^*)$, then go to step 1, else:
3. solve the following knapsack problem
   $u^* = argmax \left\{ \sum_{a\in A^d\cap S\times S} \bar{x}^{k'}_a u_a \ : \ \sum_{a\in A^d\cap S\times S} \left( \sum_{k\in K\setminus\{k'\}} \bar{x}^{k'}_a \right) u_a \leq Q' \right\}$   (25)
4. define $A' := \{a \ : \ u^*_a = 0\}$ and calculate the violation $viol(k', S, A')$ of (24) for the current $k', S, A'$.
5. if $viol(k', S, A') > viol(k', S^*, A'^*)$, then update $S^* := S, A'^* := A'$.
Output: $(k', S^*, A'^*)$
```

The contraction algorithm (step 1) tends to produce a cut in G minimising the leftmost sum in (24), while by solving the knapsack problem the second sum is minimised. Step 2 is a shortcut that reduces the amount of computations for the cutting planes that cannot be stronger than the previously found one. Parameter L must be chosen in order to balance the quality of the generated inequalities and the computational effort. The value $L = |V|^2\log(|V|)$ guarantees that the cut minimising the first sum in (24) is found with a high probability (see, e.g., Karger & Stein, 1996), but it may not correspond to the most violated constraint. We use a fixed value $L = 100$ for the sake of efficiency.

Though the knapsack problem (step 2) is NP-hard, there exist a number of efficient specialised algorithms. We use the one by Pisinger (1996).

To the best of our knowledge, this family of cutting planes is not described in the literature.

In Appendix A we present one more family of valid inequalities that we call CI3. These inequalities, however, did not prove useful.

3.3. Symmetry breaking and surrogate demands

Clearly, in case all the depots are placed in the same vertex (a single-depot case), the described above model has an inherent symmetry w.r.t. a permutation of routes. In contrast to the node routing, there is also another source of symmetry inherent to both single- and multi-depot case. Such a symmetry relies on the fact that feasible routes may intersect, i.e. some arcs may (usually, will) be traversed by more than one vehicle. This creates certain freedom in assigning demands to routes: serving the demand of some arc by any of the vehicles traversing it does not affect the objective value (of course, if capacity constraints permit).

These symmetries can be also seen in our formulation. It is easy to see that all the described constraints are either valid for each route or contain variables from all routes with equal coefficients. Even if depots are placed in different vertices, only constraints (19) may break the symmetry if derived from a cut separating at least one pair of depots.

Let us start with a single-depot case. One can reduce the symmetry by considering only a subspace of solutions where the routes are sorted by the demand they serve, for example, in a decreasing order. That is, the following constraints can be included into the formulation while preserving at least one optimal solution:

$$\sum_{a\in A^d} d_a(x^k_a - x^{k+1}_a) \geq 0, \quad k = 1, \ldots, |K| - 1. \tag{26}$$

In general, these constraints are not very efficient because there are feasible solutions with some routes serving equal demands. Now we are going to show how these constraints can be modified to become more useful. The modification is based on the following general observation.

Observation 2. *Constraints*

$$\sum_{a\in A^d} \alpha_a(x^k_a - x^{k+1}_a) \geq 0, \quad k = 1, \ldots, |K| - 1 \tag{27}$$

preserve at least one optimal solution for any fixed set of real coefficients $\{\alpha_a\}_{a\in A^d}$.

The proof is trivial: take any optimal solution and permute the routes so that the corresponding sums become sorted.

Observation 2 means that instead of actual demands one can choose 'surrogate' coefficients for (26) so that the left hand side tends to be strictly positive for most of the feasible solutions. A straightforward (yet, not very practical) choice is given in the following proposition.

Proposition 1. *Constraints*

$$\sum_{i=1}^{|A^d|} 2^i(x^k_i - x^{k+1}_i) \geq 1, \quad k = 1, \ldots, K - 1 \tag{28}$$

preserve at least one optimal solution.

This proposition is easy to verify if one treats $(x^k_1, \ldots, x^k_{|A^d|})$ as a binary representation of some integer. As each route in a feasible solution has a unique set of x-values, the corresponding integers are also all different, allowing for 1 in the right hand side.

By performing a more rigorous analysis, it is possible to select a sequence of coefficients which is dominated by $\{2^i\}$ but still guarantees 1 in the right hand side of inequality (28). Still, we conjecture that each such sequence has an exponential (average) rate of growth, and is practically inapplicable due to computational issues. On the other hand, we will show in Section 5 that constraints (27) equipped with coefficients of the form $(1 + \epsilon)^i$ may be useful, where ϵ is a constant small enough to ensure that all the coefficients are of a reasonable magnitude.

In a multi-depot setting, Observation 2 does not hold in general. Constraints (27) can still be used for obtaining high quality solutions, but optimality is not guaranteed in this case.

4. The branch-and-cut algorithm

Given the LP-relaxation of formulation ((1–5), (12–13), (19)) and the families of constraints described in Section 3 we develop a branch-and-cut algorithm. Note that the cutting plane approach alone does not work because all the constraints mentioned above do not suffice for a complete description of the convex hull of feasible solutions.

As input of the algorithm, we assume the formulation ((1–5), (12–13)) extended with bridge inequalities (20). The subtour elimination inequalities (19) are added during the cutting phase as needed.

4.1. Cutting

Each node of the BnB tree (subproblem) is subjected to a cutting phase when violated inequalities (19), (21) and (24) are added sequentially until none can be found. If the rounded up lower bound of the node exceeds the best global upper bound or an integer solution is found, the node is fathomed. Otherwise, a branching step takes place.

4.2. Branching

A variety of branching rules were implemented and tested. The first is the classical branching on the most fractional variable: if f_x denotes the fractional part of some variable x, then the variable with the highest value of $(0.5 - |0.5 - f_x|)$ is selected. We also consider a

Table 1
Performance of our approach for artificial instances of different sizes.

Name	$\|V\|$	$\|A\|$	$\|A^d\|$	$\|K\|$	LB	UB	Gap, percent	Time, seconds	No. of cuts	No. of nodes
t02	20	32	32	3	1286	1286	0	0	29	1
t03	30	53	53	3	1763	1763	0	1	180	1
t04	40	70	70	3	2737	2737	0	2	237	1
t05	50	87	87	3	3765	3765	0	2	340	2
				4	4021	4021	0	16	474	3
t06	60	104	104	3	4619	4619	0	29	483	9
t07	70	116	116	3	5478	5478	0	14	374	5
t08	80	132	132	3	6379	6379	0	9	610	2
t09	90	148	148	3	7359	7359	0	71	720	8
t10	100	164	164	3	8317	8317	0	23	671	5
R20x12	61	123	123	3	400	400	0	7	103	3
				4	416	416	0	8	241	1
				5	432	432	0	129	295	16
				6	448	448	0	104	273	12
				7	464	464	0	420	651	34
				8	480	480	0	961	846	60
R30x18	105	187	187	3	724	724	0	2	24	1
R40x23	243	521	521	3	1420	1420	0	339	248	1

weighted extension of this rule by multiplying the just mentioned criterion by the weight of the corresponding arc.

Observing the relations between x-variables, we have also developed another rule for selecting a variable to branch upon. Let us fix some arc a and consider the corresponding variables $x_a^1, \dots, x_a^{|K|}$. While in any integer feasible solution exactly one of them must be nonzero, a feasible solution to the LP- relaxation may assign several positive values to these variables. Intuitively, this can be interpreted in the following way: instead of selecting a single route to serve arc a the LP-relaxation 'prefers' some routes (those with higher values) to other ones. In the worst case, there is no preference and all the variables obtain the value of $1/|K|$. Another interpretation is as follows. Suppose, one is going to perform a randomised rounding of the fractional solution. Clearly, rounding each x-variable independently is not efficient because this may generate a lot of infeasible solutions. It rather makes sense to randomly select one variable for each demand arc that will be set to 1. A natural distribution for this randomised selection is described by the values of the variables. Now, the case when all these variables are equal to $1/|K|$ is the worst one because it represents the highest uncertainty, or, equivalently, the highest entropy. Thus, we propose first to select arc a with the highest entropy defined as $-\sum_k x_a^k \ln(x_a^k)$ and then for this fixed a select the most fractional variable x_a^k. Such a procedure tries both to reduce uncertainty and to perform the deepest cut. A weighted extension of this branching rule is also possible where entropies are multiplied by the weights of the corresponding arcs. This branching rule can be also applied to z-variables by considering the fractional parts of their values.

The strength of the above two rules is that they essentially fix some variables. Their weakness is due to the local effect: at each branching only 1 variable is affected if branching was performed on a z-variable, and K variables are affected otherwise. This effect may result in very slowly growing lower bounds and deep branch-and-bound trees. This motivated us to consider alternative branching rules that would facilitate a faster update of the lower bound.

One of such possibilities is branching on the flows through vertices: the sums of in- and outcoming flow variables must be integral for each vertex. For vertices with an in- or out-degree of 1 this rule is equivalent to branching on a single z-variable, but for vertices with higher degrees it has a more global impact. A similar branching rule was used e.g. in Bode and Irnich (2012).

An even stronger possibility is the branching on cuts in G: a one-way flow through each cut in the input graph must be integral. Except of affecting many variables (those corresponding to arcs in a cut) this rule has another advantage. If the flow through the cut is set to 0, none of the arcs separated by the cut from the depot vertex can be served by the current tour and some x-variables can be fixed to 0. Actually, this latter case is the most interesting; that is why we consider only cuts where a one-way flow does not exceed 1. Finding a cut to branch upon is essentially equivalent to finding a minimum cut and can be efficiently done.

5. Computational experiments

The purpose of our numerical experiments is twofold. First of all, we want to check the possibility of solving the CARP defined on large networks and the impact of the proposed cutting planes, symmetry breaking constraints and branching rules. Next, we want to study the influence of a multi-depot setting on the complexity of the problem. Thus, the experiments can be divided in two groups: those aimed at studying the properties of the approach and of the multi-depot CARP itself, respectively.

For testing purposes, we propose a library of benchmark instances that contains:

1. Self-generated artificial instances (see Table 1):
   - nested structure (a wide range of weights and demands: $max.weight/min.weight \approx 50$);
   - grid structure (a small range of weights and demands: $max.weight/min.weight \approx 5$).
2. Instances from the CARP literature (see Table 2).
3. Instances derived from real road networks (see Table 3).

The library and its detailed description are available at http://smartlogisticslab.ieis.tue.nl/#benchm.

For the experimenting purposes, the formulation was solved with CPLEX 12.6 (single thread mode), the proposed cutting planes and branching rules were implemented as callbacks and coded in C++. The subtour eliminating constraints were implemented as lazy cuts. Based on the testing experience, a limit of 75 new cuts per node was set for constraints CI2 (24) and to 15 – for constraints (19) if applied to fractional solutions. We have also observed that constraints CI1 (21) are rarely found deep in the branch-and-bound tree, that is why we tried separating them only at the first three levels of the tree. Unless stated otherwise, the weighted entropy-based branching rule was used. The computing times are reported for a laptop with Intel Core-i5 2.60 GHz CPU and 4GB RAM.

In the first series of experiments, we assigned all the depots to the same vertex (the first one). To have a unified setting, in all the experiments the capacity of a vehicle was defined as $\lceil D/(|K| - 0.5)\rceil$, where D is the total demand. For each input graph, we solved the CARP for $3, 4, \dots$ vehicles. Tables 1–3 summarise the sizes of the dataset and our computational experience with the instances that we could

Table 2
Performance of our approach for some instances from the literature.

Name	$\|V\|$	$\|A\|$	$\|A^d\|$	$\|K\|$	LB	UB	Gap, percent	Time, seconds	No. of cuts	No. of nodes
val1A	24	78	78	3	292	292	0	0	54	1
val2A	24	68	68	3	388	388	0	1	241	1
val3A	24	70	70	3	134	134	0	3	250	6
val4A	41	138	138	3	688	688	0	3	308	1
val5A	34	130	130	3	734	734	0	3	120	1
val6A	31	100	100	3	380	380	0	2	69	1
				4	380	380	0	6	793	2
				5	384	384	0	62	739	18
				6	390	390	0	101	993	24
val7A	40	132	132	3	498	498	0	1	82	1
val8A	30	126	126	3	694	694	0	1	79	1
val9A	50	184	184	3	556	556	0	3	95	1
val10A	50	194	194	3	752	752	0	3	72	1
val10B	50	194	194	3	752	752	0	2	72	1
val10C	50	194	194	3	752	752	0	2	72	1
val10D	50	194	194	3	752	752	0	2	72	1
egl-e1-A	77	196	102	3	3472	3472	0	43	581	8
				4	3742	3778	0.96	24 hours[a]	45,480	5246
				5	4048	4090	1.03	24 hours[a]	34,128	4385
egl-e2-A	77	196	144	3	4194	4194	0	183	1181	21
egl-e3-A	77	196	174	3	4798	4798	0	112	998	10
egl-e4-A	77	196	196	3	5252	5252	0	105	1168	9
egl-s1-A	140	380	150	3	3637	–	–	24 hours[a]	8863	1466
egl-s2-A	140	380	294	3	6953	6953	0	5130	3255	151
egl-s3-A	140	380	318	3	7195	7201	0.09	24 hours[a]	11,044	656
egl-s4-A	140	380	380	3	8464	8464	0	2410	1630	39

[a] Solution procedure was interrupted, the best feasible solution reported.

Table 3
Performance of our approach for real road networks of different sizes.

Name	$\|V\|$	$\|A\|$	$\|A^d\|$	$\|K\|$	LB	UB	Gap, percent	Time, seconds	No. of cuts	No. of nodes
chicA3	195	509	299	3	45519	45,519	0	2460	2095	25
chicA4	95	300	205	3	65,125	65,125	0	276	602	12
				4	66,281	66,281	0	337	995	17
				5	67,982	67,982	0	2253	1633	52
cord3	163	464	439	3	494,902	494,902	0	1111	1111	15
dach1	148	396	380	3	37,170	37,170	0	3816	3407	57
kyivA4	202	479	424	3	92,769	92,769	0	39,021	5486	250
kyivC4	174	397	363	3	93,323	93,323	0	14,317	6866	174
kyivD2	215	525	473	3	48,390	48,390	0	6083	2590	46
kyivD3	149	389	352	3	51,885	51,885	0	303	1771	9
kyivD4	101	286	253	3	56,090	56,090	0	198	944	5
mons3	238	489	457	3	88,186	88,186	0	1755	166	13
mons4	129	298	274	3	94,451	94,451	0	72	290	2
				6	102,887	102,887	0	1062	681	17
				9	111,323	111,323	0	12,794	1815	116
nyA1	298	588	180	3	40,408	40,581	0.43	24 hours[a]	2575	241
nyA2	269	556	179	3	43,274	43,274	0	26,332	1534	66
nyA3	102	321	150	3	58,857	58,857	0	90	611	3
nyC4	152	462	303	3	132,620	132,620	0	1066	402	10
romeA1	337	629	435	3	34,375	34,375	0	3174	2330	16
tour4	121	394	377	3	101,408	101,408	0	98	348	1
tueA1	259	561	501	3	29,238	29,238	0	29,453	8526	124
tueA2	206	465	412	3	29,391	29,391	0	2266	2652	25
tueA3	146	330	291	3	29,650	29,650	0	849	1715	19
tueA4	93	225	207	3	30,750	30,750	0	43	804	6
tueA5	59	146	135	3	32,318	32,318	0	14	663	4
vypas1	114	320	288	3	76,266	76,266	0	969	3094	32
vypas2	95	262	232	3	54,846	54,846	0	1031	2214	37
vypas3	75	214	184	3	42,366	42,366	0	331	1627	26
vypas4	61	174	152	3	32,590	32,590	0	203	1206	25
vypas5	51	144	124	3	25,128	25,128	0	86	874	23
vypas6	37	106	92	3	17,532	17,532	0	5	284	1
				5	18,722	18,722	0	57	634	11
				7	20,222	20,630	0	17,940	8892	3379
vypas7	30	92	84	3	17,572	17,572	0	2	408	1

[a] Solution procedure was interrupted, the best feasible solution reported.

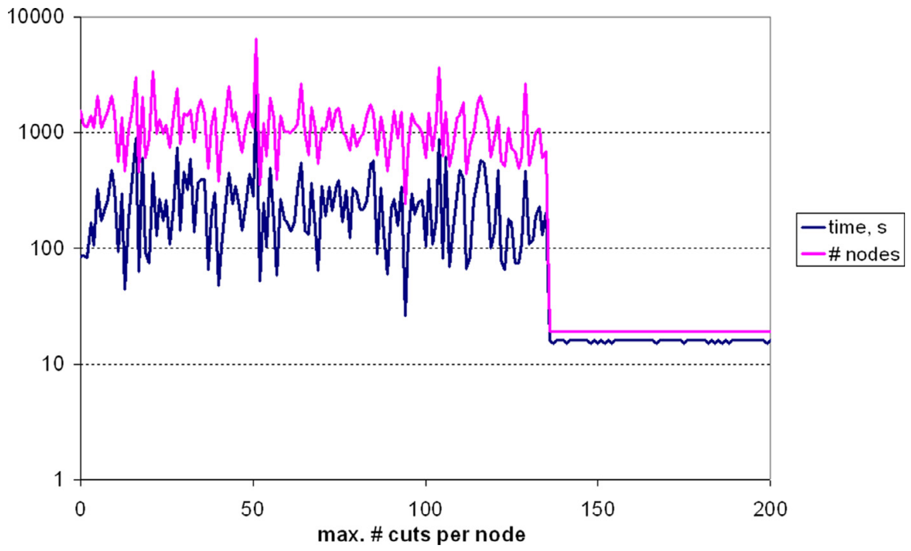

Fig. 5. Performance of constraints CI3 (24); instance 'egl-e1-A', 3 vehicles.

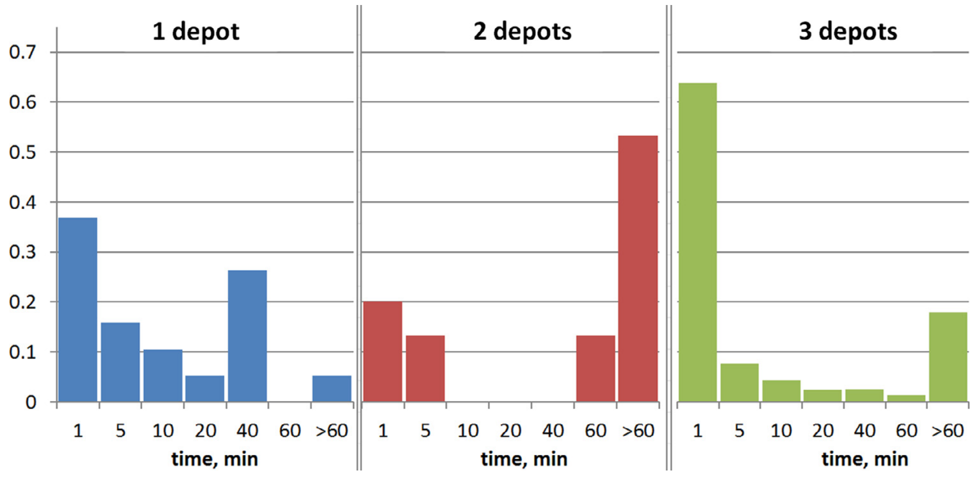

Fig. 6. Normalised histogram of CPU times for an instance with three vehicles and different mutual locations of depots.

solve. The first column in the tables contains the titles of the instances (networks), 'LB' and 'UB' denote the root lower bound and the global upper bound (value of the best feasible solution found), respectively. The rest of the column titles are self-descriptive.

As can be seen from Tables 1–3, the lower bound is usually tight. It can be also seen that the performance of the proposed approach does only slowly degrade as the size of the underlying network increases, but strongly deteriorates with an increase in the number of vehicles. This can be explained by the inherent symmetries of the model and by the fact that each vehicle (route) adds many variables to the formulation. It is remarkable that, in contrast to many other problems, even in case of exact lower bounds many branchings are needed to find an integral solution. This behaviour was observed for all variations of our framework, as well as for the CPLEX solver with the default settings. Note, that for instances from the literature a direct performance comparison with other papers cannot be made because we replace each undirected edge with two arcs, thus doubling the problem size.

The behaviour of the cutting planes can be characterised as follows. Constraints CI1 (21) are crucial for a tight root lower bound: without them the CPU time needed for obtaining the same bound increases by orders of magnitude. Constraints (19) are necessary to eliminate subtours, but it also makes sense to apply them to fractional solutions without disconnected subtours for the sake of performance. The proposed constraints CI3 (24) are very useful for reducing the size of the BnB tree and the CPU times (see Fig. 5). We observed that the heuristic for their separation described in Section 3.2.2 produces high quality cutting planes, in comparison with the MILP that solves the separation problem exactly, and is considerably faster. The contribution of bridge inequalities (20) is very limited for the instances tested, but they may be useful for instances with more bridges. Concerning the generic cutting planes implemented in CPLEX, typically, very few of them (less than 5) are found during the solution, mainly zero-half cuts.

Among the branching rules mentioned in Section 4, the weighted entropy-based rule leads to the smallest BnB tree and CPU time for most of the instances tested. Generally, branching on variables performed better that branching on cuts or node flows, weighted versions of rules performed better than simple ones (a difference of 10–20 percent).

We would like to mention that the symmetry breaking scheme described in Section 3.3 allows reducing the CPU times at least by a factor of 2 if the root lower bound is not tight. Otherwise, the performance slightly deteriorates because the symmetry breaking constraints reduce the number of optimal solutions and it may take more time before one is found.

The final series of experiments was aimed at assessing the impact of the location of depots on the performance of the model. In particular, for a selected network ("vypas6" in Table 3) and a fixed number of vehicles ($|K| = 3$) we tried solving the problem while locating each depot independently at a random vertex in the network. In total, more than 1000 trials were made. In some of them, all depots appeared to be assigned to the same vertex, thus representing a single-depot situation (all such cases were tried), in some cases two depots coincided – a case with effectively two depots, otherwise a valid 3-depot case occurs. Fig. 6 summarises the results obtained. As can be seen from the figure, the case when all depots are placed in different vertices is statistically the easiest because the highest part of such instances can be solved within a minute. At the same time, there exist a considerable portion of the instances that could not be solved to optimality even within an hour. The single-depot case, while being more complex in terms of average performance, has better worst-case performance as the portion of unsolved instances is considerably less. Finally, the case when two out of three depots coincide appears to be the most complex one from the statistics perspective.

6. Conclusion

We analysed properties and proposed an approach for solving the CARP that uses the inherent sparsity of the problem to the full extent. Further, the proposed model is flexible enough to accommodate directed, asymmetric and undirected cases. In addition, only implicit presence of information about the location of depots in the model further extends its flexibility allowing any mutual location of depots to be handled.

As our numerical experiments suggest, despite the inherent symmetries in the sparse CARP models, they still can be useful for solving realistic instances defined on large networks, especially if the number of vehicles is small. Furthermore, possibility of placing the depots in different vertices is beneficial as it may reduce the complexity of the problem.

Based on our experience with the proposed approach, we see the following directions for future research. First of all, the symmetry-breaking techniques could be further developed in order to decrease the negative impact of the number of vehicles on the performance of the model. Secondly, in a multidepot setting, a procedure for finding an optimal (partial) assignment of demand arcs to routes based on their proximity to depots could be sought; this might substantially reduce the complexity of the problem.

Appendix A. Capacity inequalities (CI3)

There exist another family of valid inequalities that can be applied to each tour separately. In order to derive them, let us rewrite (2) in a slightly different way:

$$\sum_{a\in A^d} \frac{d_a}{Q} x_a^k \le 1, \quad k \in K. \tag{A.1}$$

Clearly, one obtains a valid (but looser) inequality if takes the summation in the left hand side over some proper subset of A^d. Taking this into account and recalling that $x_a^k \le z_a^k$ one may write

$$\sum_{a\in A'} \frac{d_a}{Q} x_a^k \le \sum_{a\in A'} z_a^k, \tag{A.2}$$

where A' is some subset of A^d. On the other hand, invoking the subtour elimination inequalities (18), it is not hard to understand that the following inequality is valid:

$$\sum_{a\in A''} \frac{d_a}{Q} x_a^k \le \sum_{a\in\delta^-(S)} z_a^k, \tag{A.3}$$

where $S \subset V$ is some set of vertices including the depot vertex, $A'' = \{(i,j) \mid (i,j) \in A^d,\ i,j \in V \setminus S\}$. Now, generalising the two possibilities presented by (A.2) and (A.3) we come up with the following valid inequality:

$$\sum_{a\in A^1\cup A^2} d_a x_a^k \le Q \sum_{a\in\delta^-(S)\cup A^2} z_a^k, \tag{A.4}$$

where S is some subset of vertices containing the depot vertex, $A^1 \subseteq A^d$ – some subset of arcs unreachable from the depot vertex in graph $G(V, A \setminus \delta^-(S))$, A^2 – some subset of A^d.

The separation problem for inequalities (A.4) can be formulated as an MILP (see Appendix B.4) and solved reasonably fast due to a similarity with the mincut problem. It can be seen that in case $\sum_{a\in A^1\cup A^2} d_a < Q$ inequality (A.4) is dominated by (18), which is, in turn, dominated by (19). That is why it is beneficial to extend the separation problem for constraints (A.4) with a constraint ensuring that the sum of demands in the left hand side of this inequality exceeds Q.

Appendix B. Auxiliary MILP formulations used in the paper

All the formulations in this section are easier understood using a notion of the cut polytope. Consider a directed graph $G(V, A)$; for some subset $S \in V$ the cut $\delta(S)$ decomposes V into two subsets $S^1 = S$ and $S^2 = V \setminus S$. Let index k enumerate these two subsets, let x- and z-variables have the following meaning:

- $x_i^k \in \{0, 1\}$ is nonzero iff $i \in S^k$, $k \in \{1, 2\}$;
- $z_a^k \in \{0, 1\}$ is nonzero iff arc $a \in S^k \times S^k \cap A$, $k \in \{1, 2\}$.

Given this notation, the cut polytope $\mathcal{P}_W$ is described by the following inequalities:

$$\begin{aligned}
&x_i^1 + x_i^2 = 1, \quad i \in V\\
&z_a^k \ge x_i^k + x_j^k - 1, \quad a = (i,j) \in A,\ k \in \{1,2\}\\
&z_a^k \le x_i^k, \quad a = (i,j) \in A,\ k \in \{1,2\}\\
&z_a^k \le x_j^k, \quad a = (i,j) \in A,\ k \in \{1,2\}\\
&\sum_{i\in V} x_i^2 \ge 1\\
&x_i^1 = 1, \quad i \in W \subset V
\end{aligned} \tag{B.1}$$

The last two inequalities ensure that all 'special' vertices (depots) are in S^1, and the cut is valid (S^2 is non-empty). Given this formulation, a cut contains all arcs a for which expression $(1 - z_a^1 - z_a^2)$ evaluates to 1.

B.1. Separation problem for subtour elimination constraints (19)

For some route $k' \in K$ the separation problem is:

$$\sum_{a\in A} s_a u_a - 0.5 \sum_{a\in A} w_a (1 - z_a^1 - z_a^2) \to \max \tag{B.2}$$

$$\sum_{a\in A} u_a = 1, \tag{B.3}$$

$$u_a \le z_a^2, \quad a \in A \tag{B.4}$$

$$u_a \in \{0, 1\}, \quad a \in A \tag{B.5}$$

$$(\mathbf{x}, \mathbf{z}) \in \mathcal{P}_W \tag{B.6}$$

where $s_a = \bar{x}_a^{k'}$ – values of the x-variables in the fractional solution, $w_a = \bar{z}_a^{k'}$ – values of z-variables in the fractional solution, W is a singleton corresponding to the depot of route k'.

The sums in the objective function correspond to the r.h.s. and l.h.s. of inequality (19), respectively. The multiplier 0.5 at the second sum is needed because the sum is equal to the value of bidirectional flow through the cut.

B.2. Separation problem for capacity inequalities CI1 (21)

The separation problem is formulated as:

$$u - 0.5\sum_{a\in A} w_a(1 - z_a^1 - z_a^2) \to \max \tag{B.7}$$

$$Qu \geq \sum_{a\in A} s_a(1 - z_a^1) \tag{B.8}$$

$$Qu \leq \sum_{a\in A} s_a(1 - z_a^1) + Q - \epsilon \tag{B.9}$$

$$(\mathbf{x}, \mathbf{z}) \in \mathcal{P}_W \tag{B.10}$$

$$u \in \mathbb{Z}_+ \tag{B.11}$$

where $w_a = \sum_{k\in K} \bar{z}_a^k$, W is a set of all depot vertices, ϵ is a small positive constant (we used $\epsilon = 1$). Variable u defines the value of the r.h.s. of inequality (21), while z-variables define the cut to be used in the l.h.s.

B.3. Separation problem for capacity inequalities CI2 (24)

For any route $k' \in K$, the separation problem is formulated as:

$$0.5\sum_{a\in A} w_a(1 - z_a^1 - z_a^2) + \sum_{a\in A} s_a(z_a^1 - z_a^0) \to \min \tag{B.12}$$

$$z_a^0 = 0, \quad a \in A \setminus A^d \tag{B.13}$$

$$z_a^0 \leq z_a^1, \quad a \in A \tag{B.14}$$

$$\sum_{a\in A} d_a z_a^0 \leq Q' - 1 \tag{B.15}$$

$$(\mathbf{x}, \mathbf{z}) \in \mathcal{P}_W \tag{B.16}$$

$$z_a^0 \in \{0, 1\}, \quad a \in A \tag{B.17}$$

where $w_a = \bar{z}_a^{k'}$, $s_a = \sum_{k\in K\setminus\{k'\}} \bar{x}_a^k$, W is a singleton corresponding to the depot of route k'. Variables z_a^0 define a complement of set A' relative to the set of arcs with both endpoints in S, while variables z_a^1, z_a^2 define the cut $\delta(S)$. The objective defines the value of the l.h.s. of inequality (24).

B.4. Separation problem for capacity inequalities CI3 (A.4)

For any route $k \in K$, the separation problem is formulated as:

$$\sum_{a\in A^d} d_a \bar{x}_a^k(u_a^1 - u_a^2) - Q\sum_{a\in A} \bar{z}_a^k(u_a^2 + u_a^3) \to \max \tag{B.18}$$

$$u_a^1 + z_a^1 \leq 1, \quad a \in A \tag{B.19}$$

$$u_a^2 \leq z_a^1, \quad a \in A \tag{B.20}$$

$$u_a^3 = 0.5(1 - z_a^1 - z_a^2), \quad a \in A \tag{B.21}$$

$$\sum_{a\in A^d} d_a(u_e^1 + u_e^2) \geq Q + 1 \tag{B.22}$$

$$(\mathbf{x}, \mathbf{z}) \in \mathcal{P}_W \tag{B.23}$$

$$u_a^1, u_a^2 \in \{0, 1\}, \quad a \in A \tag{B.24}$$

$$u_a^3 \in \mathbb{R}, \quad a \in A \tag{B.25}$$

where variables u_a^1 and u_a^2 describe sets A^1 and A^2, correspondingly, and W is a singleton corresponding to the depot of route k. The objective defines the violation of inequality (A.4).

References

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical Programming*, *8*, 146–164.

Balas, E., Ceria, S., & Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, *58*, 295–324.

Bartolini, E., Cordeau, J.-F., & Laporte, G. (2013). An exact algorithm for the capacitated arc routing problem with deadheading demand. *Operations Research*, *61*(2), 315–327.

Belenguer, J. M., & Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, *30*, 705–728.

Bode, C., & Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, *60*(5), 1167–1182.

Bode, C., & Irnich, S. (2014). The shortest-path problem with resource constraints with (k,2)-loop elimination and its application to the capacitated arc-routing problem. *European Journal of Operational Research*, *238*(2), 415–426.

Corberan, A., Oswald, M., Plana, I., Reinelt, G., & Sanchis, J. M. (2012). New results on the windy postman problem. *Mathematical Programming Series A*, *132*, 309–332.

Corberan, A., & Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, *56*(1), 50–69.

Golden, B. L., & Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, *11*, 305–315.

Gottlieb, E. S., & Rao, M. R. (1990). The generalized assignment problem: Valid inequalities and facets. *Mathematical Programming*, *46*, 31–52.

Grandinetti, L., Guerriero, F., Laganá, D., & Pisacane, O. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, *30*, 705–728.

Hertz, A. (2005). Recent trends in arc routing. In M. C. Golumbic, & I. B.-A. Hartman (Eds.), *Graph theory, combinatorics and algorithms. Vol. 34: Operations research/computer science interfaces series* (chap. 9, pp. 215–236). USA: Springer.

Karger, D. R., & Stein, C. (1996). A new approach to the minimum cut problem. *Journal of the ACM*, *43*, 601–640.

Longo, H., de Aragão, M. P., & Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the CVRP. *Computers & Operations Research*, *33*(6), 1823–1837.

Maniezzo, V. (2004). Algorithms for large directed CARP instances: Urban solid waste collection operational support. (Technical report UBLCS-2004-16). Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy.

Pisinger, D. (1996). A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, *45*, 758–767.