OR Spectrum (2021) 43:497–542
https://doi.org/10.1007/s00291-021-00635-y

ORIGINAL ARTICLE

Mixed-integer programming approaches for the time-constrained maximal covering routing problem

Markus Sinnl[1,2]

Received: 13 June 2019 / Accepted: 20 April 2021 / Published online: 10 May 2021

Abstract

In this paper, we study the recently introduced time-constrained maximal covering routing problem. In this problem, we are given a central depot, a set of facilities, and a set of customers. Each customer is associated with a subset of the facilities which can cover it. A feasible solution consists of k Hamiltonian cycles on subsets of the facilities and the central depot. Each cycle must contain the depot and must respect a given distance limit. The goal is to maximize the number of customers covered by facilities contained in the cycles. We develop two exact solution algorithms for the problem based on new mixed-integer programming models. One algorithm is based on a compact model, while the other model contains an exponential number of constraints, which are separated on-the-fly, i.e., we use branch-and-cut. We also describe preprocessing techniques, valid inequalities and primal heuristics for both models. We evaluate our solution approaches on the instances from literature and our algorithms are able to find the provably optimal solution for 267 out of 270 instances, including 123 instances, for which the optimal solution was not known before. Moreover, for most of the instances, our algorithms only take a few seconds, and thus are up to five magnitudes faster than previous approaches. Finally, we also discuss some issues with the instances from literature and present some new instances.

Keywords Routing problems · Covering problems · Branch-and-cut

✉ Markus Sinnl
markus.sinnl@jku.at

[1] Institute of Production and Logistics Management, Johannes Kepler University Linz, Linz, Austria

[2] JKU Business School, Johannes Kepler University Linz, Linz, Austria

1 Introduction

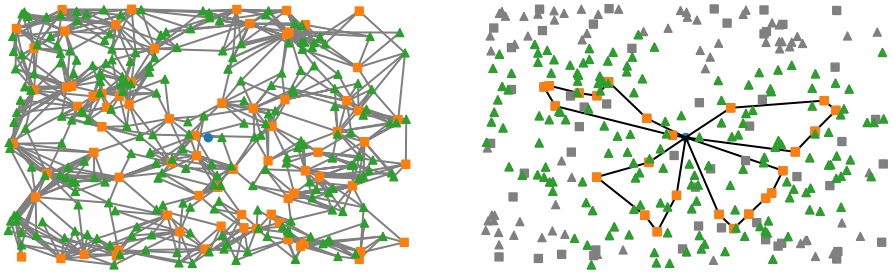
Vehicle routing problems and covering problems are important and fundamental problems in Operations Research and Logistics. In this paper, we study the recently introduced *time-constrained maximal covering routing problem (TCMCRP)*, which is a generalization of well-known routing problems such as the *orienteering problem* (see, e.g., Golden et al. (1987); Gunawan et al. (2016)), and in particular the *team orienteering problem* (see, e.g., Chao et al. (1996)), and the *maximal covering location problem* (see, e.g., Church and Velle (1974)). The problem was introduced in Amiri and Salari (2019), and applications in health care were discussed.

In the TCMCRP, we are given a directed graph $G = (V, A)$, where $V = 0 \cup F \cup C$ is the set of vertices. The vertex 0 represents the *central depot*, F the set of *facilities* and C the set of *customers*. The arc set $A = A_{0F} \cup A_{FC}$ is defined as set of *routing arcs* $A_{0F} = \{(i, i') : i, i' \in 0 \cup F\}$ (i.e., the complete directed graph on $0 \cup F$) and *assignment arcs* $A_{FC} \subseteq \{(i, j) : i \in F, j \in C\}$ (i.e., the assignment arcs are a subset of all possible facility/customer connections). Each arc $(i, i') \in A_{0F}$ has a travel distance $d_{ii'} > 0$ associated with it. Moreover, let P represent the set of $k = |P|$ available vehicles, and let L_p be a *distance limit* for each $p \in P$. A feasible solution consists of one Hamiltonian cycle on a subset of $0 \cup F$ for each $p \in P$. Each of these cycles must contain 0 and respect the distance limit L_p. A facility F can only appear in at most one cycle. We will also refer to these cycles as *tours* $T_1, \ldots, T_k$. A customer is *covered* by a solution if there exists an assignment arc in A_{FC} between the customer and a facility visited in one of the tours. The goal is to find a feasible solution which maximizes the number of covered customers.

Note that in the instances from literature, L_p is the same for all vehicles, and the distance function is Euclidean. Both are common assumptions in vehicle routing problems. Moreover, the mixed-integer programming (MIP) formulation presented in Amiri and Salari (2019) implicitly assumes that the distance function satisfies the triangle inequality[1]. In this work, we present two new MIP formulations for the problem. The first formulation assumes that the distance function satisfies the triangle inequality and L_p is the same for all vehicles. The second formulations do not need these assumptions. Figure 1 shows an exemplary instance graph of the TCMCRP and its optimal solution for four vehicles and a given distance limit.

Contribution and paper outline The TCMCRP was recently introduced in Amiri and Salari (2019), where the authors presented a flow-based MIP model, an iterated local search, a tabu search and a variable neighborhood search for it. They evaluated their algorithms on instances derived from the well-known TSPLIB (Reinelt 1991). In this paper, we develop two exact solution algorithms for the problem based on new MIP-models. One algorithm is based on a compact model, i.e., a model with a polynomial number of variables and constraints. The other model contains an exponential number of constraints. We also describe preprocessing techniques, valid

[1] cf. constraint (12) in their formulation and the associated explanation.



(a) Instance-graph. **(b)** Optimal solution.

Fig. 1 Exemplary instance graph of the TCMCRP and its optimal solution for four vehicles and a given distance limit. The blue circle is the central depot, orange boxes are facilities and green triangles are customers. The gray edges in 1a between facilities and customers denote which customers are covered by each facility. For better readability, the arcs between facilities, and facilities and the central depot are not displayed. In the solution 1b, the arcs of the optimal solution are indicated in black, and all facilities and customers not in the solution are grayed out

inequalities and primal heuristics for both models. Since for the compact model the presented set of valid inequalities has exponential size, we use branch-and-cut (see, e.g., Conforti et al. (2014)) in both solution algorithms.

In a computational study, we evaluate our solution approaches on the instances from Amiri and Salari (2019). The study reveals that our algorithms are able to find the provably optimal solution for 123 instances, where the optimal solution was not known before, and only 3 instances remain unsolved. Moreover, for most of the instances, our algorithms only take a few seconds, and thus are up to five magnitudes faster than the algorithms presented in Amiri and Salari (2019). Finally, we also discuss some issues with the instances used in Amiri and Salari (2019) (e.g., not all customers have a facility associated with it, and the optimal solution often contains just all reachable customers; the calculation of L_p is not the same as described in the paper), and introduce a set of new and more difficult instances.

The paper is organized as follows: In the remainder of this section, we give an overview of related work. In Sect. 2, we present our two new MIP-models, together with preprocessing/variable-fixing procedures and valid inequalities. In Sect. 3, we discuss additional details about the developed branch-and-cut framework, such as separation procedures for the valid inequalities and primal heuristics. Section 4 contains the computational study, and Sect. 5 concludes the paper.

Related work As the studied problem is a quite general routing/covering problem, there is naturally a vast number of related work; the paper introducing the TCMCRP (Amiri and Salari 2019) contains a quite exhaustive and up-to-date discussion of related problems. We thus focus the discussion about related work on the *team orienteering problem (TOP)*, since both of our models are extensions of models for the TOP. For a general overview on routing problems, we refer to, e.g., Toth and Vigo (2014) and for a general overview on facility location/covering problems, we refer to, e.g., Laporte et al. (2015) (Chapter 5).

The TOP is an extension of the *orienteering problem (OP)* to multiple vehicles. The OP was first introduced in Tsiligirides (1984). In the OP, we are given a central depot, a set of profitable customers which can be visited, and a distance limit. The goal is to find the most profitable Hamiltonian cycle on a subset of the customers, the cycle must also contain the depot and respect a given distance limit. Sometimes the OP is defined with a start depot and an end depot, and a Hamiltonian path on a subset of the customers from start to end is searched. Moreover, there is a variant of the OP called *selective traveling salesman problem* (see Gendreau et al. (1998); Laporte and Martello (1990)), with additional compulsory vertices, which must be in any feasible cycle/path. There exist also many other variants with additional side-constraints (such as capacities or time-windows), for more details, see, e.g., the survey Gunawan et al. (2016). Regarding successful exact approaches for the OP, there are several papers (Fischetti et al. 1998; Gendreau et al. 1998; Leifer and Rosenwein 1994) using branch-and-cut approaches based on models with *generalized subtour elimination constraints (GSECs)/connectivity cuts(CCs)*. Similar GSECs/CCs will also be used in our approaches.

The TOP was introduced in Chao et al. (1996), where a heuristic was proposed. The TOP extends the OP by introducing k (homogeneous) vehicles, i.e., the goal is now to find k Hamiltonian cycles/paths containing the depot and respecting the distance limit, instead of a single one. Note that the TOP can be seen as a special case of the TCMCRP, with a one-to-one-correspondence between facilities and customers. A variant of the TOP with heterogeneous vehicles (denoted as *multiple tour maximum collection problem*) was considered in Butt and Ryan (1999). Several column generation, and branch-and-price(-and-cut) approaches were proposed for the TOP (Boussier et al. 2007; Butt and Ryan 1999; Keshtkaran et al. 2016; Poggi et al. 2010). An exponential size formulation using GSECs and solved by branch-and-cut was developed in Dang et al. (2013); El-Hajj et al. (2016). In Bianchessi et al. (2018), the authors presented a compact model for the TOP based on a formulation of Maffioli and Sciomachen (1997) for the *sequential ordering problem*. They strengthen the model by separating CCs and were able to solve additional instances to optimality.

Aside from the TOP, another strongly related problem to the TCMCRP is the *time-constrained maximal covering salesman problem (TCMCSP)*, which was introduced in Naji-Azimi and Salari (2014). The TCMCSP is the single-vehicle variant of the TCMCRP. In Naji-Azimi and Salari (2014), the authors presented a flow-based MIP model and some heuristics for the TCMCSP (Amiri and Salari (2019) is basically the extension of the approaches in Naji-Azimi and Salari (2014) to the TCMCRP). In Ozbaygin et al. (2016), an exact solution algorithm based on GSECs for a variant of the TCMCSP was proposed.

2 Mixed integer programming models and valid inequalities

We first present the compact model, together with its associated preprocessing and valid inequalities, and then the exponential-sized model, together with its associated preprocessing and valid inequalities. Note that for the compact model the presented

set of valid inequalities has exponential size. Thus, we also use branch-and-cut in the algorithm based on the compact model. For later use, for a subset $S \subseteq 0 \cup F$, let $\delta^+(S) = \{(i,i') \in A_{0F} : i \in S, i' \notin S\}$ and $\delta^-(S) = \{(i,i') \in A_{0F} : i \notin S, i' \in S\}$ be the set of outgoing, resp., incoming arcs of the cut induced by S. In both models, we allow solutions using less than k vehicles, as we present some valid inequalities based on optimality-arguments, namely constraints (C-FD). These inequalities may cut off some optimal solutions, if an instance has multiple optimal solutions, and the remaining optimal solutions may use less than k vehicles. This situation is detailed below in Example 1.

2.1 Compact model

This formulation follows the approach proposed for the TOP in Bianchessi et al. (2018). For this formulation and its associated valid inequalities, we assume that the distance function fulfills the triangle inequality and that the vehicles are homogeneous. Let L denote the homogeneous distance limit, i.e., $L_p = L$ for $p \in P$. Let binary variables $x_{ii'} = 1$, for $(i,i') \in A_{0F}$, iff arc (i,i') is traveled by a vehicle in the solution. Let binary variables $y_i = 1$, $i \in F$, iff facility i is visited in the solution, and binary variables $z_j = 1$, $j \in C$, iff customer j is covered by the solution. Moreover, let continuous variables $f_{ii'}$ for $(i,i') \in A_{0F}$ indicate the traveled distance from the central depot at facility i' for a vehicle arriving from i. Let integer variable $w \in \{1, \ldots, k\}$ indicate the number of vehicles used in the solution. The compact model, denoted by (C), is as follows.

$$\max \quad \sum_{j \in C} z_j \tag{C-OBJ}$$

$$s.t. \quad \sum_{(i,j) \in A_{FC}} y_i \geq z_j \qquad \forall j \in C \tag{C-LINK}$$

$$\sum_{(i,i') \in A_{0F}} x_{ii'} = y_i \qquad \forall i \in F \tag{C-OUT}$$

$$\sum_{(i',i) \in A_{0F}} x_{i'i} = y_i \qquad \forall i \in F \tag{C-IN}$$

$$\sum_{(0,i) \in A_{0F}} x_{0i} = w \tag{C-OUT0}$$

$$\sum_{(i,0) \in A_{0F}} x_{i0} = w \tag{C-IN0}$$

$$f_{0i} = d_{0i} x_{0i} \qquad \forall i \in F \tag{C-FLOW0}$$

$$\sum_{i' \in \delta^+(i)} f_{ii'} - \sum_{i' \in \delta^-(i)} f_{i'i} = \sum_{(i,i'') \in A_{0F}} d_{ii''} x_{ii''} \qquad \forall i \in F \tag{C-FLOW}$$

$$f_{ii'} \leq (L - d_{i'0}) x_{ii'} \qquad \forall (i, i') \in A_{0F}, i' \neq 0 \tag{C-DIST}$$

$$y_i \in \{0, 1\} \qquad \forall i \in F \tag{C-Y}$$

$$z_j \in \{0, 1\} \qquad \forall j \in C \tag{C-Z}$$

$$x_{ii'} \in \{0, 1\} \qquad \forall (i, i') \in A_{0F} \tag{C-X}$$

$$f_{ii'} \geq 0 \qquad \forall (i, i') \in A_{0F} \tag{C-F}$$

$$w \in \{1, \ldots, k\} \tag{C-W}$$

The objective function (C-OBJ) and constraints (C-LINK) ensure that a customer is only counted in the objective function, if a facility covering it is visited in the solution. Constraints (C-IN) and (C-OUT) ensure that each visited facility has one incoming and one outgoing arc. Constraints (C-OUT0) and (C-IN0) make sure that there are exactly w vehicles leaving and entering the depot. The solution defined by the previous four set of constraints (plus integrality of the variables) will consist of w or more cycles, with w of these cycles containing the central depot, i.e., subtours are possible. Potential subtours are prohibited using flow-conservation constraints (C-FLOW0) and (C-FLOW). These constraints ensure that the flow-variables $f_{ii'}$ encode the distance traveled from the depot to facility i'. We note that there is no feasible way to set the values of the flow-variables of arcs contained in a potential subtour, as the total traveled distance can only increase along any selected path. Thus, these constraints forbid subtours. Moreover, together with constraints (C-DIST), these constraints also model the distance limit of a tour: For any facility i in the solution, the traveled distance must allow to go back from i to the central depot within the distance limit. Note that constraints (C-DIST) need that the distance function fulfills the triangle inequality, otherwise there could be a shorter, non-direct connection from i' to the central depot, and the constraints would be too restrictive. Constraints (C-DIST) also link the flow-variables and the arc-variables, i.e., flow is only allowed on an arc, if the arc is selected in the solution. Finally, constraints (C-Y) to (C-W) define the variables.

Valid Inequalities Next, we present some valid inequalities for (C), including variable-fixing procedures, which can be applied in a preprocessing step.

The first set of inequalities is concerned with dominance between facilities. The inequalities use optimality-arguments, i.e., they may cut off some feasible solutions, but there is at least one optimal solution fulfilling all of them.

Theorem 1 *Let $i, i' \in F$, and $C(i'') = \{j \in C : (i'', j) \in A_{FC}\}$ for $i'' = i, i'$. Suppose $C(i') \subseteq C(i)$. Then, the following facility dominance inequalities*

$$y_i + y_{i'} \leq 1 \tag{C-FD}$$

are valid for (C) , *i.e., there exists at least one optimal solution fulfilling all of them. Moreover,*

$$x_{ii'} = 0 \text{ and } x_{i'i} = 0 \tag{C-FDA}$$

is also valid.

Proof As i covers at least the same customers also covered by i', there is no improvement in the objective function by including i' in any solution containing i. Moreover, as the distance function fulfills the triangle inequality, any tour containing both i and i' will never be shorter than a tour just containing i. Thus, for each solution containing both i and i', another solution just containing i with the same objective, and same or shorter tour-lengths can be constructed. □

We note that inequalities (C-FD) may cut off all optimal solutions using exactly k vehicles, as shown in the following example.

Example 1 Let $k = 3$, $F = \{i_1, i_2, i_3\}$, $C = \{j_1, j_2, j_3\}$ and $A_{FC} = \{(i_1, j_1), (i_1, j_2), (i_2, j_1), (i_3, j_3)\}$. Moreover, suppose that due to the distances and the given distance limit, each tour can only visit one facility. We have two optimal solutions: (i) using two vehicles, with one visiting i_1 and one visiting i_3, (ii) using three vehicles, and each vehicle visits one facility. As facility i_1 dominates facility i_2, facility dominance inequalities (C-FD) only allow solution (i).

The following variable fixing exploits the distance limit L, similar ideas have been used in Bianchessi et al. (2018); Dang et al. (2013); El-Hajj et al. (2016) for the TOP and in Ozbaygin et al. (2016) for the TCMCSP, they can be seen as special-case of the *path inequalities* for the OP proposed in Fischetti et al. (1998).

Theorem 2 *Let* $i \in F$ *with* $d_{0i} + d_{i0} > L$. *Then*

$$y_i = 0 \tag{C-FIXF}$$

is valid for (C).

Let $i, i' \in F$ *with* $d_{0i} + d_{ii'} + d_{i'0} > L$. *Then*

$$x_{ii'} = 0 \tag{C-FIXA}$$

is valid for (C).

Let $F(j) = \{i \in F : (i,j) \in A_{FC}\}$ *for* $j \in C$. *If for all* $i \in F(j)$, *we have* $d_{0i} + d_{i0} > L$, *then*

$$z_j = 0 \tag{C-FIXC}$$

is valid for (C).

Proof Obvious, as the distance limit does not allow the shortest cycle containing (i, i'), resp., i. Moreover, if no facility covering j can be reached given the distance limit, j cannot be in any solution. □

The following global constraint (C-DISTG) on the length of all tours can also be added, as well as constraints (C-FLOWER) which impose lower bounds on the flow-variables $f_{ii'}$ (see Bianchessi et al. (2018)).

$$\sum_{(i,i')\in A_{0F}} d_{ii'} x_{ii'} \leq wL, \tag{C-DISTG}$$

$$f_{ii'} \geq (d_{0i} + d_{ii'}) x_{ii'}. \tag{C-FLOWER}$$

While the inequalities in the model already ensure that the solution is connected (and hence, consists of k cycles containing the central depot), the model can be strengthened by adding connectivity cuts (C-CC) (see, e.g., Bianchessi et al. (2018)).

$$\sum_{(i'',i')\in\delta^-(S)} x_{i''i'} \geq y_i \quad \forall S\subseteq F, i \in S : |S| \geq 2 \tag{C-CC}$$

The inequalities ensure that there is at least one arc going from $(F \setminus S) \cup 0$ to S, if a facility i in S is chosen in a solution. While these inequalities would also ensure that the solution is connected, we cannot replace the flow-conservation constraints (C-FLOW) with them, as the flow-conservation constraints are also needed for modeling the distance limit. As there are exponentially many inequalities (C-CC), we separate them on-the-fly in a branch-and-cut, see Sect. 3.1 for the separation. In the following, we present various liftings of these inequalities.

Theorem 3 *Let* $S\subseteq F$ *with* $|S| \geq 2$ *and* $\mathcal{F} \subseteq S$ *with* $d_{0i} + d_{ii'} + d_{i'0} > L$ *for all pairs* $i, i' \in \mathcal{F}$. *Then, the following inequality is valid*

$$\sum_{(i'',i')\in\delta^-(S)} x_{i''i'} \geq \sum_{i\in\mathcal{F}} y_i. \tag{C-CC-FIXA}$$

Proof If any of the facilities in $\mathcal{F}$ is in a solution, at least one of the arcs associated with the variables on the left-hand-side must be taken to ensure connectivity to the central depot. Moreover, each tour in a solution can only contain at most one of the facilities in $\mathcal{F}$, due to the condition defining $\mathcal{F}$. Thus, each y_i with value one on the right-hand-side needs its own tour. This implies that at least as many arcs associated with variables on the left-hand-side must be in a solution as there are facilities from $\mathcal{F}$ in this solution. □

Let *LB* be a given lower bound for the objective value, e.g., the value of the current incumbent solution during branch-and-cut. Using *LB*, an optimality-based lifting of (C-CC) may be possible.

Theorem 4 *Let $S \subseteq F$ with $|S| \geq 2$. Let $Z(S) = |\{j : (i,j) \in A_{FC}, i \notin S\}|$, i.e., the number of customers, which can be served by facilities not in S. Suppose $Z(S) \leq LB$, then the following inequality is valid*

$$\sum_{(i'',i') \in \delta^-(S)} x_{i''i'} \geq 1 \qquad \text{(C-CC-OPT)}$$

Proof As $Z(S) \leq LB$, facilities outside of S cannot serve enough customers to provide an improved solution, thus, at least one tour must visit facilities in S to provide a solution with value better than LB. □

Another lifted version of (C-CC) can be obtained using the facility dominance inequalities (C-FD).

Theorem 5 *Let $i, i' \in S \subseteq F$, and $C(i'') = \{j \in C : (i'',j) \in A_{FC}\}$ for $i'' = i, i'$. Suppose $C(i') \subseteq C(i)$. Then, the following inequality is valid*

$$\sum_{(i''',i'') \in \delta^-(S)} x_{i'''i''} \geq y_i + y_{i'} \qquad \text{(C-CC-FD)}$$

Proof Both i, i' are in S and there exists an optimal solution, where at most one of them will be visited, following from the same arguments as in the proof of Theorem 1. □

Finally, there is also a version of inequalities (C-CC), which use the customer variables on the right-hand-side.

Theorem 6 *Let $j \in C$ and $F(j) = \{i \in F : (i,j) \in A_{FC}\}$. Let $S \subseteq F$ and suppose $F(j) \subseteq S$. Then, the following inequality is valid*

$$\sum_{(i',i) \in \delta^-(S)} x_{i'i} \geq z_j \qquad \text{(C-CC-CUST)}$$

Proof If z_j is one, customer j is covered in the solution. Thus, at least one of the facilities in $F(j)$ must be in the solution. As $F(j) \subseteq S$, at least one arc must go into S to allow at least one of the facilities in $F(j)$ to be connected to the central depot. □

Note that in any LP-relaxation solution $(x^*, y^*, z^*, f^*, w^*)$ of (C), $z_j^* \geq y_i^*$ for $(i,j) \in A_{FC}$, as the objective function maximizes $\sum_{j \in C} z_j$. Thus, whenever there is a violated inequality (C-CC) with S fulfilling the conditions of (C-CC-CUST), there is an inequality (C-CC-CUST) with at least the same violation.

2.2 Exponential model

This model follows the formulation of Dang et al. (2013) and El-Hajj et al. (2016) for the TOP and allows for heterogeneous vehicles, i.e., different distance limits L_p for each $p \in P$. Binary variables z_j, $j \in C$ have the same meaning as in model (C).

Let binary variables $y_i^p = 1$, $i \in F$, $p \in P$, iff facility i gets visited by the tour of vehicle $p \in P$. Moreover, let binary variables $x_{ii'}^p = 1$, for $(i, i') \in A_{0F}$, iff arc (i, i') is traveled by vehicle $p \in P$ in the solution. Let binary variable $w_p = 1, p \in P$ iff vehicle p is used in the solution. The model, denoted by (E), is as follows.

$$\max \quad \sum_{j \in C} z_j \tag{E-OBJ}$$

$$s.t. \quad \sum_{p \in P} \sum_{(i,j) \in A_{FC}} y_i^p \geq z_j \qquad \forall j \in C \tag{E-LINK}$$

$$\sum_{p \in P} y_i^p \leq 1 \qquad \forall i \in F \tag{E-ONEF}$$

$$\sum_{(i,i') \in A_{0F}} x_{ii'}^p = y_i^p \qquad \forall i \in F, p \in P \tag{E-OUT}$$

$$\sum_{(i',i) \in A_{0F}} x_{i'i}^p = y_i^p \qquad \forall i \in F, p \in P \tag{E-IN}$$

$$\sum_{(0,i') \in A_{0F}} x_{0i'}^p = w_p \qquad \forall p \in P \tag{E-OUT0}$$

$$\sum_{(i',0) \in A_{0F}} x_{i'0}^p = w_p \qquad \forall p \in P \tag{E-IN0}$$

$$\sum_{(i'',i') \in \delta^-(S)} x_{i''i'}^p \geq y_i^p \qquad \forall S \subseteq F, i \in S : |S| \geq 2, p \in P \tag{E-CC}$$

$$\sum_{(i,i') \in A_{0F}} d_{ii'} x_{ii'}^p \leq L_p w_p \qquad \forall p \in P \tag{E-DIST}$$

$$y_i^p \in \{0, 1\} \qquad \forall i \in F, p \in P \tag{E-Y}$$

$$z_j \in \{0, 1\} \qquad \forall j \in C \tag{E-Z}$$

$$x_{ii'}^p \in \{0, 1\} \qquad \forall (i, i') \in A_{0F}, p \in P \tag{E-X}$$

$$w_p \in \{0, 1\} \quad p \in P \tag{E-W}$$

The objective function (E-OBJ) and constraints (E-LINK) are the same as in model (C). Constraints (E-ONEF) make sure that each facility is only visited by one vehicle (in case of distances satisfying the triangle inequality, these constraints are redundant, as using a facility in more than one tour will only result in larger distances). For each vehicle $p \in P$, constraints (E-OUT) and (E-IN) ensure that if a facility is visited by vehicle p, the vehicle enters and leaves the facility. Moreover, constraints (E-OUT0) and (E-IN0) ensure that each used vehicle enters and leaves the depot. Thus, the previous four sets of constraints make sure that the solution for each used vehicle consists of one or more cycles, and one of these cycles starts and ends at the depot. Connectivity cuts (E-CC) ensure that a facility i visited by a vehicle p must be connected to the central depot: For any set S containing i, at least one arc going from $(F \setminus S) \cup 0$ to S must be taken by vehicle p. Thus, the solution contains exactly one cycle for each vehicle, and this cycle must contain the central depot. As the set of inequalities (E-CC) is of exponential size, we separate them on-the-fly, when they are violated. Separation of the inequalities is discussed in Sect. 3.1. The distance limit is enforced by constraints (E-DIST). The variables are defined by constraints (E-Y) to (E-W).

Valid Inequalities Similar to model (C), several valid inequalities and variable-fixing procedures can be defined. Some of these inequalities are adaptions of the inequalities for (C); however, there are also additional inequalities. While formulation (E) makes no assumption on the distance function, all of the valid inequalities need that the distance function is symmetric and fulfills the triangle inequality.

Facility domination inequalities (C-FD) can be adapted as follows for facilities $i, i' \in F$ to obtain inequalities (E-FD)

$$\sum_{p \in P} (y_i^p + y_{i'}^p) \leq 1. \tag{E-FD}$$

Adaption of the variable fixings (C-FDA), (C-FIXF), (C-FIXA) and (C-FIXC) are straightforward. We denote them as (E-FDA), (E-FIXF), (E-FIXA) and (E-FIXC). Moreover, the following conflict-constraints (E-CLQ) can be derived (see also the *incompatibility clique cuts* for the TOP in Dang et al. (2013) and El-Hajj et al. (2016))

Theorem 7 *Let $p \in P$ and $\mathcal{F} \subseteq F$ with $d_{0i} + d_{ii'} + d_{i'0} > L_p$ for all pairs $i, i' \in \mathcal{F}$. Then, the following inequality is valid.*

$$\sum_{i \in \mathcal{F}} y_i^p \leq w_p. \tag{E-CLQ}$$

Proof Due to the assumption that the distances fulfill the triangle inequality and the condition $d_{0i} + d_{ii'} + d_{i'0} > L_p$ holds for all pairs $i, i' \in \mathcal{F}$, at most one of the facilities $i \in \mathcal{F}$ can be visited by vehicle p. □

There is an exponential number of inequalities (E-CLQ). We thus do not use all of them in our solution framework, but only a subset, resp., we separate them

on-the-fly, for more details on this see Sect. 3.1. Inequalities (E-CLQ) can be used for lifting (E-CC) as shown in the following.

Theorem 8 *Let $S \subseteq F$ with $|S| \geq 2$. Let $p \in P$ and $\mathcal{F} \subseteq S$ with $d_{0i} + d_{ii'} + d_{i'0} > L_p$ for all pairs $i, i' \in \mathcal{F}$. Then, the following inequality is valid*

$$\sum_{(i'',i') \in \delta^-(S)} x^p_{i''i'} \geq \sum_{i \in \mathcal{F}} y^p_i \tag{E-CC-CLQ}$$

Proof Only at most one of the variables y^p_i on the right-hand-side can be one in a feasible solution, due to the assumption that the distances fulfill the triangle inequality and the condition $d_{0i} + d_{ii'} + d_{i'0} > L_p$ holds for all pairs $i, i' \in \mathcal{F}$. □

Moreover, a lifted version of (E-CC) using facility dominance, similar to (C-CC-FD) can also be defined.

Theorem 9 *Let $p \in P$, $i, i' \in S \subseteq F$, and $C(i'') = \{j \in C : (i'', j) \in A_{FC}\}$ for $i'' = i, i'$. Suppose $C(i') \subseteq C(i)$. Then, the following inequality is valid*

$$\sum_{(i''',i'') \in \delta^-(S)} x^p_{i'''i''} \geq y^p_i + y^p_{i'} \tag{E-CC-FD}$$

Proof Similar to the proof of Theorem 5. □

Symmetry Breaking Inequalities In case $L_p = L, p \in P$, there can be symmetric solutions. In order to break these symmetries, the following set of inequalities (E-SYM) can be used. Let $p_1, p_2, \ldots, p_k$ denote an arbitrary ordering of the vehicles. Inequalities (E-SYM) impose that a vehicle with lower index needs to visit at least as many facilities in its tour than a vehicle with a higher index (see also Dang et al. (2013) and El-Hajj et al. (2016) for similar inequalities for the TOP)

$$\sum_{i \in F} y^{p_\ell}_i \geq \sum_{i \in F} y^{p_{\ell+1}}_i, \quad 1 \leq \ell \leq k-1. \tag{E-SYM}$$

3 Algorithmic frameworks

In this section, we discuss further details of our solution frameworks based on models (*C*), resp., (*E*). The solutions frameworks were implemented in C++ using CPLEX 12.9 as MIP solver. In Sect. 4, we report results obtained by several combinations of the ingredients presented in this section.

3.1 Separation algorithms

Model (C) has polynomial size, however, the family of valid inequalities (C-CC) is of exponential size. Thus, we do not add all of them in the beginning, but separate them on-the-fly, when they are violated, i.e., we use branch-and-cut. The same holds for inequalities (E-CC) of model (E). In fact, the separation procedure for both families of inequalities is the same, the only difference is that inequalities (C-CC)are defined on variables x, y, and (E-CC) on x^p, y^p, $p \in P$. Naturally, all the lifted versions/variants of (C-CC) and (E-CC) are also families of inequalities with exponential size. Finally, the family of inequalities (E-FD) of (E) is also of exponential size.

For separation, we used the `LazyConstraintCallback`, which gets called by CPLEX to check integer solutions and the `UserCutCallback`, which gets called by CPLEX to check fractional solutions. When using cuts which remove integer solutions, nonlinear reductions and dual presolve reductions of CPLEX must be deactivated. CPLEX does this automatically, when the `LazyConstraintCallback` is used CPLEX (2020a, 2020b).

Separation of (C-CC) *and* (E-CC) *for fractional solutions* In case (x^*, y^*) is fractional, it is well-known that connectivity cuts like (C-CC) can be separated using maximum flow computations on a graph, where the arc capacities are set to x^* (see, e.g., Bianchessi et al. (2018) and Fischetti et al. (1998)). A connectivity cut with facility i on the right-hand-side is violated if the maximum flow from the central depot 0 to i is less than the value of y_i^*. Let S be the set containing i in such a minimum cut. The set S is giving a violated constraint (C-CC). In order to speed up the separation, we sort the facilities in decreasing order by the values of y_i^* and separate in this order. Whenever we find a violated inequality, we remove all facilities in S for separation of further inequalities. The maximum flow/minimum cut computation is done using the algorithm of Cherkassky and Goldberg (1995). This algorithm may also return a second minimum cut S', and in such a case, we also add the violated constraint induced by S'. Moreover, we add a small $\epsilon = 10^{-5}$ to all capacities before separation to favor the separation of *minimal cardinality cuts*, see, e.g., Koch and Martin (1998) for more details on the last two techniques. Separation of inequalities (E-CC) is done in a similar way.

Separation of (E-CC) *for integer solutions* As the flow-conservation constraints (C-FLOW) already ensure connectivity of the solution in model (C), inequalities (C-CC) will never be violated when (x^*, y^*) is integer. In model (E), when (x^{p*}, y^{p*}) is integer, the induced solution will consist of one or more cycles for each vehicle p. For each cycle S not containing the central depot 0, a constraint (E-CC) is added. These cycles can be found by finding the connected components of the induced solution using, e.g., a breadth-first search. Each component not containing the central depot is such a cycle S. For i on the right-hand-side of (C-CC), we take a facility with the largest number of associated customers, ties are broken by taking the one with smallest index.

Separation of Inequalities (E-CLQ) The set $\mathcal{F}$ of facilities on the left-hand-side of inequalities (E-CLQ) for a given vehicle p is a clique in the graph with edge set

$E = \{\{i, i'\} \in F \times F : d_{0i} + d_{ii'} + d_{i'0} > L_p\}$. Thus, for exact separation of these inequalities, we would need to solve the NP-hard *maximum weighted clique problem* (see, e.g.,(Bomze et al. 1999)) with LP-values (y^{p*}) as vertex weights. To allow for fast separation, we have implemented a greedy heuristic for separating these inequalities. Let $deg(i)$ be the degree of a vertex in the graph defined by (F, E). The heuristic is described in Algorithm 1.

Input: instance $(V = (0, F, C), A, d, k, L)$, LP-values (y^{p*}, w^*) and $\epsilon = 10^{-5}$
Output: (possibly empty) set $ViolIneqs$ of separated inequalities (E-CLQ)

1 $ViolIneqs \leftarrow \emptyset$;
2 **for** $p \in 1, \ldots, k$ **do**
3 $\quad E = \{\{i, i'\} \in F \times F : d_{0i} + d_{ii'} + d_{i'0} > L_p\}$;
4 $\quad E' \leftarrow \{\{i, i'\} \in E : y_i^{p*} > \epsilon$ and $y_{i'}^{p*} > \epsilon\}$;
5 $\quad E' \leftarrow$ sort E' in descending order by $deg(i)y_i^{p*} + deg(i')y_{i'}^{p*}$;
6 $\quad$ **for** $\{i, i'\}$ *in* E' **do**
$\quad\quad$ /* try to build a violated inequality (E-CLQ) starting from the current edge $\{i, i'\}$ */
7 $\quad\quad \mathcal{F} = \{i, i'\}$;
8 $\quad\quad add \leftarrow true$;
$\quad\quad$ /* try to add more vertices to the clique $\mathcal{F}$ */
9 $\quad\quad$ **do**
10 $\quad\quad\quad add \leftarrow false$;
$\quad\quad\quad$ /* collect all vertices, which would form a clique in E' with the vertices in $\mathcal{F}$ */
11 $\quad\quad\quad F' \leftarrow i'' \in F \setminus \mathcal{F} : \{i'', i''\} \in E'$ for all $i'' \in \mathcal{F}$;
12 $\quad\quad\quad$ **if** $F' \neq \emptyset$ **then**
13 $\quad\quad\quad\quad add \leftarrow true$;
$\quad\quad\quad\quad$ /* we use a score based on degree in (F, E) and y^{p*} as selection criterion */
14 $\quad\quad\quad\quad i^* \leftarrow argmax_{i'' \in F'} deg(i'')y_{i''}^{p*}$;
15 $\quad\quad\quad\quad \mathcal{F} \leftarrow \mathcal{F} \cup i^*$;
16 $\quad\quad$ **while** add;
$\quad\quad$ /* add the resulting inequality induced by $\mathcal{F}$ if it is violated */
17 $\quad\quad$ **if** $\sum_{i'' \in \mathcal{F}} y_{i''}^{p*} > w_p^*$ **then**
18 $\quad\quad\quad ViolIneqs \leftarrow ViolIneqs \cup \sum_{i'' \in \mathcal{F}} y_{i''}^{p*} \leq w_p$;
$\quad\quad\quad$ /* to speed-up the separation and generate a set of diverse cuts */
19 $\quad\quad\quad$ remove all edges induced by $\mathcal{F}$ from E';
20 **return** $ViolIneqs$;

Algorithm 1: Separation heuristic for inequalities (E-CLQ)

In some of our tested settings, we initialize the model with a subset of the inequalities of family (E-CLQ). This subset is constructed by a modified version of the separation heuristic. In this modified version, we use $deg(i) + deg(i')$ in line 5 and $deg(i'')$ in line 14 as of course there are no LP-values available at initialization.

Separation of Lifted Inequalities (C-CC-FIXA), (C-CC-OPT), (C-CC-FD), (C-CC-CUST), (E-CC-CLQ), (E-CC-FD)

We do not explicitly separate the lifted inequalities, but instead try to lift inequalities (C-CC) (resp., (E-CC)) when they are found by the separation routine. As discussed in the previous paragraph, inequalities (C-CC) can only be violated by fractional solutions. Thus, when only using them, it would be enough to use the `UserCutCallback` of CPLEX. However, in their lifted versions, these inequalities may cut off integer solutions, as some of the liftings are optimality-based. Thus, for settings using the liftings of (C-CC), we install an empty `LazyConstraintCallback` so that CPLEX deactivates its nonlinear reductions and dual presolve reductions.

For lifting a violated inequality (C-CC) given by some $S \subseteq F$ and $i \in F$, we proceed with the following steps. We only move to the next step, if none of the previous steps was successful. A lifting is deemed successful, if the (LP-)value of the variables on right-hand-side is larger than y_i^*.

1. Try to lift it to an inequality (C-CC-FIXA). This means, we have to find a set $\mathcal{F} \subseteq S$ fulfilling the same conditions as in the separation of inequalities (E-CLQ). We use a modified version of the heuristic presented for separating (E-CLQ) (on the subset S) for finding $\mathcal{F}$. In this modified version, we do not construct E', but only try to grow a single set $\mathcal{F}$. This set is initialized with i, and vertex $i' = \underset{i'' \in F \cap S : d_{0i} + d_{ii''} + d_{i''0}}{\arg\max} deg(i'') y^*_{i''}$. If no i' exists (due to $i'' \in F \cap S : d_{0i} + d_{ii''} + d_{i''0}$ being an empty set), we do not proceed with the lifting, and go to the next step.
2. Try to lift it to an inequality (C-CC-OPT). This is done by checking, if $Z(S) \leq LB$ for the detected set S and the current incumbent objective value LB.
3. Try to lift it to an inequality (C-CC-FD). This is done by checking all candidates fulfilling the facility dominance condition. If there is more than one candidate, we take the one with largest LP-value, ties are broken by taking the one with smallest index.
4. Try to lift it to an inequality (C-CC-CUST), if there are any $j \in C$ with $F(j) \subseteq S$. If yes, let j^* be the customer with maximum z^*_j-value among the customers fulfilling the condition, ties are broken by taking the one with smallest index. If $z^*_{j^*} > y^*_i$, we do the lifting.

In case of a violated inequality (E-CC), we proceed with the following steps. Again, we only move to the next step, if none of the previous steps was successful.

1. Try to lift it to an inequality (E-CC-CLQ). This is done in a similar way as the lifting of (C-CC) to (C-CC-FIXA).
2. Try to lift it to an inequality (E-CC-FD). This is done in a similar way as the lifting of (C-CC) to (C-CC-FD).

Additional details of the separation routine In order to avoid spending too much time in the separation routines, we limit the number of rounds of the separation-loop to 20 in the root node of the branch-and-cut, and to five in all other nodes. Moreover, when using formulation (C) we initialize our model with $x_{ii'} + x_{i'i} \leq y_i$ for each $i \in F$ and the five nearest $i' \in F$ to i, these constraints are a special case of (C-CC) for $|S| = 2$. The corresponding inequalities in case of formulation (E) are $x^v_{ii'} + x^v_{i'i} \leq y^v_i$. These inequalities forbid subtours of size two. Note that they do not involve the central depot 0 as one of the vertices, as a tour going to a single facility is feasible, and would be forbidden by a constraint $x_{0i} + x_{i0} \leq y_i$.

3.2 Branching priorities

Due to the structure of our models, branching on different set of variables will have different impacts on the structure of the solutions obtainable in the nodes of the branch-and-bound/branch-and-cut tree. CPLEX, which is the branch-and-cut solver we use, allows to give branching priorities to variables. In our implementation, we give the highest branching priorities to the facility variables y (resp., y^v), as for fixed facility variables, the customers covered in the solution can be found by inspection, and fixed facility variables have also implications on the arcs in the solution.

3.3 Primal heuristics

We implemented primal heuristics to construct feasible solutions guided by the LP-solutions for (C) and (E) using the `HeuristicCallback` of CPLEX. The heuristics consist of a construction phase, which is slightly different for (C) and (E), and an improvement phase, which does not use the LP-solutions, and thus is the same for both underlying models.

Construction heuristic guided by the LP-solution of (C) The heuristic iteratively constructs k cycles in a greedy fashion using a modified distance function $\bar{d}_{ii'} = d_{ii'}(1 - x^*_{ii'} + \epsilon)$, where x^* is the LP-solution and $\epsilon = 10^{-5}$. We refer to a (partial) solution $\mathcal{S}$ as $(\mathcal{T}, \mathcal{F}, \mathcal{C})$, where $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$ is the set of tours of the solution, $\mathcal{F}$ is the set of facilities visited in the solution, and $\mathcal{C}$ is the set of customers covered. Tours will be indicated by the ordered set of facilities (and central depot 0) visited in the tour. For a facility i and a subset $F' \subset F$ of facilities, let $\mathcal{C}(i, F')$ be the set of customers covered by i, but not by any facility in F'. The heuristic is described in Algorithm 2.

Input: instance $(V = (0, F, C), A, d, k, L)$ and LP-solution values x^*
Output: solution $(\mathcal{T}, \mathcal{F}, \mathcal{C})$
$(\mathcal{T}, \mathcal{F}, \mathcal{C}) \leftarrow (\emptyset, \emptyset, \emptyset)$;
for $(i, i') \in A_{0F}$ **do**
  $\bar{d}_{ii'} = d_{ii'}(1 - x^*_{ii'} + \epsilon)$;
for $p \in 1, \ldots, k$ **do**
  $distance_p = 0$;
  $\mathcal{T}_p \leftarrow 0$;
  /* store current vertex in the tour for later use */
  $i^* \leftarrow 0$;
  $add \leftarrow false$;
  /* try to add facilities to the current tour as long as the distance limit allows it */
  do
    $add \leftarrow false$;
    /* collect all facilities, which cover at least one uncovered customer */
    $candidateF \leftarrow i \in F \setminus \mathcal{F} : |\mathcal{C}(i, \mathcal{F})| \geq 1$;
    /* only keep facilities, which can be added to the current tour due to the distance limit */
    $candidateF \leftarrow i \in candidateF : d_{i^*i} + d_{i0} + distance(\mathcal{T}_p) \leq L$;
    if $|candidateF| > 0$ **then**
      $add \leftarrow true$;
      /* choose the candidate facility with lowest modified distance */
      $i^* \leftarrow \arg\min_{i \in candidateF} \bar{d}_{i^*i}$;
      /* update the covered customers */
      $\mathcal{C} \leftarrow \mathcal{C} \cup j : (i^*, j) \in A_{FC}$;
      /* update the set of visited facilities */
      $\mathcal{F} \leftarrow \mathcal{F} \cup i^*$;
      /* add the facility at the end of the current tour */
      $\mathcal{T}_p \leftarrow T_p + i^*$;
  while add;
  /* go back to the central depot to end the tour */
  $\mathcal{T}_p \leftarrow T_p + 0$;
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_p$;
return $(\mathcal{T}, \mathcal{F}, \mathcal{C})$;

Algorithm 2: Construction heuristics based on LP-solution values for (C)

Construction Heuristic Guided by the LP-solution of (E) The construction heuristic in this case is similar to the one used with (C); the only difference is the modified distance function $\bar{d}$. As we have LP-values (x^{p*}) for each $p \in P$, we use a modified distance function $\bar{d}^p$ for each vehicle p where $\bar{d}^p_{ii'} = d_{ii'}(1 - x^{p*}_{ii'} + \epsilon)$ for $\epsilon = 10^{-5}$.

Improvement heuristic The improvement heuristic tries to improve a constructed solution $(\mathcal{T}, \mathcal{F}, \mathcal{C})$ by inserting facilities covering customers, which are not yet covered by facilities in the solution. The heuristic is detailed in Algorithm 3.

Input: instance $(V = (0, F, C), A, d, k, L)$ and solution $(\mathcal{T}, \mathcal{F}, \mathcal{C})$
Output: potentially improved solution $(\mathcal{T}, \mathcal{F}, \mathcal{C})$

```
1  improved ← false;
2  do
3      improved ← false;
4      F ← removeDominatedFacilities(F);
5      for p ∈ 1, ..., k do
6          T_p ← removeDominatedFacilitiesFromTour(T_p, F);
       /* collect all facilities, which cover at least one uncovered customer */
7      candidateF ← i ∈ F \ F : |C(i, F)| ≥ 1;
8      sortedCandidateF ← candidateF sorted by |C(i, F)|;
       /* set up variables to store the improving facility to add, and the tour and position to add it
          to; MAX denotes a large value ≥ L */
9      i* ← null;
10     bestPos ← null;
11     bestInsertionCost ← MAX;
12     selectedTour ← null;
13     for i ∈ sortedCandidateF do
14         for p ∈ 1, ..., k do
               /* method findBestPosToInsert(T_p, i, d, L) returns the position (bestPosInTour) giving the
                  smallest increase in tour length (facilityInsertionCost) when inserting facility i in
                  tour T_p. If the distance limit L does not allow insertion of i in tour T_p, the method
                  returns MAX as facilityInsertionCost */
15             (bestPosInTour, facilityInsertionCost) ← findBestPosToInsert(T_p, i, d, L);
               /* check, if we have found an improving insertion */
16             if facilityInsertionCost < bestInsertionCost then
                   /* update the necessary data */
17                 i* ← i;
18                 bestPos ← bestPosInTour;
19                 bestInsertionCost ← facilityInsertionCost;
20                 selectedTour ← p;
           /* we have found a facility providing an improving insertion, insert it, and start next
              iteration of the improvement heuristic */
21         if i* ≠ null then
22             improved ← true;
               /* update the tour by inserting i* at the best position */
23             T_selectedTour ← insertFacilityInTourAtPos(i*, T_selectedTour, bestPos);
               /* update the covered customers */
24             C ← C ∪ j : (i*, j) ∈ A_FC;
               /* update the set of visited facilities */
25             F ← F ∪ i*;
26             break;
27 while improved;
28 return (T, F, C);
```

Algorithm 3: Improvement heuristic

First, all facilities, which are dominated by another facility in the solution are removed, and the tours are updated. Let *candidateF* be a list of all facilities not in the current solution, and which cover at least one customer not covered by the current solution. We sort the facilities in *candidateF* in descending order based on the number of uncovered customers they would cover. This sorted list is denoted as *sortedCandidateF*. We then iterate through facilities $i \in$ *sortedCandidateF* and try insertion of i in each tour $\mathcal{T}_1, \ldots, \mathcal{T}_k$ at each possible position. Let i^* be the first facility on the list *sortedCandidateF*, which can be inserted in a tour. We insert i^* at the position and tour, which leads to the smallest increase in tour length. Then, the whole procedure is restarted from the beginning, as adding i^* to the solution may lead to some new facilities in

the solution being dominated. The algorithm terminates when there is no more improving facility i^* found.

4 Computational results

The runs were carried out on an Intel Xeon E5-2670v2 machine with 2.5 GHz and 3GB of memory using a single thread, and all CPLEX parameters (except branching priorities) are left at their default values.

4.1 Comparison with the MIP Approach of Amiri and Salari (2019)

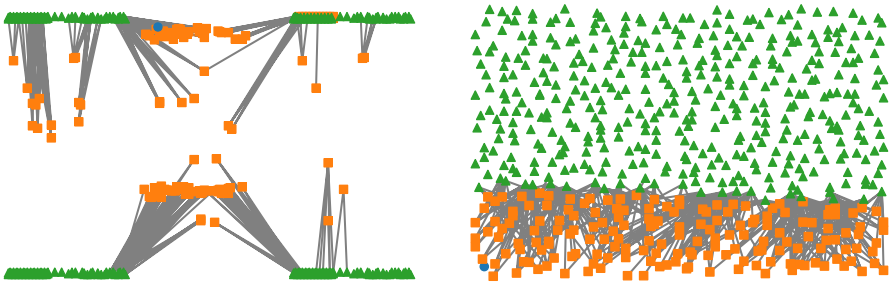
In this section, we compare our approaches with the MIP approach presented in Amiri and Salari (2019) using the instances introduced in the same paper. We obtained these instances on request from the authors and made them available at https://msinnl.github.io/pages/instancescodes.html. These instances are denoted as `AMSAL` in the following. For the runs in this section, we used a timelimit of 600 seconds.

The instances are based on TSPLIB-instances with 52, 76, 100, 150, 200, 318, 417, 575, 657 and 724 vertices. For each underlying TSPLIB-instance, three different instances were created by taking 50%, 60% and 70% of the vertices as customers, and the remaining vertices except one as facilities and one vertex as central depot. For each facility, the customers it can cover are randomly chosen from the five nearest ones. Here, we discovered an issue with the instances, namely there are often some customers which cannot be covered by any facility, and thus are useless (this can affect up to 90% of the customers of an instance in some cases). For most of the instances, this even led to the situation that all customers which could be covered by some facility in the instance were in the optimal solution (especially after taking account also customers unreachable due to the distance limit as described in Theorem 2). Figure 2 depicts two instances to illustrate this issue (we refer to instances by `|F| − |C| − k − L`). The issue is extremely pronounced, as the facility–customer-split is not chosen randomly among the vertices of the underlying TSPLIB-instance, but the first 50% (resp., 40%, 30%) of vertices in the TSPLIB-instance are taken as facilities. From the figures, it can be seen that these vertices are clustered by location.

Three different values for the number of vehicles (k) are considered, namely $k = 2, 3, 4$. In Amiri and Salari (2019), the following formula, for $\alpha \in \{1, 0.9, 0.8\}$ is given to define the value for L:

$$L = \frac{\alpha}{k} \cdot |F| \cdot \frac{\sum_{i \in F \cup 0} \sum_{i' \in F \cup 0} d_{ii'}}{(|F| \cdot (|F| + 1))/2}$$

However, when verifying this formula, we noticed that it does not give the correct value, which could be obtained with the following formula instead:



(a) Instance `125-291-2-9372.69`. **(b)** Instance `172-402-2-1732.68`.

Fig. 2 Two instance graphs of the TCMCRP from the set `AMSAL` from Amiri and Salari (2019). The blue circle is the central depot, orange boxes are facilities, and green triangles are customers. The gray edges between facilities and customers denote which customers are covered by each facility

$$L = \cdot\frac{\alpha}{k} \cdot |F| \cdot \frac{\sum_{i\in F\cup 0}\sum_{i'\in F\cup 0} d_{ii'}}{(|F| \cdot (|F|+1))/4}.$$

The formula could be verified, as the respective values of L are also written explicitly in the obtained instance files and in the result tables in Amiri and Salari (2019). However, in some of the instance files, there were wrong values, and also some entries in the tables in Amiri and Salari (2019) have wrong entries, in particular in Table 3 containing the so-called *large* instances containing 318 vertices and more. We discuss this in more detail later. We corrected these errors in the instance files for our computational study and our uploaded instances also consist of the corrected instances. In total, this set contains $10 \cdot 3 \cdot 3 \cdot 3 = 270$ instances (ten underlying graphs, and the different parameters for $|C|$, k and L). From the discussion above, one can already see that these instances are maybe not too meaningful for benchmarking. However, as they are the only instances from literature for the problem, we still consider them, but also introduce new instances to evaluate our approaches in Sect. 4.2.

Comparison of framework ingredients First, we are interested in the effect of the different models, valid inequalities, branching priorities and the primal heuristic. We compare the following settings:

- `C`: Model (C) without any variable fixing and valid inequalities. The primal heuristic is not used and branching priorities are left at default.
- `C+`: `C` with variable fixing ((C-FDA), (C-FIXA),(C-FIXC),(C-FIXF)) and valid inequalities (C-FD), (C-DISTG), (C-FLOWER) added at initialization. Note that the resulting model is still compact.
- `C++`: `C+` with the separation of connectivity cuts (C-CC) (and the liftings (C-CC-OPT), (C-CC-FD), (C-CC-CUST), (C-CC-FIXA)) and also branching priorities as described in Sect. 3.2.
- `C++H`: `C++` with the primal heuristic as described in Sect. 3.3.

- C++HS: C++H, where some of the polynomial-sized families of inequalities are separated by enumeration instead of adding it at initialization. The separated inequalities are (C-FD), (C-DIST) and (C-FLOWER). Moreover, explicit upper bounds $f_{ii'} \leq max(L - d_{i'0}, 0)$ are added at initialization to provide an initial upper bound on the $f_{ii'}$-variables, as constraints (C-DIST), which upper-bound $f_{ii'}$ in connection with $x_{ii'}$, are now separated.
- E: Model (E) without any variable fixing and valid inequalities and lifting. The primal heuristic is not used and branching priorities are left at default.
- E+: Setting E with variable fixing ((E-FDA),(E-FIXA),(E-FIXC),(E-FIXF)) and symmetry breaking inequalities (E-SYM) added at initialization. Valid inequalities (E-FD) are separated by enumeration. Moreover, a subset of inequalities (E-CLQ) is also added at initialization as described in Sect. 3.1.
- E++: Model (E) with variable fixing ((E-FDA),(E-FIXA),(E-FIXC),(E-FIXF)), and symmetry breaking inequalities (E-SYM) added at initialization. Valid inequalities (E-FD) are separated by enumeration. Inequalities (E-CLQ) are separated, and liftings ((E-CC-CLQ) and (E-CC-FD)) of the connectivity cuts (E-CC) are used. Moreover, branching priorities as described in Sect. 3.2 are also used.
- E++H: E++ with the primal heuristic as described in Sect. 3.3.
- E++H2: E++H, where inequalities (E-CLQ) are not only separated, but a subset of them is also added at initialization as described in Sect. 3.1.

Figures 3 and 4 give plots of the runtime to optimality for the instances from set AMSAL for these settings.

Looking at Figs. 3 and 4, the approaches based on model (E) seem more promising, as all settings except E manage to solve nearly all of the instances to optimality within the given timelimit (E++H2 and E++H solver more than 85% of the instances within 25 seconds). The situation for settings based on (C) is more diverse, C++HS and C++H also manage to solve nearly all of the instances to optimality, but the runtime is worse. C+ and E have nearly identical performance (solving about 75% of the instances), and both are considerably better than C (solving about 50% of the instances) and C++ (solving about 62.5% of the instances). Thus, it seems the variable fixing and the valid inequalities are quite helpful regardless of the model, while connectivity cuts (C-CC) are actually hurting the performance for the approach based on (C). Potential explanations for this could be that (i) separation takes too long (leading to a slow node-throughput in the branch-and-bound), (ii) the internal CPLEX heuristics are less effective in finding feasible solutions due to the presence of separated inequalities, (iii) the upper bounds of the LP-relaxation after adding the variable fixing and the valid inequalities are already quite good due to the particular structure of the AMSAL instances. Interestingly, settings C++HS and C++H, which also use the connectivity cuts (C-CC), work better than C+. Both these settings are based on C++, but (i) additionally use the primal heuristic (C++H), resp., ii) also separate some inequalities (C++HS) instead of initializing with them, which leads to a smaller initial model and a faster node-throughput in the branch-and-bound.

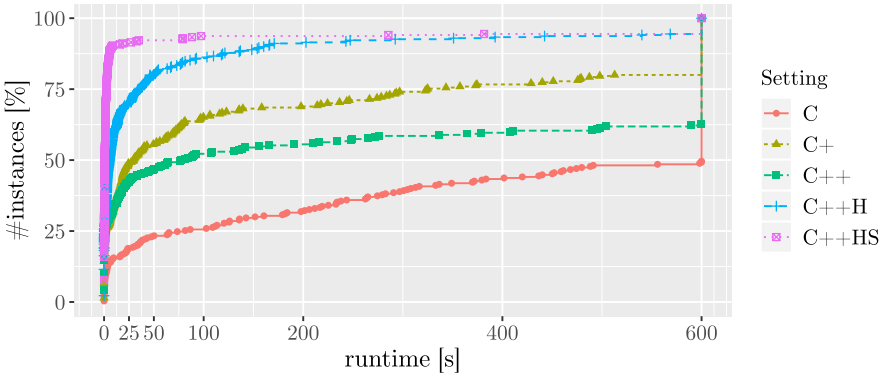


Fig. 3 Runtime to optimality for instance set AMSAL and different settings based on (*C*)

Detailed results Next, we give a detailed overview of the results for the instance AMSAL and settings C+, C++HS, and E++H2. Settings C++HS, and E++H2 are the best settings based on (*C*) and (*E*), respectively. We also include C+ in the tables as the model used in this setting is compact and the setting uses no heuristic. Thus, it can be given directly to a MIP-solver without the need of implementing separation procedures or heuristics. This could make it an attractive choice for a practitioner. We also compare our results with the ones reported by the MIP approach of Amiri and Salari (2019). The results in Amiri and Salari (2019) were obtained using CPLEX 12.3 on an Intel Core i7 2.93 GHz processor with 3.49 GB of RAM. Tables 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 gives the results for these instances. In the tables, we report for each instance the number of vertices ($|V|$), the number of facilities ($|F|$), the number of customers ($|C|$), the number of vehicles (k), the distance limit (L), the number of customers covered by at least one facility which is reachable from the central depot ($|C_R|$ "reachable customers," i.e., following Theorem 2). For each setting, we report the runtime ($t[s]$), value of the best solution found (z^*), optimality gap ($g[\%]$, calculated as $100 \cdot ((UB - z^*)/z^*)$, where UB is the value of the upper bound found by the setting), and number of branch-and-bound nodes (*#nBBn*; only for our settings, as these are not reported in Amiri and Salari (2019)). A **bold** entry in z^* indicates that optimality has be proven (as can also be seen by the optimality gap $g[\%]$ being zero). An entry of "-" in z^* and $g[\%]$ indicates that no feasible solution could be found within the timelimit. A **bold** entry in $|C_R|$ indicates, that the optimal solution consists of all reachable customers. An *italic* entry in the column z^* for the results of Amiri and Salari (2019) indicates that the reported solution value has some issues, which are discussed in detail in the following paragraph.

Note that for instances with up to 200 vertices, the facility–customer coverage allocation is the same for each underlying graph and facility/customer split. This means that for the same `|F|-|C|-p`, the optimal value of instances with larger L always gives an upper bound to the optimal value of instances with smaller L. The following instances files contained wrong values for L, and accordingly, there were also wrong results reported for these instances in Amiri and Salari (2019): instance `15-36-2-1090.05` was the same as instance

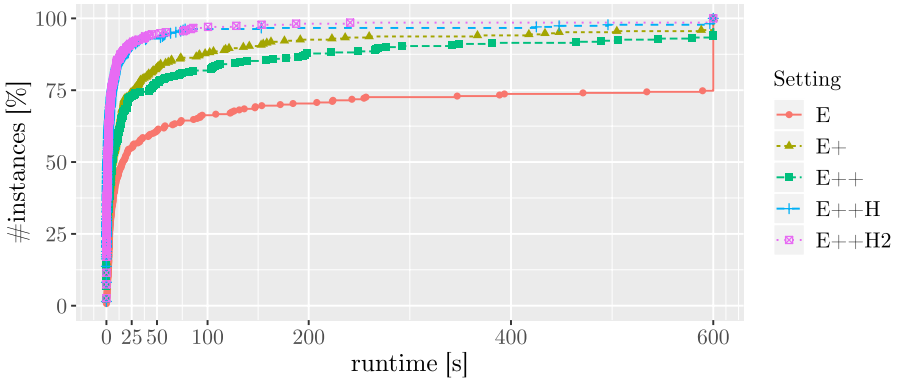


Fig. 4 Runtime to optimality for instance set AMSAL and different settings based on (*E*)

15-36-2-1368.07, and instance 59-140-4-5005.31 was the same as instance 59-140-4-5630.98. With regard to instance 15-36-2-1368.07, the value in the instance file and in the result table in Amiri and Salari (2019) is also not the same, it is 1386.07 versus 1386.03, with the former being the correct value when verified using the formula mentioned in the beginning of the section. For 15-36-2-1090.05, the value of L in the table is also not the same as the one obtained by calculation with the formula, the correct value is 1094.46. Moreover, for instances 29-70-2-5574.4, 29-70-2-4955.06, 29-70-3-4129.21, 29-70-3-3716.29 the table in Amiri and Salari (2019) reports an optimal value of 36, while we found different (lower) optimal values. The (optimal) results for instances 25-26-4-894.71 and 25-26-4-795.30 also seem wrong compared to our results, and also conflict with the construction of the instances, as the optimal solution value reported for the instance with lower distance limit is larger than the one for the instance with larger distance limit Finally, in Table 3 in Amiri and Salari (2019) (containing the instances with 318 vertices and more), there are many wrong combinations of |F|-|C|-p-L (i.e., not as occurring in the instance files). Moreover, $L = 1061.21$ in the table should read 51061.21, and $L = 7243.53$ is occurring twice in the table, while 6699.96 is occurring in an instance file, but missing in the table. As the values of L are unique in the instances, the result of Table 3 could still be useful for comparison by assuming that the best objective value reported for a given L gives a correct value and only $|F|$, $|C|$ and k was mixed up in the table. However, when we compared the reported solution values for a given L with our obtained solution values for the instance with this L, it was often larger than the optimal value we found and even larger than the number of true customers of the instance. Thus, it is not clear to us how to directly compare the results, resp., which of the solution values in Table 3 of Amiri and Salari (2019) correspond to which instance. Hence we do not report the detailed results of Amiri and Salari (2019) for instances of AMSAL with 318 vertices or more, but just note, that for only 38 of these 135 instances, the MIP approach of Amiri and Salari (2019) managed to find a feasible solution within the given timelimit of 18000 seconds, and for 32 of them, optimality is

Table 1 Detailed results for instance set AMSAL with 52 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
15	36	2	1368.07	19	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00
15	36	2	1231.26	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
15	36	2	1094.46	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	*15*	0.00
15	36	3	912.05	12	0	**11**	0.00	0	0	**11**	0.00	0	0	**11**	0.00	0	0	**11**	0.00
15	36	3	820.84	**9**	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00
15	36	3	729.64	**8**	0	**8**	0.00	0	0	**8**	0.00	0	0	**8**	0.00	0	0	**8**	0.00
15	36	4	684.04	**8**	0	**8**	0.00	0	0	**8**	0.00	0	0	**8**	0.00	0	0	**8**	0.00
15	36	4	615.63	**8**	0	**8**	0.00	0	0	**8**	0.00	0	0	**8**	0.00	0	0	**8**	0.00
15	36	4	547.23	**1**	0	**1**	0.00	0	0	**1**	0.00	0	0	**1**	0.00	0	0	**1**	0.00
20	31	2	1707.37	17	0	**15**	0.00	50	0	**15**	0.00	9	0	**15**	0.00	0	4	**15**	0.00
20	31	2	1536.63	17	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00	0	1	**15**	0.00
20	31	2	1365.89	17	0	**14**	0.00	0	0	**14**	0.00	3	0	**14**	0.00	6	1	**14**	0.00
20	31	3	1138.24	**13**	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00
20	31	3	1024.42	**13**	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00
20	31	3	910.59	13	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	1	**12**	0.00
20	31	4	853.68	**10**	0	**10**	0.00	0	0	**10**	0.00	0	0	**10**	0.00	0	0	**10**	0.00
20	31	4	768.31	**9**	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00
20	31	4	682.95	**9**	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00
25	26	2	1988.24	21	0	**20**	0.00	0	0	**20**	0.00	0	0	**20**	0.00	0	30	**20**	0.00
25	26	2	1789.42	20	0	**19**	0.00	0	0	**19**	0.00	0	0	**19**	0.00	0	20	**19**	0.00
25	26	2	1590.59	20	0	**18**	0.00	20	0	**18**	0.00	0	0	**18**	0.00	0	3	**18**	0.00
25	26	3	1325.49	18	0	**17**	0.00	0	0	**17**	0.00	5	0	**17**	0.00	0	57	**17**	0.00

Table 1 (continued)

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
25	26	3	1192.94	**15**	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00	0	2	**15**	0.00
25	26	3	1060.39	**15**	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00
25	26	4	994.12	**15**	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00	0	1	*14*	0.00
25	26	4	894.71	15	0	**14**	0.00	0	0	**14**	0.00	0	0	**14**	0.00	0	22	*11*	0.00
25	26	4	795.30	**11**	0	**11**	0.00	0	0	**11**	0.00	0	0	**11**	0.00	0	0	*12*	0.00

Table 2 Detailed results for instance set AMSAL with 76 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
22	53	2	11960.48	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
22	53	2	10764.43	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
22	53	2	9568.38	**10**	0	**10**	0.00	0	0	**10**	0.00	0	0	**10**	0.00	0	0	**10**	0.00
22	53	3	7973.65	**10**	0	**10**	0.00	0	0	**10**	0.00	0	0	**10**	0.00	0	0	**10**	0.00
22	53	3	7176.29	**7**	0	**7**	0.00	0	0	**7**	0.00	0	0	**7**	0.00	0	0	**7**	0.00
22	53	3	6378.92	**7**	0	**7**	0.00	0	0	**7**	0.00	0	0	**7**	0.00	0	0	**7**	0.00
22	53	4	5980.24	**5**	0	**5**	0.00	0	0	**5**	0.00	0	0	**5**	0.00	0	0	**5**	0.00
22	53	4	5382.22	**5**	0	**5**	0.00	0	0	**5**	0.00	0	0	**5**	0.00	0	0	**5**	0.00
22	53	4	4784.19	**5**	0	**5**	0.00	0	0	**5**	0.00	0	0	**5**	0.00	0	0	**5**	0.00
30	45	2	17564.36	**16**	0	**16**	0.00	0	0	**16**	0.00	0	0	**16**	0.00	0	3	**16**	0.00
30	45	2	15807.92	**16**	0	**16**	0.00	0	0	**16**	0.00	0	0	**16**	0.00	0	1	**16**	0.00
30	45	2	14051.49	14	0	**13**	0.00	25	0	**13**	0.00	0	0	**13**	0.00	0	14	**13**	0.00
30	45	3	11709.57	**13**	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00
30	45	3	10538.61	**13**	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00	0	0	**13**	0.00
30	45	3	9367.66	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
30	45	4	8782.18	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
30	45	4	7903.96	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
30	45	4	7025.74	**7**	0	**7**	0.00	0	0	**7**	0.00	0	0	**7**	0.00	0	0	**7**	0.00
37	38	2	22400.71	**17**	0	**17**	0.00	0	0	**17**	0.00	0	0	**17**	0.00	0	5	**17**	0.00
37	38	2	20160.64	**17**	0	**17**	0.00	0	0	**17**	0.00	0	0	**17**	0.00	0	2	**17**	0.00
37	38	2	17920.57	**15**	0	**15**	0.00	0	0	**15**	0.00	0	0	**15**	0.00	0	3	**15**	0.00
37	38	3	14933.80	**14**	0	**14**	0.00	0	0	**14**	0.00	0	0	**14**	0.00	0	1	**14**	0.00

Table 2 (continued)

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	*#BB*	$t[s]$	z^*	$g[\%]$	*#BB*	$t[s]$	z^*	$g[\%]$	*#BB*	$t[s]$	z^*	$g[\%]$
37	38	3	13440.42	**14**	0	**14**	0.00	0	0	**14**	0.00	0	0	**14**	0.00	0	0	**14**	0.00
37	38	3	11947.04	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
37	38	4	11200.35	**12**	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00	0	0	**12**	0.00
37	38	4	10080.32	**11**	0	**11**	0.00	0	0	**11**	0.00	0	0	**11**	0.00	0	0	**11**	0.00
37	38	4	8960.28	**9**	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00	0	0	**9**	0.00

Table 3 Detailed results for instance set AMSAL with 100 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
29	70	2	6193.82	43	74	**36**	0.00	8865	382	**36**	0.00	26707	4	**36**	0.00	103	350	**36**	0.00
29	70	2	5574.44	40	1	**34**	0.00	69	1	**34**	0.00	40	0	**34**	0.00	0	6	*36*	0.00
29	70	2	4955.06	35	0	**30**	0.00	10	0	**30**	0.00	0	0	**30**	0.00	3	3	*36*	0.00
29	70	3	4129.21	**27**	0	**27**	0.00	0	0	**27**	0.00	0	0	**27**	0.00	0	0	*36*	0.00
29	70	3	3716.29	**24**	0	**24**	0.00	0	0	**24**	0.00	0	0	**24**	0.00	0	0	*36*	0.00
29	70	3	3303.37	**21**	0	**21**	0.00	0	0	**21**	0.00	0	0	**21**	0.00	0	0	**21**	0.00
29	70	4	3096.91	**21**	0	**21**	0.00	0	0	**21**	0.00	0	0	**21**	0.00	0	0	**21**	0.00
29	70	4	2787.22	**16**	0	**16**	0.00	0	0	**16**	0.00	0	0	**16**	0.00	0	0	**16**	0.00
29	70	4	2477.53	**14**	0	**14**	0.00	0	0	**14**	0.00	0	0	**14**	0.00	0	0	**14**	0.00
39	60	2	8326.08	**51**	10	**51**	0.00	250	0	**51**	0.00	0	0	**51**	0.00	0	92	**51**	0.00
39	60	2	7493.47	**51**	97	**51**	0.00	4530	5	**51**	0.00	12	1	**51**	0.00	0	2	**51**	0.00
39	60	2	6660.87	51	278	**49**	0.00	8130	600	49	4.08	10115	4	**49**	0.00	23	10564	**49**	0.00
39	60	3	5550.72	49	600	48	2.08	55152	600	48	2.08	26586	11	**48**	0.00	76	18000	48	2.08
39	60	3	4995.65	47	600	40	5.00	32010	600	40	5.00	26343	14	**40**	0.00	304	5434	**40**	0.00
39	60	3	4440.58	39	34	**38**	0.00	4098	88	**38**	0.00	7309	2	**38**	0.00	57	314	**38**	0.00
39	60	4	4163.04	**34**	0	**34**	0.00	27	0	**34**	0.00	0	0	**34**	0.00	0	10	**34**	0.00
39	60	4	3746.74	**32**	0	**32**	0.00	0	0	**32**	0.00	0	0	**32**	0.00	0	1	**32**	0.00
39	60	4	3330.43	**27**	0	**27**	0.00	0	0	**27**	0.00	0	0	**27**	0.00	0	2	**27**	0.00
49	50	2	10458.99	**45**	2	**45**	0.00	0	0	**45**	0.00	0	0	**45**	0.00	0	8	**45**	0.00
49	50	2	9413.09	**45**	39	**45**	0.00	610	0	**45**	0.00	0	0	**45**	0.00	0	27	**45**	0.00
49	50	2	8367.19	**45**	3	**45**	0.00	0	6	**45**	0.00	3	0	**45**	0.00	0	37	**45**	0.00
49	50	3	6972.66	**45**	24	**45**	0.00	180	1	**45**	0.00	0	0	**45**	0.00	0	1018	**45**	0.00

Table 3 (continued)

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
49	50	3	6275.93	**45**	97	**45**	0.00	1974	4	**45**	0.00	2	1	**45**	0.00	0	1175	**45**	0.00
49	50	3	5578.13	43	600	42	2.38	15598	600	42	2.38	13620	14	**42**	0.00	70	18000	42	2.38
49	50	4	5229.49	43	600	42	2.38	16707	600	42	2.38	17807	38	**42**	0.00	191	18000	42	2.38
49	50	4	4706.55	**37**	1	**37**	0.00	0	0	**37**	0.00	0	0	**37**	0.00	0	127	**37**	0.00
49	50	4	4183.60	**33**	0	**33**	0.00	0	0	**33**	0.00	0	0	**33**	0.00	0	12	**33**	0.00

Table 4 Detailed results for instance set AMSAL with 150 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
44	105	2	9210.94	**72**	40	**72**	0.00	1502	8	**72**	0.00	42	0	**72**	0.00	0	454	**72**	0.00
44	105	2	8289.84	**72**	324	**72**	0.00	5585	2	**72**	0.00	0	0	**72**	0.00	0	1729	**72**	0.00
44	105	2	7368.75	72	600	69	4.35	13506	600	70	2.86	8063	45	**70**	0.00	247	18000	70	2.86
44	105	3	6140.62	**72**	476	**72**	0.00	8927	35	**72**	0.00	189	13	**72**	0.00	25	18000	69	4.35
44	105	3	5526.56	72	2	**69**	0.00	0	2	**69**	0.00	0	4	**69**	0.00	1	56	**69**	0.00
44	105	3	4912.50	68	600	64	4.56	21031	600	64	4.69	12997	84	**64**	0.00	501	18000	64	1.56
44	105	4	4605.47	67	600	66	1.52	32037	600	66	1.52	20136	42	**66**	0.00	29	18000	66	1.52
44	105	4	4144.92	64	600	58	2.19	39896	600	58	6.90	15176	153	**58**	0.00	913	18000	58	2.58
44	105	4	3684.37	54	158	**49**	0.00	12351	600	49	2.04	25698	76	**49**	0.00	546	1555	**49**	0.00
59	90	2	12364.81	**71**	72	**71**	0.00	486	0	**71**	0.00	0	0	**71**	0.00	0	45	**71**	0.00
59	90	2	11128.36	**71**	323	**71**	0.00	2598	0	**71**	0.00	0	0	**71**	0.00	0	614	**71**	0.00
59	90	2	9891.85	**71**	96	**71**	0.00	486	0	**71**	0.00	0	0	**71**	0.00	0	2193	**71**	0.00
59	90	3	8243.20	**71**	43	**71**	0.00	400	0	**71**	0.00	0	0	**71**	0.00	0	18000	70	1.43
59	90	3	7418.88	**71**	6	**71**	0.00	0	1	**71**	0.00	0	2	**71**	0.00	0	6268	**71**	0.00
59	90	3	6594.56	**71**	135	**71**	0.00	602	16	**71**	0.00	22	4	**71**	0.00	0	13768	69	2.90
59	90	4	6182.40	**71**	123	**71**	0.00	1213	1	**71**	0.00	0	9	**71**	0.00	0	2920	**71**	0.00
59	90	4	5564.16	**71**	287	**71**	0.00	1597	19	**71**	0.00	31	100	**71**	0.00	44	5404	**71**	0.00
59	90	4	4945.92	68	600	65	3.08	7931	600	65	3.08	6202	600	65	3.08	591	18000	65	4.43
74	75	2	15315.62	**64**	40	**64**	0.00	200	0	**64**	0.00	0	0	**64**	0.00	0	32	**64**	0.00
74	75	2	13784.06	**64**	106	**64**	0.00	340	0	**64**	0.00	0	0	**64**	0.00	0	24	**64**	0.00
74	75	2	12252.49	**64**	5	**64**	0.00	0	0	**64**	0.00	0	0	**64**	0.00	0	174	**64**	0.00
74	75	3	10210.41	**64**	478	**64**	0.00	2588	0	**64**	0.00	0	0	**64**	0.00	0	176	**64**	0.00
74	75	3	9189.37	**64**	236	**64**	0.00	1008	0	**64**	0.00	0	0	**64**	0.00	0	279	**64**	0.00

Table 4 (continued)

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
74	75	3	8168.33	**64**	218	**64**	0.00	589	0	**64**	0.00	0	0	**64**	0.00	0	18000	63	1.59
74	75	4	7657.81	**64**	12	**64**	0.00	0	0	**64**	0.00	0	1	**64**	0.00	0	3799	**64**	0.00
74	75	4	6892.03	**64**	18	**64**	0.00	0	0	**64**	0.00	0	1	**64**	0.00	0	417913	**64**	0.00
74	75	4	6126.25	**64**	27	**64**	0.00	0	1	**64**	0.00	0	10	**64**	0.00	0	7620	**64**	0.00

Table 5 Detailed results for instance set AMSAL with 200 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
59	140	2	12513.29	**104**	6	**104**	0.00	0	1	**104**	0.00	0	1	**104**	0.00	0	89	**104**	0.00
59	140	2	11261.96	**104**	274	**104**	0.00	1624	33	**104**	0.00	25	9	**104**	0.00	3	173	**104**	0.00
59	140	2	10010.63	**104**	600	101	2.97	3178	600	103	0.97	3997	20	**104**	0.00	14	18000	102	1.96
59	140	3	8342.19	**104**	498	**104**	0.00	3771	27	**104**	0.00	10	86	**104**	0.00	22	18000	92	13.04
59	140	3	7507.97	**104**	600	103	0.97	2517	79	**104**	0.00	57	78	**104**	0.00	19	10308	**104**	0.00
59	140	3	6673.75	103	600	101	1.98	1973	600	101	1.98	1066	600	101	1.98	152	18000	101	1.98
59	140	4	6256.64	103	600	97	5.15	1832	600	101	0.99	1558	600	97	5.15	46	18000	100	2.74
59	140	4	5630.98	**90**	600	89	1.12	15649	600	89	1.12	3347	188	**90**	0.00	103	18000	88	2.27
59	140	4	5005.31	87	1	**83**	0.00	0	9	**83**	0.00	5	241	**83**	0.00	100	18000	88	2.27
79	120	2	16673.37	**107**	60	**107**	0.00	200	0	**107**	0.00	0	0	**107**	0.00	0	60	**107**	0.00
79	120	2	15006.03	**107**	81	**107**	0.00	250	0	**107**	0.00	0	0	**107**	0.00	0	89	**107**	0.00
79	120	2	13338.69	**107**	246	**107**	0.00	579	0	**107**	0.00	0	0	**107**	0.00	0	262	**107**	0.00
79	120	3	11115.58	**107**	293	**107**	0.00	580	0	**107**	0.00	0	0	**107**	0.00	0	9168	**107**	0.00
79	120	3	10004.02	**107**	479	**107**	0.00	1080	1	**107**	0.00	0	0	**107**	0.00	0	18000	103	3.88
79	120	3	8892.46	**107**	100	**107**	0.00	120	79	**107**	0.00	16	5	**107**	0.00	0	18000	100	7.00
79	120	4	8336.68	**107**	325	**107**	0.00	1030	2	**107**	0.00	0	1	**107**	0.00	0	18000	104	2.89
79	120	4	7503.01	**107**	600	99	8.08	717	97	**107**	0.00	20	128	**107**	0.00	4	18000	103	3.88
79	120	4	6669.35	**106**	600	99	7.07	1102	286	**106**	0.00	111	600	105	0.95	101	18000	95	11.58
99	100	2	20883.45	**94**	267	**94**	0.00	506	0	**94**	0.00	0	0	**94**	0.00	0	101	**94**	0.00
99	100	2	18795.10	**94**	39	**94**	0.00	30	0	**94**	0.00	0	0	**94**	0.00	0	178	**94**	0.00
99	100	2	16706.76	**94**	15	**94**	0.00	0	0	**94**	0.00	0	0	**94**	0.00	0	254	**94**	0.00
99	100	3	13922.30	**94**	600	79	18.99	506	0	**94**	0.00	0	1	**94**	0.00	0	96	**94**	0.00
99	100	3	12530.07	**94**	214	**94**	0.00	440	0	**94**	0.00	0	1	**94**	0.00	0	460	**94**	0.00

Table 5 (continued)

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2				Amiri and Salari (2019)		
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$
99	100	3	11137.84	**94**	600	85	10.59	725	0	**94**	0.00	0	1	**94**	0.00	0	646	**94**	0.00
99	100	4	10441.72	**94**	357	**94**	0.00	507	0	**94**	0.00	0	1	**94**	0.00	0	1058	**94**	0.00
99	100	4	9397.55	**94**	429	**94**	0.00	380	0	**94**	0.00	0	1	**94**	0.00	0	8875	**94**	0.00
99	100	4	8353.38	**94**	451	**94**	0.00	498	1	**94**	0.00	0	1	**94**	0.00	0	3986	**94**	0.00

Table 6 Detailed results for instance set AMSAL for instances with 318 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2			
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
95	222	2	10719.94	**43**	9	**43**	0.00	0	0	**43**	0.00	0	0	**43**	0.00	0
95	222	3	7146.63	**43**	16	**43**	0.00	0	0	**43**	0.00	0	1	**43**	0.00	0
95	222	4	5359.97	**36**	7	**36**	0.00	0	1	**36**	0.00	0	1	**36**	0.00	0
95	222	2	12059.93	**40**	11	**40**	0.00	0	0	**40**	0.00	0	0	**40**	0.00	0
95	222	3	8039.95	**39**	8	**39**	0.00	0	0	**39**	0.00	0	0	**39**	0.00	0
95	222	4	6029.97	**40**	14	**40**	0.00	0	0	**40**	0.00	0	1	**40**	0.00	0
95	222	2	13399.92	**43**	19	**43**	0.00	40	0	**43**	0.00	0	0	**43**	0.00	0
95	222	3	8933.28	**39**	4	**39**	0.00	0	0	**39**	0.00	0	0	**39**	0.00	0
95	222	4	6699.96	**41**	21	**41**	0.00	0	0	**41**	0.00	0	1	**41**	0.00	0
127	190	2	17407.31	**36**	20	**36**	0.00	0	0	**36**	0.00	0	1	**36**	0.00	0
127	190	3	11604.87	**41**	8	**41**	0.00	0	0	**41**	0.00	0	1	**41**	0.00	0
127	190	4	8703.65	**37**	5	**37**	0.00	0	0	**37**	0.00	0	1	**37**	0.00	0
127	190	2	19583.22	**39**	18	**39**	0.00	0	0	**39**	0.00	0	0	**39**	0.00	0
127	190	3	13055.48	**43**	35	**43**	0.00	11	0	**43**	0.00	0	1	**43**	0.00	0
127	190	4	9791.61	**44**	22	**44**	0.00	0	0	**44**	0.00	0	1	**44**	0.00	0
127	190	2	21759.13	**40**	13	**40**	0.00	0	0	**40**	0.00	0	0	**40**	0.00	0
127	190	3	14506.09	**40**	10	**40**	0.00	0	0	**40**	0.00	0	1	**40**	0.00	0
127	190	4	10879.57	**38**	7	**38**	0.00	0	0	**38**	0.00	0	1	**38**	0.00	0
158	159	2	22073.87	**40**	21	**40**	0.00	0	1	**40**	0.00	0	1	**40**	0.00	0
158	159	3	14715.91	**37**	80	**37**	0.00	0	1	**37**	0.00	0	2	**37**	0.00	0
158	159	4	11036.93	**37**	34	**37**	0.00	0	1	**37**	0.00	0	3	**37**	0.00	0
158	159	2	24833.10	**41**	30	**41**	0.00	0	1	**41**	0.00	0	1	**41**	0.00	0
158	159	3	16555.40	**42**	8	**42**	0.00	0	1	**42**	0.00	0	2	**42**	0.00	0
158	159	4	12416.55	**39**	30	**39**	0.00	0	1	**39**	0.00	0	2	**39**	0.00	0
158	159	2	27592.33	**40**	19	**40**	0.00	0	1	**40**	0.00	0	1	**40**	0.00	0
158	159	3	18394.89	**39**	9	**39**	0.00	0	1	**39**	0.00	0	1	**39**	0.00	0
158	159	4	13796.17	**39**	6	**39**	0.00	0	1	**39**	0.00	0	3	**39**	0.00	0

reported (and for 112 of the other instances, leading to a total number of 144 out of 270, for which optimality is reported in Amiri and Salari (2019)).

The results show that E++H2 manages to solve 266 out of 270 instances to optimality, C++HS 255 instances and C+ 216 instances. For one of the four instances not solved to optimality by E++H2, instance 200-79-120-4-6669.35, C++HS find the optimal solution. We note that for the larger instances, C+ often does not manage to leave the root node (see, e.g., Table 10) within the given timelimit, as the size of the model likely becomes prohibitive. This means CPLEX may take a very long time for solving the initial LP-relaxation and also for its internal heuristics, which rely on the LP-relaxation.

For smaller instances, the performance is quite similar to E++H2 and C++HS. Moreover, for 239 instances of the 270 instances, the value of the optimal solution

Table 7 Detailed results for instance set AMSAL for instances with 417 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2			
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
125	291	2	9372.69	**75**	600	74	1.35	540	0	**75**	0.00	0	1	**75**	0.00	0
125	291	3	6248.46	**72**	15	**72**	0.00	0	0	**72**	0.00	0	1	**72**	0.00	0
125	291	4	4686.35	**75**	198	**75**	0.00	300	0	**75**	0.00	0	2	**75**	0.00	0
125	291	2	10544.28	**73**	11	**73**	0.00	0	0	**73**	0.00	0	1	**73**	0.00	0
125	291	3	7029.52	**73**	56	**73**	0.00	50	0	**73**	0.00	0	1	**73**	0.00	0
125	291	4	5272.14	**71**	9	**71**	0.00	0	0	**71**	0.00	0	2	**71**	0.00	0
125	291	2	11715.87	**75**	12	**75**	0.00	0	0	**75**	0.00	0	1	**75**	0.00	0
125	291	3	7810.58	**72**	18	**72**	0.00	0	0	**72**	0.00	0	1	**72**	0.00	0
125	291	4	5857.93	**67**	600	21	219.05	400	0	**67**	0.00	0	2	**67**	0.00	0
166	250	2	12877.39	**63**	38	**63**	0.00	0	1	**63**	0.00	0	1	**63**	0.00	0
166	250	3	8584.93	**64**	69	**64**	0.00	0	1	**64**	0.00	0	2	**64**	0.00	0
166	250	4	6438.69	**65**	62	**65**	0.00	0	1	**65**	0.00	0	2	**65**	0.00	0
166	250	2	14487.06	**63**	18	**63**	0.00	0	1	**63**	0.00	0	1	**63**	0.00	0
166	250	3	9658.04	**61**	64	**61**	0.00	0	1	**61**	0.00	0	2	**61**	0.00	0
166	250	4	7243.53	**65**	600	28	132.14	160	1	**65**	0.00	0	2	**65**	0.00	0
166	250	2	16096.74	**58**	13	**58**	0.00	0	1	**58**	0.00	0	1	**58**	0.00	0
166	250	3	10731.16	**62**	13	**62**	0.00	0	1	**62**	0.00	0	1	**62**	0.00	0
166	250	4	8048.37	**63**	17	**63**	0.00	0	1	**63**	0.00	0	2	**63**	0.00	0
208	208	2	17827.96	**58**	77	**58**	0.00	0	1	**58**	0.00	0	2	**58**	0.00	0
208	208	3	11885.31	**54**	20	**54**	0.00	0	1	**54**	0.00	0	3	**54**	0.00	0
208	208	4	8913.98	**58**	53	**58**	0.00	0	1	**58**	0.00	0	7	**58**	0.00	0
208	208	2	20056.46	**58**	31	**58**	0.00	0	1	**58**	0.00	0	1	**58**	0.00	0
208	208	3	13370.97	**58**	36	**58**	0.00	0	1	**58**	0.00	0	2	**58**	0.00	0
208	208	4	10028.23	**57**	118	**57**	0.00	0	1	**57**	0.00	0	5	**57**	0.00	0
208	208	2	22284.96	**57**	31	**57**	0.00	0	1	**57**	0.00	0	2	**57**	0.00	0
208	208	3	14856.64	**59**	80	**59**	0.00	0	1	**59**	0.00	0	2	**59**	0.00	0
208	208	4	11142.48	**60**	29	**60**	0.00	0	1	**60**	0.00	0	4	**60**	0.00	0

is identical to the number of reachable terminals $|C_R|$. In particular, for all instances with 318 vertices or more, this is happening.

4.2 Evaluating our MIP approaches on new instances

As the previous section has shown, due to the structure of the instances from AMSAL, it is quite easy to find tight upper bounds. Thus, the main difficulty to solve these instances to optimality is to find the optimal solutions and our approaches seem quite effective also for this purpose. In this section, we introduce a new set of instances, denoted as NEW. The instances are designed in such a way, that for all customers, there are at least two facilities covering it in the underlying graph. Note that

Table 8 Detailed results for instance set AMSAL for instances with 575 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2			
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
172	402	2	1732.69	**27**	24	**27**	0.00	0	1	**27**	0.00	0	1	**27**	0.00	0
172	402	3	1155.12	**26**	17	**26**	0.00	0	1	**26**	0.00	0	1	**26**	0.00	0
172	402	4	866.34	**30**	14	**30**	0.00	0	1	**30**	0.00	0	2	**30**	0.00	0
172	402	2	1949.27	**30**	24	**30**	0.00	0	1	**30**	0.00	0	1	**30**	0.00	0
172	402	3	1299.51	**27**	9	**27**	0.00	0	1	**27**	0.00	0	1	**27**	0.00	0
172	402	4	974.64	**28**	7	**28**	0.00	0	1	**28**	0.00	0	2	**28**	0.00	0
172	402	2	2165.86	**26**	10	**26**	0.00	0	1	**26**	0.00	0	1	**26**	0.00	0
172	402	3	1443.91	**29**	6	**29**	0.00	0	1	**29**	0.00	0	1	**29**	0.00	0
172	402	4	1082.93	**28**	6	**28**	0.00	0	1	**28**	0.00	0	2	**28**	0.00	0
229	345	2	2589.77	**27**	70	**27**	0.00	0	2	**27**	0.00	0	2	**27**	0.00	0
229	345	3	1726.51	**27**	16	**27**	0.00	0	2	**27**	0.00	0	3	**27**	0.00	0
229	345	4	1294.88	**30**	19	**30**	0.00	0	2	**30**	0.00	0	4	**30**	0.00	0
229	345	2	2913.49	**28**	75	**28**	0.00	0	2	**28**	0.00	0	2	**28**	0.00	0
229	345	3	1942.32	**29**	10	**29**	0.00	0	1	**29**	0.00	0	3	**29**	0.00	0
229	345	4	1456.74	**27**	24	**27**	0.00	0	2	**27**	0.00	0	4	**27**	0.00	0
229	345	2	3237.21	**26**	13	**26**	0.00	0	2	**26**	0.00	0	2	**26**	0.00	0
229	345	3	2158.14	**30**	16	**30**	0.00	0	2	**30**	0.00	0	3	**30**	0.00	0
229	345	4	1618.60	**29**	11	**29**	0.00	0	1	**29**	0.00	0	3	**29**	0.00	0
287	287	2	3624.82	**27**	17	**27**	0.00	0	3	**27**	0.00	0	3	**27**	0.00	0
287	287	3	2416.55	**29**	17	**29**	0.00	0	3	**29**	0.00	0	5	**29**	0.00	0
287	287	4	1812.41	**28**	72	**28**	0.00	0	3	**28**	0.00	0	8	**28**	0.00	0
287	287	2	4077.93	**30**	374	**30**	0.00	30	3	**30**	0.00	0	3	**30**	0.00	0
287	287	3	2718.62	**31**	49	**31**	0.00	0	3	**31**	0.00	0	4	**31**	0.00	0
287	287	4	2038.96	**30**	18	**30**	0.00	0	3	**30**	0.00	0	7	**30**	0.00	0
287	287	2	4531.03	**30**	20	**30**	0.00	0	2	**30**	0.00	0	3	**30**	0.00	0
287	287	3	3020.69	**30**	61	**30**	0.00	0	3	**30**	0.00	0	5	**30**	0.00	0
287	287	4	2265.51	**30**	15	**30**	0.00	0	3	**30**	0.00	0	6	**30**	0.00	0

due to the nature of the problem, it can still happen, that after applying the distance limit-based variable fixing/preprocessing as described in Theorem 2 that some customers cannot be reached for some values of L. For the runs in this section, we used a timelimit of 1800 seconds.

The instances are made available at https://msinnl.github.io/pages/instancescodes.html and are constructed as follows: $|F|$ facilities and $|C|$ customers are picked by taking random integers within [0, 1000] as the location of them in the Euclidean plane. The central depot is placed at point (500, 500). This is done for the following pairs of ($|F|$, $|C|$): (75, 225), (100, 300), (125, 375), for each pair three graphs are constructed. For the facility-customer coverage, we randomly pick between two and five of the nearest facilities for each customer. With this we ensure, that each customer can be covered (and also no trivial one-to-one relationship between

Table 9 Detailed results for instance set AMSAL for instances with 657 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2			
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
197	459	2	14375.04	**63**	297	**63**	0.00	12	1	**63**	0.00	0	2	**63**	0.00	0
197	459	3	9583.36	**67**	68	**67**	0.00	0	1	**67**	0.00	0	5	**67**	0.00	0
197	459	4	7187.52	**68**	600	–	–	0	1	**68**	0.00	0	9	**68**	0.00	0
197	459	2	16171.92	**65**	116	**65**	0.00	0	1	**65**	0.00	0	2	**65**	0.00	0
197	459	3	10781.28	**67**	245	**67**	0.00	0	1	**67**	0.00	0	8	**67**	0.00	0
197	459	4	8085.96	**68**	600	–	–	0	1	**68**	0.00	0	8	**68**	0.00	0
197	459	2	17968.80	**68**	62	**68**	0.00	0	1	**68**	0.00	0	1	**68**	0.00	0
197	459	3	11979.20	**66**	76	**66**	0.00	0	1	**66**	0.00	0	3	**66**	0.00	0
197	459	4	8984.40	**67**	136	**67**	0.00	0	1	**67**	0.00	0	6	**67**	0.00	0
262	394	2	22208.68	**76**	78	**76**	0.00	0	2	**76**	0.00	0	3	**76**	0.00	0
262	394	3	14805.79	**76**	600	–	–	0	2	**76**	0.00	0	7	**76**	0.00	0
262	394	4	11104.34	**78**	600	1	7699.92	0	2	**78**	0.00	0	16	**78**	0.00	0
262	394	2	24984.76	**77**	68	**77**	0.00	0	2	**77**	0.00	0	3	**77**	0.00	0
262	394	3	16656.51	**78**	600	2	3799.98	0	2	**78**	0.00	0	7	**78**	0.00	0
262	394	4	12492.38	**77**	513	**77**	0.00	0	2	**77**	0.00	0	17	**77**	0.00	0
262	394	2	27760.85	**76**	52	**76**	0.00	0	3	**76**	0.00	0	3	**76**	0.00	0
262	394	3	18507.23	**73**	343	**73**	0.00	0	3	**73**	0.00	0	7	**73**	0.00	0
262	394	4	13880.42	**81**	33	**81**	0.00	0	2	**81**	0.00	0	15	**81**	0.00	0
328	328	2	30838.17	**83**	283	**83**	0.00	0	4	**83**	0.00	0	4	**83**	0.00	0
328	328	3	20558.78	**84**	600	–	–	10	3	**84**	0.00	0	20	**84**	0.00	0
328	328	4	15419.08	**83**	106	**83**	0.00	0	4	**83**	0.00	0	26	**83**	0.00	0
328	328	2	34692.94	**82**	440	**82**	0.00	0	4	**82**	0.00	0	6	**82**	0.00	0
328	328	3	23128.62	**83**	224	**83**	0.00	0	4	**83**	0.00	0	10	**83**	0.00	0
328	328	4	17346.47	**80**	600	–	–	0	5	**80**	0.00	0	29	**80**	0.00	0
328	328	2	38547.71	**78**	139	**78**	0.00	0	4	**78**	0.00	0	5	**78**	0.00	0
328	328	3	25698.47	**82**	600	50	64.00	1	5	**82**	0.00	0	9	**82**	0.00	0
328	328	4	19273.85	**81**	600	–	–	0	3	**81**	0.00	0	38	**81**	0.00	0

some facility and customer can occur). Based on these underlying graphs, instances are created by choosing $k \in \{2, 3, 4\}$, and setting L using the following scheme for $\alpha = \{0.1, 0.15, 0.2, 0.25, 0.3\}$: Let $L(\alpha)$ be the sum of the distances of the $\lceil \alpha |F| \rceil$ nearest facilities to the central depot. The value of L is set as $L(\alpha)/l$. In total, this set contains $3 \cdot 3 \cdot 3 \cdot 5 = 135$ instances (three underlying graphs for each ($|F|$, $|C|$) and the different parameters for k and L).

Again, we first analyze the effect of our different settings for the algorithms. Figures 5 and 6 give plots of the runtime to optimality and Figs. 7 and 8 gives plots of the optimality gap for the instances from set NEW and our settings.

From Figures 5 and 6, we can see that these instances seem much more difficult to solve to optimality than the ones from set AMSAL. The best performing settings, namely C++HS, C++H, E++H2 and E++H, only manage to solve about 45% of the

Table 10 Detailed results for instance set AMSAL for instances with 724 vertices

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2			
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
217	506	2	19912.23	**176**	600	171	2.92	100	2	**176**	0.00	0	2	**176**	0.00	0
217	506	3	13274.82	**187**	600	82	128.05	2	1	**187**	0.00	0	6	**187**	0.00	0
217	506	4	9956.12	**187**	600	156	19.87	1	1	**187**	0.00	0	9	**187**	0.00	0
217	506	2	22401.26	**179**	292	**179**	0.00	0	2	**179**	0.00	0	2	**179**	0.00	0
217	506	3	14934.17	**179**	600	94	90.43	2	2	**179**	0.00	0	5	**179**	0.00	0
217	506	4	11200.63	**184**	600	126	46.03	1	1	**184**	0.00	0	13	**184**	0.00	0
217	506	2	24890.29	**180**	600	–	–	70	2	**180**	0.00	0	2	**180**	0.00	0
217	506	3	16593.53	**183**	600	159	15.09	1	2	**183**	0.00	0	4	**183**	0.00	0
217	506	4	12445.15	**176**	600	142	23.94	3	2	**176**	0.00	0	17	**176**	0.00	0
289	434	2	29871.47	**187**	487	**187**	0.00	0	3	**187**	0.00	0	4	**187**	0.00	0
289	434	3	19914.32	**183**	600	–	–	0	3	**183**	0.00	0	13	**183**	0.00	0
289	434	4	14935.74	**175**	600	43	306.98	0	3	**175**	0.00	0	20	**175**	0.00	0
289	434	2	33605.41	**184**	600	122	50.82	2	3	**184**	0.00	0	5	**184**	0.00	0
289	434	3	22403.60	**176**	600	55	220.00	1	3	**176**	0.00	0	10	**176**	0.00	0
289	434	4	16802.70	**183**	600	62	195.16	1	3	**183**	0.00	0	22	**183**	0.00	0
289	434	2	37339.34	**177**	373	**177**	0.00	0	3	**177**	0.00	0	3	**177**	0.00	0
289	434	3	24892.89	**182**	600	121	50.41	2	3	**182**	0.00	0	10	**182**	0.00	0
289	434	4	18669.67	**174**	600	132	31.82	1	3	**174**	0.00	0	23	**174**	0.00	0
361	362	2	40848.98	**205**	600	–	–	0	5	**205**	0.00	0	9	**205**	0.00	0
361	362	3	27232.65	**200**	600	–	–	0	5	**200**	0.00	0	23	**200**	0.00	0
361	362	4	20424.49	**214**	600	–	–	0	5	**214**	0.00	0	32	**214**	0.00	0
361	362	2	45955.09	**195**	600	123	58.54	1	5	**195**	0.00	0	10	**195**	0.00	0
361	362	3	30636.73	**207**	600	3	6799.98	0	5	**207**	0.00	0	53	**207**	0.00	0

Table 10 (continued)

$\|F\|$	$\|C\|$	k	L	$\|C_R\|$	C+				C++HS				E++H2			
					$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
361	362	4	22977.55	**204**	600	–	–	0	5	**204**	0.00	0	36	**204**	0.00	0
361	362	2	51061.22	**208**	600	–	–	0	5	**208**	0.00	0	10	**208**	0.00	0
361	362	3	34040.81	**206**	600	–	–	0	7	**206**	0.00	0	26	**206**	0.00	0
361	362	4	25530.61	**216**	600	–	–	0	7	**216**	0.00	0	59	**216**	0.00	0

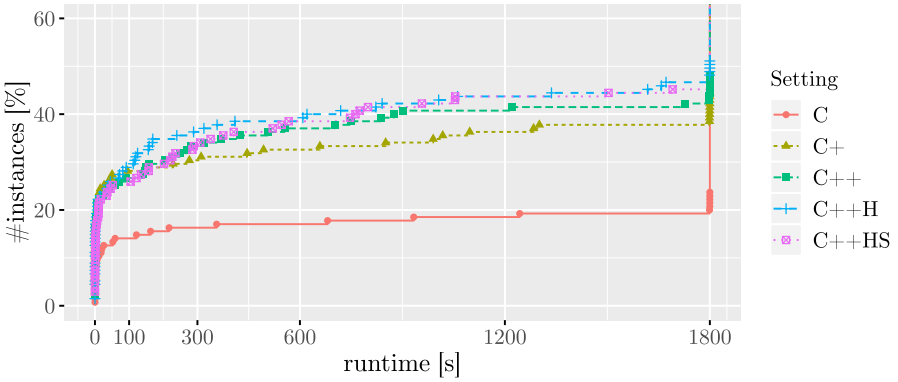


Fig. 5 Runtime to optimality for instance set NEW and different settings based on (*C*). For better readability, the y-axis only goes up to 60% of the instances

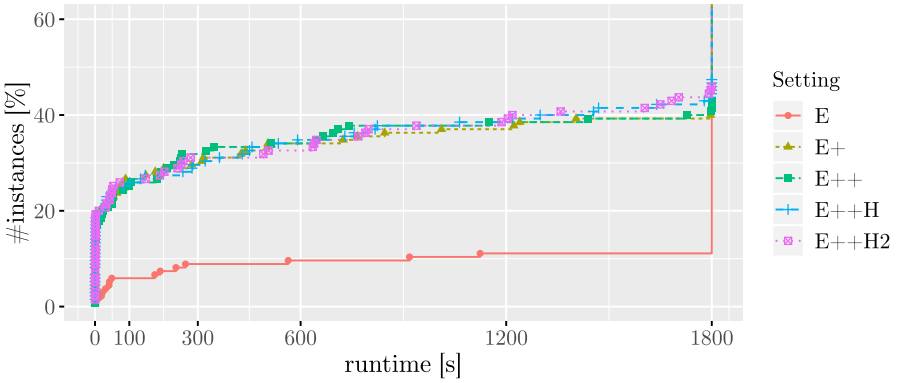


Fig. 6 Runtime to optimality for instance set NEW and different settings based on (*E*). For better readability, the y-axis only goes up to 60% of the instances

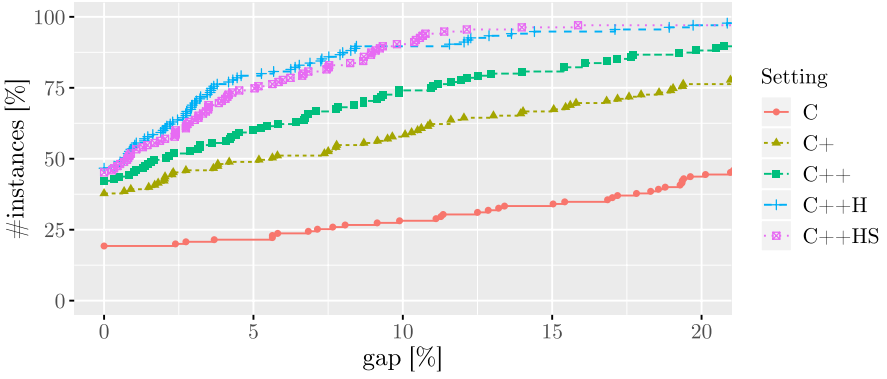


Fig. 7 Optimality gap for instance set NEW and different settings based on (*C*). For better readability, the x-axis only goes up to 20% gap

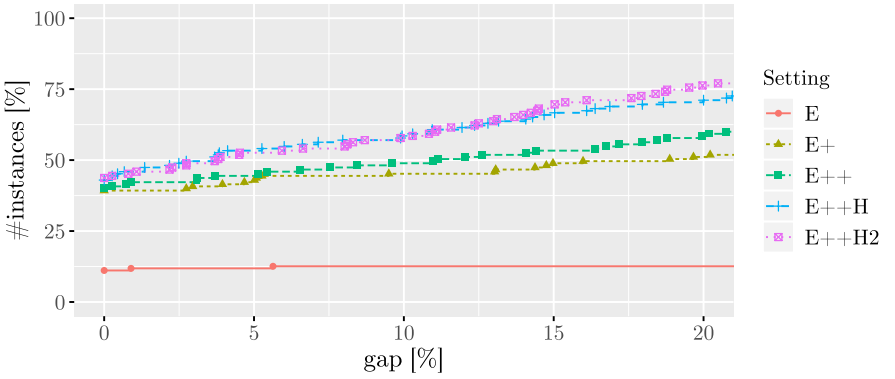


Fig. 8 Optimality gap for instance set NEW and different settings based on (*E*). For better readability, the x-axis only goes up to 20% gap

instances to optimality within the timelimit. In general, all settings based on (*E*), except E, have a quite similar performance. For settings based on (*C*), the basic setting C has the worst performance. Thus, as already seen in the results for set AMSAL, the variable fixings seem quite effective. However, differently to the results for set AMSAL, C++ now performs better than C+. Looking also at the optimality gaps given in Figs. 7 and 8, the settings based on (*C*) seem to performance a little better than the settings based on (*E*). Moreover, the performance difference of the settings becomes more pronounced.

Tables 11, 12, 13 give detailed results for the instances of set NEW and settings C+, C++HS and E++H2. In the tables, we report for each instance the number of vehicles (k), the distance limit (L) and the number of customers covered by at least one facility which is reachable from the central depot within the given distance limit ($|C_R|$ "reachable customers," i.e., following Theorem 2). For each setting, we report the runtime ($t[s]$), value of the best solution found (z^*), optimality gap ($g[\%]$), and number of branch-and-bound nodes ($\#nBBn$). A **bold** entry in z^* indicates that optimality has be proven (as can also be seen by the optimality gap $g[\%]$ being zero). A **bold** entry in $|C_R|$ indicates, that the optimal solution consists of all reachable customers.

In the tables, we can see that the instances with smaller distance limit seem to be easier, as more of them can be solved to optimality compared to instances with larger distance limit. This is not too surprising, as with smaller distance limit, the feasible region of the problem becomes smaller, and a smaller number of facilities are reachable within the distance limit. In general, the instances seem to become harder when the number of nodes is larger. The number of vehicles does not seem to influence the performance much. There are 18 out of 135 instances, where the optimal solution value is equal to $|C_R|$. Setting C+ manages to solve 51 instances to optimality, C++HS manages to solve 61 instances to optimality and E++H2 manages to solve 59 instances to optimality.

Table 11 Detailed results for instance set NEW for instances with 301 vertices

k	L	$\lvert C_R\rvert$	C+				C++HS				E++H2			
			$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
2	422.52	45	0	**36**	0.00	55	0	**36**	0.00	0	0	**36**	0.00	0
2	465.91	58	0	**49**	0.00	17	1	**49**	0.00	21	1	**49**	0.00	40
2	538.19	56	0	**46**	0.00	47	0	**46**	0.00	9	1	**46**	0.00	3
2	824.88	145	47	**78**	0.00	4460	158	**78**	0.00	5375	49	**78**	0.00	366
2	900.85	206	1800	99	2.02	86072	547	**99**	0.00	12676	279	**99**	0.00	2237
2	1048.14	224	1800	135	9.15	25743	104	**135**	0.00	267	506	**135**	0.00	1445
2	1188.67	225	1800	130	26.36	20773	227	**135**	0.00	1144	1205	**135**	0.00	3492
2	1277.91	225	1800	146	22.23	19684	1800	148	4.52	3999	1800	146	28.88	3808
2	1461.77	225	1800	175	17.98	11649	1800	181	4.29	3901	1800	181	12.98	2918
2	1712.49	225	1800	175	23.51	10258	1800	194	3.04	2400	1800	184	19.96	2533
2	1801.24	225	1800	184	19.36	9185	1800	205	2.40	2223	1800	187	18.72	2308
2	2047.24	225	1800	203	10.84	3091	799	**224**	0.00	164	1800	211	6.64	422
2	2336.58	**225**	1800	219	2.74	2814	1055	**225**	0.00	800	1800	217	3.69	224
2	2374.58	225	1800	221	1.81	3891	1800	224	0.45	5341	1800	220	2.27	226
2	2664.08	**225**	445	**225**	0.00	524	5	**225**	0.00	0	33	**225**	0.00	10
3	258.17	39	0	**34**	0.00	0	0	**34**	0.00	0	0	**34**	0.00	0
3	285.40	**27**	0	**27**	0.00	0	0	**27**	0.00	0	0	**27**	0.00	0
3	349.69	**45**	0	**45**	0.00	0	0	**45**	0.00	0	0	**45**	0.00	0
3	448.37	63	0	**58**	0.00	40	1	**58**	0.00	3	1	**58**	0.00	3
3	552.02	78	1	**61**	0.00	440	1	**61**	0.00	7	1	**61**	0.00	2
3	598.50	121	4	**93**	0.00	1728	34	**93**	0.00	1678	42	**93**	0.00	434
3	641.46	136	12	**100**	0.00	2141	157	**100**	0.00	6334	43	**100**	0.00	502
3	789.53	179	15	**104**	0.00	1567	11	**104**	0.00	85	202	**104**	0.00	747
3	816.54	192	311	**138**	0.00	35957	1800	138	1.45	39070	241	**138**	0.00	1314
3	959.52	213	1800	151	3.66	52735	1800	151	3.31	18594	1800	150	8.02	3904
3	1145.94	224	1800	175	5.18	23560	376	**177**	0.00	1445	1800	171	17.59	1483
3	1151.62	225	1800	192	7.50	24894	1800	192	3.41	7940	1800	191	11.04	1840
3	1319.82	225	1800	195	10.49	20878	1800	198	5.84	5952	1800	196	13.11	1309
3	1522.41	225	1800	220	2.27	4077	1800	223	0.90	3646	1800	200	12.50	121
3	1574.57	225	1800	213	5.63	3705	1800	215	3.84	3375	1800	203	10.84	1001
4	211.30	**23**	0	**23**	0.00	0	0	**23**	0.00	0	0	**23**	0.00	0
4	219.33	**20**	0	**20**	0.00	0	0	**20**	0.00	0	0	**20**	0.00	0
4	241.75	**17**	0	**17**	0.00	0	0	**17**	0.00	0	0	**17**	0.00	0
4	401.70	54	0	**50**	0.00	5	2	**50**	0.00	178	2	**50**	0.00	9
4	415.60	59	0	**58**	0.00	5	3	**58**	0.00	293	1	**58**	0.00	12
4	449.44	44	0	**43**	0.00	0	2	**43**	0.00	112	0	**43**	0.00	0
4	556.82	90	8	**78**	0.00	2369	285	**78**	0.00	15876	50	**78**	0.00	499
4	604.39	121	39	**96**	0.00	17840	289	**96**	0.00	18182	11	**96**	0.00	104
4	644.47	123	5	**94**	0.00	1070	2	**94**	0.00	45	0	**94**	0.00	0
4	777.57	163	229	**130**	0.00	28517	1800	130	0.77	32484	938	**130**	0.00	2508
4	884.61	185	990	**154**	0.00	76798	1800	154	1.30	23420	1359	**154**	0.00	2486

Table 11 (continued)

k	L	$\lvert C_R \rvert$	C+				C++HS				E++H2			
			$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
4	920.99	199	851	**157**	0.00	43617	1691	**157**	0.00	21797	1800	155	11.58	2153
4	1041.71	222	1800	175	4.19	39710	1800	176	3.49	10974	1800	173	14.24	1221
4	1176.85	225	1800	200	3.81	25538	1800	200	2.86	7930	1800	179	21.91	34
4	1219.09	224	1800	201	5.80	29762	1800	205	3.16	6979	1800	186	18.78	534

5 Conclusion

In this paper, we studied the recently introduced *time-constrained maximal covering routing problem*. The problem is a generalization of well-known problems such as the *(team) orienteering problem* and *maximal covering location*. In the problem, we are given a central depot, facilities and customers. Each customer can be served by a subset of the facilities. Moreover, we are given distances between the facilities (and central depot), a distance limit L and a number of vehicles k. A feasible solution consists of k Hamiltonian cycles on subsets of the facilities and the central depot. All cycles must contain the central depot and respect the distance limit L. The goal is to maximize the number of customers covered by the facilities in the solution. The problem was introduced in Amiri and Salari (2019), where an exact mixed-integer programming (MIP) approach and several metaheuristics were proposed.

We introduced two new MIP formulations and presented exact solution frameworks based on these MIP formulations. We evaluated our solution approaches on the instances from literature for the problem (from Amiri and Salari (2019)). The computational study revealed that our algorithms were able to find the provably optimal solution for 267 out of 270 instances, including 123 instances, for which the optimal solution was not known before. Moreover, for most of the instances, our algorithms only took a few seconds, being up to five magnitude faster than the exact MIP approach presented in Amiri and Salari (2019). The computational study also showed that the instances from Amiri and Salari (2019) have some issues, which potentially decrease their usefulness as benchmark instances. In particular, there are often many customers, which are not associated with any facility in the instance, and thus can never be in any feasible solution. In many instances, the value of the optimal solution is then simply similar to the number of the remaining customers. We thus also introduced a new set of more challenging instances.

There are several avenues for further work: Naturally, similar to other routing problems, additional side-constraints based on real-life considerations can be added, such as multiple-depots, time-windows, capacities, or uncertainty. Moreover, weighted customers and assignment-costs for customers could also be interesting extensions. To deal with more difficult instances, such as the ones introduced in this paper, the design and (re-)evaluation of metaheuristics could be a promising topic. We note that Amiri and Salari (2019) already proposed and tested some metaheuristics for the problem, however, as discussed, the instances used in their

Table 12 Detailed results for instance set NEW for instances with 401 vertices

k	L	$\lvert C_R\rvert$	C+				C++HS				E++H2			
			$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
2	486.16	91	9	**53**	0.00	3122	10	**53**	0.00	298	53	**53**	0.00	247
2	620.83	128	95	**80**	0.00	15737	9	**80**	0.00	61	72	**80**	0.00	220
2	784.34	208	1097	**89**	0.00	44356	202	**89**	0.00	1979	646	**89**	0.00	2135
2	927.20	262	1800	129	10.91	25332	520	**129**	0.00	1994	1722	**129**	0.00	3345
2	1086.09	294	1800	154	17.57	17713	1800	154	2.39	5510	1800	154	14.46	1786
2	1345.72	300	1800	172	39.66	5710	1800	179	7.51	845	1800	159	61.88	1162
2	1435.36	300	1800	206	22.55	4488	1800	212	1.69	1526	1800	197	33.56	939
2	1672.82	300	1800	171	66.32	4216	1800	223	10.57	580	1800	189	55.63	637
2	1973.23	300	1800	203	44.57	1565	1800	259	8.67	415	1800	218	37.36	592
2	2019.76	300	1800	215	37.21	1085	1800	265	6.63	420	1800	249	20.48	604
2	2309.20	300	1800	227	32.16	1037	1800	284	5.04	386	1800	270	11.11	159
2	2638.19	300	1800	225	33.33	927	1800	284	5.63	199	1800	273	9.89	102
2	2690.88	300	1800	253	18.58	710	1800	299	0.33	273	1800	272	10.29	107
2	3004.45	**300**	1800	295	1.69	1200	567	**300**	0.00	26	1800	289	3.81	35
2	3331.40	**300**	1800	294	2.04	1344	46	**300**	0.00	0	148	**300**	0.00	45
3	357.90	44	0	**41**	0.00	2	1	**41**	0.00	18	0	**41**	0.00	0
3	377.34	53	0	**46**	0.00	0	1	**46**	0.00	27	0	**46**	0.00	0
3	400.40	75	0	**65**	0.00	83	0	**65**	0.00	3	0	**65**	0.00	0
3	699.51	186	50	**119**	0.00	4736	308	**119**	0.00	7128	768	**119**	0.00	2070
3	711.49	189	494	**124**	0.00	41932	1504	**124**	0.00	25797	1605	**124**	0.00	2972
3	724.46	178	147	**117**	0.00	12124	763	**117**	0.00	16257	789	**117**	0.00	1634
3	1066.84	300	1800	190	14.03	12627	1800	191	5.99	4127	1800	170	44.69	441
3	1120.62	293	1800	199	13.94	16679	1800	201	7.45	2480	1800	183	36.94	521
3	1155.64	297	1800	202	15.45	12933	1800	212	2.36	4142	1800	189	36.45	377
3	1459.29	300	1800	213	40.85	1882	1800	251	12.14	842	1800	218	37.61	249
3	1592.28	300	1800	248	20.97	1094	1800	267	8.94	956	1800	247	21.46	193
3	1624.41	300	1800	264	12.05	3083	1800	268	9.09	1139	1800	244	22.95	173
3	1923.85	**300**	1800	248	20.97	1056	1054	**300**	0.00	152	1800	262	14.50	102
3	2100.66	300	1800	289	3.81	1015	1800	294	2.04	513	1800	287	4.53	49
3	2131.08	300	1800	276	8.70	648	1800	292	2.74	521	1800	277	8.30	26
4	260.83	**44**	0	**44**	0.00	0	0	**44**	0.00	0	0	**44**	0.00	0
4	308.07	**43**	0	**43**	0.00	0	0	**43**	0.00	0	0	**43**	0.00	0
4	331.83	**44**	0	**44**	0.00	0	0	**44**	0.00	0	0	**44**	0.00	0
4	489.73	102	10	**82**	0.00	2653	135	**82**	0.00	6363	31	**82**	0.00	197
4	571.59	88	6	**82**	0.00	1062	36	**82**	0.00	1103	2	**82**	0.00	0
4	593.91	128	3	**103**	0.00	1144	12	**103**	0.00	322	190	**103**	0.00	495
4	772.92	209	1800	159	2.01	86529	1800	159	4.06	17704	1800	159	8.12	1408
4	876.54	258	1800	170	7.38	65194	1800	170	2.79	10037	1800	158	28.64	488
4	909.80	266	1800	205	0.89	82819	1800	205	1.87	13004	1800	194	13.69	844
4	1097.81	289	1800	231	7.78	19192	1800	232	3.94	3190	1800	219	22.11	256
4	1235.17	300	1800	247	13.03	11241	1800	254	6.67	2149	1800	218	35.59	111

Table 12 (continued)

k	L	$\lvert C_R\rvert$	C+				C++HS				E++H2			
			$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
4	1265.89	300	1800	260	10.08	13571	1800	262	7.53	2551	1800	220	34.93	2
4	1456.87	300	1800	274	9.49	1716	1800	289	3.50	1309	1800	251	19.52	104
4	1619.40	300	1800	202	48.51	1031	1800	298	0.67	457	1800	267	12.36	40
4	1673.74	300	1800	298	0.67	1297	1800	293	2.39	547	1800	292	2.74	53

Table 13 Detailed results for instance set NEW for instances with 501 vertices

k	L	$\lvert C_R\rvert$	C+				C++HS				E++H2			
			$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
2	670.15	175	1018	**91**	0.00	77176	775	**91**	0.00	10728	635	**91**	0.00	1065
2	741.09	247	1284	**100**	0.00	42536	220	**100**	0.00	1074	1704	**100**	0.00	2430
2	768.77	229	1301	**107**	0.00	48389	236	**107**	0.00	819	1650	**107**	0.00	1575
2	1246.33	375	1800	174	52.99	3176	1800	176	24.47	406	1800	168	77.70	270
2	1321.79	375	1800	165	63.45	3367	1800	198	15.87	362	1800	157	101.89	162
2	1336.87	375	1800	194	44.84	2245	1800	207	11.39	363	1800	166	86.87	199
2	1952.41	375	1800	230	57.89	493	1800	258	37.68	48	1800	247	50.72	229
2	1992.25	375	1800	216	72.84	948	1800	264	34.39	139	1800	268	38.11	289
2	2104.88	375	1800	235	59.57	309	1800	274	32.44	41	1800	263	42.59	281
2	2875.80	375	1800	298	25.84	491	1800	329	13.98	27	1800	326	15.03	97
2	2885.57	375	1800	284	32.04	363	1800	371	1.08	116	1800	365	2.74	91
2	3072.01	375	1800	336	11.61	381	1800	351	6.84	32	1800	367	2.18	86
2	3751.65	**375**	1800	321	16.82	1396	957	**375**	0.00	42	251	**375**	0.00	21
2	3776.04	**375**	1800	320	17.19	883	339	**375**	0.00	10	798	**375**	0.00	70
2	3966.92	**375**	1800	315	19.05	866	747	**375**	0.00	23	500	**375**	0.00	26
3	556.28	127	27	**95**	0.00	5962	122	**95**	0.00	1932	258	**95**	0.00	386
3	595.57	133	13	**90**	0.00	986	10	**90**	0.00	40	640	**90**	0.00	627
3	673.59	171	276	**118**	0.00	64371	1800	118	0.85	31745	1218	**118**	0.00	2007
3	940.28	323	1800	189	15.80	17372	1800	188	6.24	2300	1800	164	51.25	132
3	956.60	339	1800	177	15.60	21662	1800	175	8.94	2275	1800	165	44.56	384
3	1175.17	374	1800	223	30.16	5628	1800	245	9.80	852	1800	211	57.64	80
3	1355.44	375	1800	237	43.66	2662	1800	279	10.38	410	1800	225	59.89	76
3	1409.29	375	1800	251	36.78	1688	1800	289	10.44	495	1800	240	53.17	60
3	1726.52	375	1800	279	34.41	470	1800	331	10.76	49	1800	318	17.92	196
3	1959.48	375	1800	303	23.76	328	1800	343	9.33	104	1800	323	16.10	52
3	2020.80	375	1800	270	38.89	240	1800	345	8.70	95	1800	329	13.98	79
3	2423.42	375	1800	279	34.41	427	1800	365	2.74	40	1800	361	3.88	32
3	2532.45	375	1800	348	7.76	338	1800	374	0.27	81	1800	354	5.93	25
3	2598.26	375	1800	342	9.65	695	1800	373	0.54	52	1800	374	0.27	19
3	3085.30	**375**	1800	348	7.76	710	17	**375**	0.00	0	2	**375**	0.00	0

Table 13 (continued)

k	L	$\lvert C_R \rvert$	C+				C++HS				E++H2			
			$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$	$t[s]$	z^*	$g[\%]$	$\#BB$
4	378.59	76	1	**70**	0.00	327	3	**70**	0.00	62	1	**70**	0.00	0
4	393.43	77	0	**67**	0.00	69	6	**67**	0.00	192	0	**67**	0.00	0
4	399.85	74	0	**69**	0.00	70	52	**69**	0.00	1766	1	**69**	0.00	0
4	671.32	192	658	**137**	0.00	71872	1800	137	3.48	17993	1186	**137**	0.00	1717
4	673.95	193	1800	129	1.49	110330	1800	129	3.16	14753	1800	124	18.40	552
4	704.28	174	1800	132	2.31	147910	1800	132	5.17	16370	1800	132	4.51	2033
4	1012.70	355	1800	232	15.05	11964	1800	233	8.24	2256	1800	198	49.97	73
4	1042.77	355	1800	255	18.97	14280	1800	263	9.23	1820	1800	227	44.97	48
4	1080.47	365	1800	240	19.39	10880	1800	244	10.66	1730	1800	202	58.91	51
4	1487.78	375	1800	308	21.75	562	1800	359	4.13	309	1800	298	25.84	39
4	1562.10	375	1800	308	21.75	526	1800	360	4.17	145	1800	326	15.03	4
4	1588.27	375	1800	285	31.58	440	1800	345	8.70	200	1800	325	15.38	73
4	1939.26	375	1800	336	11.61	269	1800	372	0.81	68	1800	345	8.70	7
4	2058.55	**375**	1800	340	10.29	647	406	**375**	0.00	15	1800	372	0.81	24
4	2098.21	375	1800	339	10.62	395	1800	371	1.08	80	1800	371	1.08	43

work to evaluate their approaches had some issues. Investigating an exact approach based on a formulation with exponentially many variables (i.e., column generation/branch-and-price) could also be a fruitful topic, as such approaches often work quite well for routing problems of similar type.

Acknowledgements This works was supported by the Linz Institute of Technology (Project LIT-2019-7-YOU-211) and the JKU Business School.

Funding Open access funding provided by Johannes Kepler University Linz.

References

Amiri A, Salari M (2019) Time-constrained maximal covering routing problem. OR Spectr 41(2):415–468

Bianchessi N, Mansini R, Speranza MG (2018) A branch-and-cut algorithm for the team orienteering problem. Int T Oper Res 25(2):627–635

Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The maximum clique problem. In: Handbook of combinatorial optimization, Springer, pp 1–74
Boussier S, Feillet D, Gendreau M (2007) An exact algorithm for team orienteering problems. 4OR 5(3):211–230
Butt SE, Ryan DM (1999) An optimal solution procedure for the multiple tour maximum collection problem using column generation. Comput Oper Res 26(4):427–441
Chao I, Golden BL, Wasil EA (1996) The team orienteering problem. Eur J Oper Res 88(3):464–474
Cherkassky BV, Goldberg AV (1995) On implementing push-relabel method for the maximum flow problem. In: International conference on integer programming and combinatorial optimization, Springer, pp 157–171
Church R, Velle CR (1974) The maximal covering location problem. Pap Reg Sci 32(1):101–118
Conforti M, Cornuéjols G, Zambelli G (2014) Integer programming, vol 271. Springer, Berlin
CPLEX (2020a) Cut callback. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/progr_adv/callbacks_adv/08_cut_cb.html, accessed 2020-04-10
CPLEX (2020b) Differences between user cuts and lazy constraints. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/progr_adv/usr_cut_lazy_constr/04_diffs.html, accessed 2020-04-10
Dang DC, El-Hajj R, Moukrim A (2013) A branch-and-cut algorithm for solving the team orienteering problem. In: International conference on AI and OR techniques in constraint programming for combinatorial optimization problems, Springer, pp 332–339
El-Hajj R, Dang DC, Moukrim A (2016) Solving the team orienteering problem with cutting planes. Comput Oper Res 74:21–30
Fischetti M, Gonzalez JJS, Toth P (1998) Solving the orienteering problem through branch-and-cut. INFORMS J Comput 10(2):133–148
Gendreau M, Laporte G, Semet F (1998) A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks 32(4):263–273
Golden BL, Levy L, Vohra R (1987) The orienteering problem. Nav Res Log 34(3):307–318
Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: a survey of recent variants, solution approaches and applications. Eur J Oper Res 255(2):315–332
Keshtkaran M, Ziarati K, Bettinelli A, Vigo D (2016) Enhanced exact solution methods for the team orienteering problem. Int J Prod Res 54(2):591–601
Koch T, Martin A (1998) Solving steiner tree problems in graphs to optimality. Networks 32(3):207–232
Laporte G, Martello S (1990) The selective travelling salesman problem. Discrete Appl Math 26(2–3):193–207
Laporte G, Nickel S, da Gama FS (2015) Location science, vol 528. Springer, Berlin
Leifer AC, Rosenwein MB (1994) Strong linear programming relaxations for the orienteering problem. Eur J Oper Res 73(3):517–523
Maffioli F, Sciomachen A (1997) A mixed-integer model for solving ordering problems with side constraints. Ann Oper Res 69:277–297
Naji-Azimi Z, Salari M (2014) The time constrained maximal covering salesman problem. Appl Math Model 38(15–16):3945–3957
Ozbaygin G, Yaman H, Karasan OE (2016) Time constrained maximal covering salesman problem with weighted demands and partial coverage. Comput Oper Res 76:226–237
Poggi M, Viana H, Uchoa E (2010) The team orienteering problem: Formulations and branch-cut and price. In: 10th Workshop on algorithmic approaches for transportation modelling, optimization, and systems (ATMOS'10), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
Reinelt G (1991) TSPLIB - a traveling salesman problem library. ORSA J Comput 3(4):376–384
Toth P, Vigo D (2014) Vehicle routing: problems, methods, and applications. SIAM, Philadelphia
Tsiligirides T (1984) Heuristic methods applied to orienteering. J Oper Res Soc 35(9):797–809