A p-step Formulation for the Capacitated Vehicle Routing Problem

Twan Dollevoet[1], Pedro Munari[2], and Remy Spliet[1]

[1]Econometric Institute, Erasmus University Rotterdam, the Netherlands

[2]Production Engineering Department, Federal University of São Carlos, Brazil

Econometric Institute Report Series: EI-2020-01

Abstract

We introduce a p-step formulation for the capacitated vehicle routing problem (CVRP). The parameter p indicates the length of partial paths corresponding to the used variables. This provides a family of formulations including both the traditional arc-based and path-based formulations. Hence, it is a generalization which unifies arc-based and path-based formulations, while also providing new formulations. We show that the LP bound of the p-step formulation is increasing in p, although not monotonically. Furthermore, we prove that computing the set partitioning bound is NP-hard. This is a meaningful result in itself, but combined with the p-step formulation this also allows us to show that there does not exist a strongest compact formulation for the CVRP, if $P \neq NP$. While ending the search for a strongest compact formulation, we propose the search for the strongest formulation of the CVRP with a number of variables and constraints limited by a polynomial of fixed degree. We provide new strongest such formulations of degree three and higher by using a corresponding p-step formulation. Furthermore, the results of our experiments suggest that there are computational advantages from using the p-step formulation, instead of traditional arc-based and path-based formulations.

1 Introduction

The capacitated vehicle routing problem (CVRP) is a classical optimization problem in the field of operations research. It is the problem of delivering demands of customers from a depot using homogeneous vehicles with limited capacity at minimal costs. Together with its numerous extended versions, it has many practical applications and is much studied, see Toth and Vigo (2014).

The most successful exact approaches for the CVRP rely on an integer or mixed integer linear programming formulation. We distinguish between two types of formulations that are dominant in the scientific literature. Arc-based formulations use binary variables indicating whether an arc is used, while path-based formulations use binary variables indicating the use of a path, i.e., route. Examples of arc-based formulations are the single-commodity flow formulation (Gavish 1984), the two-index vehicle flow formulation (Laporte and Nobert 1987), and the two-commodity flow formulation (Baldacci et al. 2004). The main example of a path based formulation is the set partitioning formulation, which was first introduced by Balinski and Quandt (1964) and is studied extensively since. Arc-based formulations are known for relatively weak LP bounds which can be computed fast, while path-based formulations have stronger LP bounds which require more time to compute.

The LP bound of the set partitioning formulation, referred to as the set partitioning bound, is usually computed with a column generation algorithm which has exponential worst-case computation time. Fast algorithms for the CVRP like that of Pecin et al. (2017), typically make use of such a column generation algorithm in which they additionally employ route relaxations that allow cyclic routes. An important example is the ng-route relaxation of Baldacci et al. (2011). Applying a route relaxation effectively yields a new path-based formulation, which is often still referred to as a set partitioning formulation. The result is an LP bound which, although weaker, can usually be computed in pseudo-polynomial time. Alternatively, researchers have been pursuing a strongest compact formulation, since these provide LP bounds that can be computed in polynomial time. A recent overview in which multiple formulations are compared can be found in Letchford and Salazar-González (2015). To the best of our knowledge, the current strongest compact formulation is due to Leggieri and Haouari (2017).

In this paper, we present a new formulation of the CVRP which we refer to as the p-step formulation. It makes use of variables corresponding to partial paths of length precisely p, or at most p if the partial path starts at the depot. The p-step formulation can be considered a family of formulations, one for each positive integer value of p. Observe that when p is one, the p-step formulation is an arc-based formulation, while for p larger than the number of customers in an instance the p-step formulation is a path-based formulation. The p-step formulation could therefore be considered a generalized formulation of the CVRP, with arc-based and path-based formulations at its extremes. Moreover, it provides new formulations based on partial paths in between these extremes.

We are not the first to consider a formulation for the CVRP based on partial paths. Perhaps the best documented study is the technical report by Jepsen and Petersen (2009). They consider a giant tour representation of a solution to the CVRP which is decomposed in partial paths. A formulation is presented in which the decision variables indicate which partial paths are selected. They conclude after only some preliminary experiments that they do not want to pursue their idea further, without doing a more in depth analysis.

For our p-step formulation, we provide a detailed analysis. We show that the 1-step formulation is equally strong as the single-commodity flow formulation (Gavish 1984) and that the LP bound of the p-step formulation increases in p until ultimately reaching the set partitioning bound, although the increase is not necessarily monotonic. This provides opportunities to work with a formulation that has stronger LP bounds than the single-commodity flow formulation, but which could be computed more efficiently than the set partitioning bound. In that sense, the formulation could be regarded in the same way as the more traditional route relaxations.

In this paper, we also prove the seemingly unrelated fact that computing the set partitioning bound of the CVRP is NP-hard. This result justifies the use of exponential time algorithms, such as column generation, to compute the set partitioning bound. Moreover, it provides insight into why researchers have been more successful using route relaxations, instead of using the set partitioning formulation with elementary paths.

This complexity result in combination with our p-step formulation enables us to prove that no strongest compact formulation exists, if $P \neq NP$. The argument is that for any compact formulation, a compact p-step formulation can be found which is not weaker. This means that researchers no longer have to search for the strongest compact formulation. However, we argue that it is still relevant to instead search for strongest formulations with the number of variables and constraints of fixed polynomial degree. Subsequently, we point out new strongest known formulations with the number of variables and constraints of fixed polynomial degree three and higher.

We additionally investigate whether the p-step formulation has the potential to provide computational advantages. Note that the p-step formulation is complementary to any approach using an arc-based or path-based formulation, and merely provides additional opportunities corresponding to intermediate values of p. Hence, we are particularly interested whether faster computation times can be obtained using intermediate values of p. Demonstrating this is difficult, since it is not clear what algorithms are most efficient for intermediate values of p.

Using a branch-price-and-cut algorithm we show that the lowest computation times correspond to intermediate values of p for 4 out of 22 considered benchmark instances from the literature. To strengthen this result, we provide new instances constructed in such a way that intermediate values of p consistently outperform the arc-based and path-based formulation. Although these results are dependent on the used algorithm and instances, they provide evidence that there is potential for computational gains using the p-step formulation. We believe that our results merit further research into new algorithms to pursue this potential.

We summarize our contributions as follows. We introduce a new family of formulations that generalize the traditional arc-based and path-based formulation. We prove that computing the set partitioning bound is NP-hard, and use this result in conjunction with our p-step formulation to additionally show that there does not exist a strongest compact formulation. This way, we conclude the research in pursuit of the strongest compact formulation, but we also point out the relevant research direction of finding the strongest formulation with a limited number of variables and constraints. We provide the new strongest known formulation with a polynomial number of variables of fixed degree three and higher. Finally, we provide computational results which suggest that there is potential for computational gains of using a p-step formulation.

This paper is organized as follows. In Section 2, we present the p-step formulation. In Section 3, we analyze the LP bound of the p-step formulation. In Section 4, we present the algorithm that we use in our computational experiments. In Section 5, we present improvements to the p-step formulation which in conjunction with our algorithm serve to strengthen the LP bounds and simultaneously provide computational gains. In this section we also point out new strongest known formulations with the number of variables and constraints of fixed polynomial degree three and higher. In Section 6 we provide the results of our computational experiments. Finally, we conclude the paper in Section 7.

2 A p-step formulation

Even though the CVRP is a well known problem, we provide a description in Section 2.1 both for the sake of completeness and to introduce the notation that we use in this paper. Furthermore, in Section 2.2 we define p-steps and provide a new binary programming formulation of the CVRP which makes use of these p-steps. We describe in Section 2.3 how a compact version of this formulation is obtained.

2.1 The capacitated vehicle routing problem

Consider a directed graph $G = (N, A)$. Here $N = \{0, \ldots, n+1\}$ is a set of locations such that 0 represents the starting depot, $n+1$ the ending depot and $N' = \{1, ..., n\}$ is the set of customers, and $A = \{(i,j) : i,j \in N, i \neq j, i \neq n+1, j \neq 0\}$. Each customer $i \in N'$ has a demand $q_i > 0$. For ease of notation let $q_0 = q_{n+1} = 0$. An unlimited amount of vehicles of capacity Q is available for satisfying demand. Vehicles traverse an elementary path from 0 to $n+1$, referred to as a route, to satisfy demand of all customers along the path. The total demand of the customers on a single route cannot exceed the capacity of a vehicle, and demand cannot be split, i.e., every customer is visited exactly once. We assume $q_i \leq Q$ for all $i \in N$. The cost of traversing an arc $(i,j) \in A$ is $c_{ij} \geq 0$. The capacitated vehicle routing problem (CVRP) is the problem of finding routes to satisfy all customer demands while the total costs are minimized.

2.2 Binary programming formulation

Next, we present a new binary programming formulation of the CVRP. Let a p-step r be a pair (P_r, d_r). Here P_r is an elementary path in G that 1) either traverses exactly p arcs, thus visiting $p+1$ nodes, or 2) starts at 0 and traverses at most p arcs. Furthermore, d_r represents the cumulative demand of the customers on a route prior to arriving at the first location on P_r. We refer to d_r as the prior demand of r and we denote by $q(r)$ the total demand of the customers visited on the p-step r. The p-step is called feasible if $0 \leq d_r + q(r) \leq Q$ and we denote by R^p the collection of all feasible p-steps.

In our formulation, we concatenate p-steps to represent routes. Two p-steps r and s can be concatenated if and only if the last location $i \in N'$ of r is the first location of s, and the total demand after leaving i on r matches that of s, that is, $d_r + q(r) = d_s + q_i$. Observe that every route can be represented as a concatenation of feasible p-steps. Moreover, in our formulation we enforce that a customer can only be visited once. This way, we ensure in our formulation that any concatenation resulting in a path from 0 to $n+1$, does not only satisfy the capacity constraint, but is also elementary and therefore represents a route.

We introduce additional parameters which we use in our formulation. Let a_r^i be the degree of customer i on the path P_r corresponding to p-step r. That is, a_r^i is 2 if customer $i \in N'$ is visited once by p-step r, unless i is the first or last node on P_r in which case a_r^i is 1, and 0 otherwise. To represent the concatenation of p-steps in our formulation, we introduce e_r^i and q_r^i. Let e_r^i be 1 if i is the first location on P_r, -1 if i is the last location on P_r, and 0 otherwise. Furthermore, let q_r^i be $d_r + q_i$ if i is the first location on P_r, $-d_r - q(r)$ if i is the last location on P_r, and 0 otherwise.

Finally, let the binary p-step variable x_r indicate whether p-step r is used. The CVRP can be formulated as the following binary program.

$$\min \sum_{r \in R^p} c_r x_r \tag{1}$$

$$\sum_{r \in R^p} a_r^i x_r = 2 \qquad \forall i \in N' \tag{2}$$

$$\sum_{r \in R^p} e_r^i x_r = 0 \qquad \forall i \in N' \tag{3}$$

$$\sum_{r \in R^p} q_r^i x_r = 0 \qquad \forall i \in N' \tag{4}$$

$$x_r \in \{0,1\} \qquad \forall r \in R^p \tag{5}$$

The total costs are minimized in the objective function (1). Constraints (2) ensure that every customer is visited exactly once. Constraints (3) and (4) ensure that if a p-step is selected that ends at node i then also a p-step is selected that starts at node i with sufficient prior demand. Constraints (5) represent the binarity conditions on the p-step variables x. We refer to formulation (1)-(5) as the p-step formulation.

2.3 Compactness

Because the prior demand d_r is continuous, the number of p-steps in R^p is infinite and by extension so are the number of variables x_r, unless $Q = q_i$ for all $i \in N'$. However, we can redefine the p-step formulation to only include a finite number of p-steps. This is achieved by only considering p-steps for which the prior demand is minimal or maximal, so that any other p-step can be represented as a convex combination of the extremes. To make this more precise, first observe that the p-step formulation remains valid when replacing the equalities (4) by inequalities as follows

$$\sum_{r \in R^p} q_r^i x_r \geq 0 \quad \forall i \in N'. \tag{6}$$

In this case, a strict inequality of (6) can be interpreted as a vehicle carrying empty space which is erroneously considered as prior demand by some p-step. Moreover, the LP bounds do not change, since we can lower the erroneous prior demands to obtain a new solution in which (6) are satisfied with equality, without changing the objective value.

Next, we replace the binarity conditions (5) on the p-step variables by binarity conditions on the arc-flow, as is standard practice when using a set partitioning formulation for the CVRP. To make this explicit, let the binary arc-flow variable θ_{ij} indicate whether arc $(i,j) \in A$ is used. Also, let the binary parameter b^r_{ij} indicate whether arc $(i,j) \in A$ is used by p-step r. The binarity conditions are replaced by

$$\sum_{r \in R^p} b^r_{ij} x_r = \theta_{ij} \qquad \forall (i,j) \in A \tag{7}$$

$$x_r \geq 0 \qquad \forall r \in R^p \tag{8}$$

$$\theta_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{9}$$

Constraints (7) and (9) allow a single route in a solution to be represented by a convex combination of several routes, only if every arc in A is selected binarily. This enables us to drop the binarity conditions on the p-step variables as is done in (8). Note that it is not even necessary to impose $x_r \leq 1$ for all $r \in R^p$, since this is implied by (2), (3) and (8). We are now ready to also limit the number of p-step variables and obtain a formulation with a finite number of variables.

Proposition 1. *The p-step formulation* (1)-(5) *has the same optimal solution value and LP bound as* (1)-(3) *and* (6)-(9) *while limiting R^p to include only:*

1. *All p-steps r that start at 0 with prior demand $d_r = 0$.*
2. *All p-steps r that do not start at 0 nor end at $n+1$ with prior demand $d_r = 0$.*
3. *All p-steps r that do not start at 0 nor end at $n+1$ with prior demand $d_r = Q - q(r)$.*
4. *All p-steps r that do not start at 0 and end at $n+1$ with prior demand $d_r = Q - q(r)$.*

Proof. It is standard to replace the binarity conditions on the route variables by binarity conditions on the arc-flow. This does not affect the optimal solution value nor the LP bound. Furthermore, consider a solution of the p-step formulation, or its LP relaxation, using p-step r. If r starts at 0 with $d_r > 0$, it could be replaced by p-step $r' = (P_r, 0)$, without violating any of the constraints, in particular not (6), and without changing the cost. Similarly, if r ends at $n+1$ while $d_r < Q - q(r)$, it can be replaced by $r' = (P_r, Q - q(r))$. Finally, if r does not start at 0 nor ends at $n+1$ and has prior demand $0 < d_r < Q - q(r)$, it can be represented by a convex combination of $r_1 = (P_r, 0)$ and $r_2 = (P_r, Q - q(r))$. $\square$

Applying Proposition 1 yields a p-step formulation with a finite number of variables. In fact, it is a compact formulation of the CVRP, with the number of variables being of order n^{p+1}.

3 LP bounds

In Section 3.1, we describe the progression of LP bounds with p. That is, we compare the LP bounds of the p-step formulation for different values of p. We then show that computing the set partitioning bound is NP-hard in Section 3.2, and use that result to prove that there does not exist a strongest compact formulation if $P \neq NP$. Finally, we compare the LP bounds to those of other formulations of the CVRP in Section 3.3.

3.1 Progression of LP bounds

The p-step formulation is properly defined for $p \in \mathbb{N}_{>0}$, i.e., for all $p \in \mathbb{N}$, $p \geq 1$. Denote by z_p the LP bound of the p-step formulation for a particular choice of p. Observe that for an instance with n customers, it holds for $p \geq n+1$ that $R^p = R^{n+1}$ and as a result $z_p = z_{n+1}$ for all $p \geq n+1$. Note that z_{n+1} is the set partitioning bound. Next, we analyze how the LP bounds of the p-step formulation progress when p increases from 1 to $n+1$. It turns out that the LP bound does not necessarily increase monotonically in p, although a gradual increase does occur. In what follows, we make this statement more precise.

Proposition 2. *For any $p, m \in \mathbb{N}_{>0}$ it follows that $z_p \leq z_{pm}$.*

Proof. Consider an optimal solution x^* to the LP relaxation using pm-steps. Any given pm-step r can be cut into k separate p-steps $r_1, \ldots, r_k$, ending at locations $n_1, \ldots, n_k$ respectively, where k is at most m. The prior demands d_{r_i} are straightforwardly chosen as $d_{r_i} = d_r + \sum_{j=1}^{i-1} \left(q(r_j) - q_{n_j}\right)$. We define $x'_{r_i} = x^*_r$ for all $1 \leq i \leq k$. It is easily verified that x' is a feasible solution using exclusively p-steps, while maintaining the same objective value. This shows that $z_p \leq z_{pm}$. □

Using Proposition 2, it follows immediately that any p-step formulation is at least as strong as the arc-based formulation obtained for $p = 1$.

Corollary 3. *For any $p \in \mathbb{N}_{>0}$, it holds that $z_1 \leq z_p$.*

Furthermore, no p-step formulation is stronger than the set partitioning formulation obtained for $p = n+1$.

Corollary 4. *For any $p \in \mathbb{N}_{>0}$, it holds that $z_p \leq z_{n+1}$.*

Proof. Since for any integer $k \geq n+1$ it holds that $z_k = z_{n+1}$, by Proposition 2 it follows that $z_p \leq z_{(n+1)p} = z_{n+1}$. □

When comparing a p-step formulation with a q-step formulation, for $p < q$, the LP bound is in general only non-decreasing when q is an integer multiple of p, as described in Proposition 2. Otherwise, it can occur that the LP bound decreases.

Proposition 5. *For any $p, q \in \mathbb{N}_{>0}$ such that $p < q$ and q is not an integer multiple of p, there exists an instance of the CVRP for which $z_p > z_q$.*

The proof of Proposition 5 can be found in Appendix A. In this proof, we construct an instance for which $z_p > z_q$. Here we exploit the following reason which we found for LP bounds to decrease while increasing p to q. Suppose customers appear close together in clusters, while the clusters and the depot are far apart. Consider for example clusters of two customers, then any 2-step will cover at most two arcs of which at least one is long. In contrast, 3-steps exist that have exactly one long arc out of three. At least half of the arcs selected in a solution to the LP relaxation of the 2-step formulation will be long, while a solution using 3-steps might be constructed that uses a lower ratio of long arcs. As a result, there are instances for which $z_2 > z_3$. In Appendix A, we construct such an instance for all $p, q \in \mathbb{N}_{>0}$ such that $p < q$ and q is not an integer multiple of p. Note that we begin Appendix A by first showing another result, namely that the flow out of and into the depot are both at least $\frac{1}{Q}\sum_{i\in N'} q_i$. This is similar to what Gouveia (1995) showed for the single-commodity flow formulation of the CVRP. We use this result to prove Proposition 5.

For completeness sake, we provide the following two propositions. They show that for any p and q such that $p < q$ there exists an instance in which the LP bounds are equal, and an instance in which the LP bound is higher for q.

Proposition 6. *For any $p, q \in \mathbb{N}_{>0}$ such that $p < q$, there exists an instance of the CVRP for which $z_p = z_q$.*

Proof. Consider any instance in which the demand of every individual customer equals the vehicle capacity. It follows that $z_p = z_q$. □

Proposition 7. *For any $p, q \in \mathbb{N}_{>0}$ such that $p < q$, there exists an instance of the CVRP for which $z_p < z_q$.*

The proof of Proposition 7 can be found in Appendix B.

We now have a full characterization of the LP bounds for varying values of p. The LP bound z_p increases from z_1 to z_{n+1} when p increases, although not monotonically. A decrease is potentially, but not necessarily, observed when increasing p to a value which is not an integer multiple of p. For increases of p to integer multiples of p, a decrease never occurs.

3.2 A strongest compact formulation

Consider a p-step formulation for any fixed value of p. On the one hand, the p-step formulation is weaker than the set partitioning formulation by Corollary 4 and Proposition 7. On the other hand, the p-step formulation does provide the set partitioning bound for all instances with the number of customers strictly less than p. Therefore, for a sufficiently large but fixed p, the p-step formulation is a compact formulation which is practically indistinguishable from the non-compact set partitioning formulation. Next, we discuss the implications of this observation in the context of finding a strongest compact formulation. We first show that computing the LP bound of the set partitioning formulation of the CVRP is NP-hard. We then show that it follows that there does not exist a strongest compact formulation, if $P \neq NP$, since for any compact formulation a p-step formulation can be found that is not weaker.

Proposition 8. *Computing the set partitioning bound of the CVRP is NP-hard.*

The proof of Proposition 8 can be found in Appendix C. It is a polynomial time reduction of the partition problem. Although the partition problem is NP-hard (Karp 1972), a pseudo-polynomial time algorithm does exist (Garey and Johnson 1979). Therefore, it is still an open question whether computing the set partitioning bound of the CVRP is strongly NP-hard, or whether a pseudo-polynomial time algorithm exists.

Since the same polynomial time reduction can be used for any lower bound on the CVRP that is at least as strong as the set partitioning bound, the next corollary follows, which is used to prove that no strongest compact formulation exists for the CVRP, if $P \neq NP$.

Corollary 9. *No polynomial time algorithm exists that provides a lower bound for the CVRP which is at least as strong as the set partitioning bound, unless* $P = NP$.

Proposition 10. *If* $P \neq NP$, *there does not exist a strongest compact formulation of the CVRP.*

Proof. A compact formulation of the CVRP can be used to compute a lower bound for the CVRP in polynomial time. By Corollary 9, unless $P = NP$, no compact formulation exists that provides at least the set partitioning bound. Hence, for any given compact formulation, at least one instance of the CVRP exists for which the corresponding LP bound is strictly less than the set partitioning bound. Let n_I be the number of customers of such an instance I.

Consider a p-step formulation, with p fixed such that $p \geq n_I + 1$. This formulation provides the set partitioning bound for instance I. Hence, it is a compact formulation that is not weaker. Since this can be done for any compact formulation, we conclude that if $P \neq NP$, there does not exist a strongest compact formulation. □

The consequence of Proposition 10 is that the pursuit of a strongest compact formulation for the CVRP is over. Moreover, it shows that the usefulness of a strong compact formulation might be limited. Although compact formulations typically exhibit weaker LP bounds than non-compact formulations, their practical use stems from being able to compute the LP bounds in polynomial time. If the number of variables and constraints is large, and polynomial time is therefore still very long, this advantage disappears in practice. This is clearly the case for a p-step formulation with a fixed but large value of p. Hence, classifying formulations on whether they have a polynomial number of constraints and variables may not be sufficient for practical purposes. Instead, we think that it is better to classify formulations according to the number of variables and constraints being polynomial of fixed degree, and pursue the strongest formulation within these classes. Obviously, we should then focus on those classes of formulations for which efficient algorithms can be devised.

3.3 Comparison with other formulations

Fixing the value of p provides a p-step formulation that can be directly compared to other formulations of the CVRP. In this section, we discuss such comparisons in particular for formulations with a quadratic and cubic number of variables and constraints, and comment briefly on formulations with a higher number of variables and constraints.

Let us first consider the 1-step formulation, by comparing it to the single-commodity flow formulation (SCF) attributed to Gavish (1984). Both formulations have a quadratic number of variables and constraints.

Proposition 11. *The* 1*-step formulation is equally strong as SCF.*

The proof of Proposition 11 can be found in Appendix D, along with SCF for ease of reference. There we first provide the somewhat stronger result that actually for any choice of p, the p-step formulation is at least as strong as SCF.

For more comparisons with the 1-step formulation, we refer to Letchford and Salazar-González (2015), where a detailed comparison is presented of a selection of formulations of the CVRP which includes SCF. These comparisons immediately extend to the 1-step formulation.

It is worth noting that the compact 1-step formulation provided by (1)-(3) and (6)-(9), uses $n^2 - n$ more continuous variables and $n^2 + n$ less constraints than SCF, if we do not consider non-negativity of variables as constraints. Moreover, to compute just the LP bound using the 1-step formulation, the arc-flow variables and corresponding constraints do not need to be considered. The LP bound is obtained by solving (1)-(3), (6) and (8). This means that the LP bound can be computed with the 1-step formulation using $2n$ less variables and $2n^2 + 2n$ less constraints than SCF. In particular, the number of constraints becomes linear instead of quadratic, if we do not consider non-negativity of variables as constraints.

Next, we consider formulations of the CVRP with a cubic number of variables. In particular, we consider the formulation by Leggieri and Haouari (2017), which they refer to as F3. To the best of our knowledge, F3 was previously the strongest known compact formulation. This formulation not only has a cubic number of variables but also a cubic number of constraints. To be precise, it has $2n^3 + \mathcal{O}(n^2)$ variables and $6n^3 + \mathcal{O}(n^2)$ constraints. Given the extensive nature of F3, for the sake of brevity we substantiate our claims using empirical evidence only.

For now, we leave the question open of how the 2-step formulation compares to F3. We come back to this question in Section 5.4, where we compare a stronger 2-step formulation with F3 and show that it is not weaker. This makes the comparison with the 2-step formulation less relevant and we do not pursue it further.

Instead, consider the 3-step formulation. Although it has a quartic number of variables instead of cubic like F3, it is still noteworthy that it is not weaker than F3. Consider for example the instance E-n13-k4 introduced by Christofides and Eilon (1969). The LP bound of this instance provided by F3 is 240.972, while $z_3 = 241.648$. This is noteworthy because F3 was, to the best of our knowledge, the strongest compact formulation. Therefore, it was also the strongest formulation with at most a quartic number of variables. However, the combination of F3 and the 3-step formulation is even stronger. Clearly, it is not weaker than both F3 and the 3-step formulation separately, and it is in fact stronger than F3 as can be observed from the LP bound 241.926 for instance E-n13-k4. By using even larger values of p, even stronger formulations can be found.

4 Algorithm

We present an algorithm to compute the LP bound of the p-step formulation and embed this in a branch-and-bound algorithm to solve the CVRP. The purpose is to experimentally compare the computation times of using the p-step formulation for different values of p. To facilitate this comparison, we think it is best to use a single algorithm for all values of p. Of course, different algorithms might be better suited for specific values of p.

As the number of variables grows large when p increases, we use a column generation algorithm to compute the LP bounds. Column generation is an algorithm in which initially only a limited number of variables are included in the formulation, yielding the so called restricted master problem (RMP). The RMP is solved and next negative reduced cost variables are searched for, by solving a pricing problem. The algorithm proceeds iteratively. At each iteration, either at least one negative reduced cost variable is added to the RMP, or it is concluded that no negative reduced cost variables exist. In the latter case the solution to the current RMP is the optimal solution to the LP relaxation.

In our column generation algorithm, we use the compact formulation (1)-(3) and (6)-(9). In the RMP, we include all variables θ but we initially limit the number of p-step variables x. In Section 4.1, we describe how we model the resulting pricing problem and in Section 4.2 we provide an exact algorithm to solve this pricing problem. Furthermore, in Section 4.3 we describe a preprocessing procedure based on the p-step formulation to aid our exact pricing algorithm. In Section 4.4 we show how additional computational gains can be achieved when dealing with symmetric CVRP instances. This is relevant since many of the benchmark instances used to compare algorithms in the scientific literature are actually symmetric. In Section 4.5, we present our branch-price-and-cut algorithm in which we embed our column generation algorithm. Finally, in Section 4.6 we point out some differences between our algorithm and the fastest algorithms for the CVRP, and provide arguments for our choices in algorithm design.

4.1 Pricing problem

Let λ, μ, ν and π be the dual variables associated with constraints (2), (3), (6) and (7) respectively. For ease of notation, let $\lambda_0 = \lambda_{n+1} = \mu_0 = \mu_{n+1} = \nu_0 = \nu_{n+1} = 0$. Consider the p-step (P_r, d_r) such that s is the start and f is the final node on P_r. The reduced cost $RC(r)$ of the variable x_r is

$$RC(r) = \sum_{(i,j)\in P_r} (c_{ij} - \pi_{ij} - 2\lambda_j) - \lambda_s + \lambda_f - \mu_s + \mu_f - \nu_s\,(d_r + q_s) + \nu_f\,(d_r + q(r))\,.$$

The pricing problem is to find a feasible p-step, for which the reduced cost is minimal. Recall that a feasible p-step consists of an elementary path P_r of length at most p if $s = 0$ or exactly p if $s \neq 0$, and prior demand $0 \leq d_r \leq Q - q(r)$.

To the best of our knowledge, this problem has not been studied before. However, next we describe how to decompose the pricing problem to obtain a polynomial number of more familiar problems. Consider the decomposition into separate problems with fixed start node s and fixed final node $f \neq s$, which we also refer to as pricing problems, or (s, f) pricing problems. Note that we need not consider $n+1$ as a starting node nor 0 as a final node. This way, we obtain $(n+1)n$ pricing problems.

For each (s, f) pricing problem, it can be determined immediately whether the minimal or maximal value should be assigned to the prior demand. Observe that in the expression for the reduced cost of x_r, d_r has linear coefficient $\nu_f - \nu_s$. Hence, if $\nu_f \geq \nu_s$ it is optimal to assign d_r the minimal value 0, otherwise it is optimal to assign d_r the maximal value $Q - q(r)$.

Given the optimal assignment of the prior demand, we model each (s, f) pricing problem as an elementary shortest path problem with a capacity constraint and a requirement on the path length. To do so, we compute an appropriate constant cost and we assign costs to each arc in A, such that the total cost of an (s, f) path in G is the reduced cost of a corresponding p-step variable. These costs are dependent on whether the minimal or maximal value should be assigned to the prior demand.

If the minimal value is assigned, the constant cost is $C_{sf} = -\lambda_s + \lambda_f - \mu_s + \mu_f + (\nu_f - \nu_s)\, q_s$. To each arc (i, j) we assign the cost $c'_{ij} = c_{ij} - \pi_{ij} - 2\lambda_j + \nu_f q_j$. For the maximal value, the constant cost is $C_{sf} = -\lambda_s + \lambda_f - \mu_s + \mu_f + (\nu_f - \nu_s)\, Q$. To each arc (i, j) we assign the cost $c'_{ij} = c_{ij} - \pi_{ij} - 2\lambda_j - \nu_s q_j$. The reduced cost of the variable corresponding to a path P_r and prior demand d_r are the sum of the arc costs and the constant cost C_{sf}.

A path is considered feasible if the demand of all customers on the path does not exceed the capacity Q, and if $s = 0$ the path consists of at most p arcs, and if $s \neq 0$, the path consists of exactly p arcs. After selecting the optimal choices of prior demand, each of the $(n+1)n$ pricing problems is now the problem of finding a feasible elementary path in G that minimizes the reduced costs.

4.2 Exact pricing algorithm

Observe that the pricing problem is polynomially solvable, since the number of p-step variables is of order n^{p+1}. Hence, the pricing problem could for instance be solved in polynomial time by enumerating all p-steps. However, the number of p-step variables is exponential in p, which means that as p increases such an approach quickly becomes intractable.

Instead, we solve the (s, f) pricing problems using a bidirectional labeling algorithm (Righini and Salani 2006) with decremental state space relaxation (Righini and Salani 2008) and completion bounds. The complete algorithm resembles the algorithm of Martinelli et al. (2014) which is designed to solve elementary shortest path problems with a capacity constraint. In Appendix E, we describe the algorithm in more detail in order to highlight the slight modifications we made to deal with the additional path length limitation, and to discuss some minor implementation details for the sake of reproducibility. We make use of the unreachability vector proposed by Feillet et al. (2004) in our representation of a label. Furthermore, we make use of ng-paths introduced by Baldacci et al. (2011) to allow cycles in the decremental state space relaxation. Note that this still means we solve the elementary shortest path problem and, although possible, in our experiments we do not apply ng-path relaxation to the formulation. Finally, note that the (s, f)-pricing problems can be solved in parallel.

4.3 Preprocessing

Next, we present preprocessing techniques that we use to eliminate some redundant (s, f)-pricing problems and to reduce the size of others. Recall that a p-step (P_r, d_r) is feasible if $0 \leq d_r \leq Q - q(r)$. Therefore, if for all p-steps r for which P_r is an (s, f)-path it holds that $Q - q(r) < 0$, there does not exist such a feasible p-step. In this case we do not need to consider the (s, f)-pricing problem. For a subset $Y \subseteq N$ and $y \in \mathbb{N}_{>0}$, we define $Q(Y, y) = Q - \min\{\sum_{k \in S} q_k : S \subseteq N' \backslash Y, |S| = y\}$ as the remaining capacity of a vehicle which also visits the y customers with lowest demand, excluding customers in Y. By the preceding argument, we do not consider the (s, f)-pricing problem if $s \neq 0$ and $q_s + q_f > Q(\{s, f\}, p - 1)$. Observe that this criterion can be checked in a preprocessing phase.

Note that if there are no feasible p-steps departing from a customer, then all feasible p-steps arriving at this customer can be removed from the formulation as well, since they can never be part of a feasible solution. Similarly, all p-steps departing from a customer can be removed if no feasible p-steps arrive there. In this case, we remove the corresponding (s, f) pricing problems during preprocessing.

In our implementation, we actually introduce a copy G_{sf} of the graph G for each (s, f)-pricing problem and subsequently solve the pricing problem on G_{sf} instead. We eliminate nodes and arcs from G_{sf} as follows. For $i \in N'$, such that $s \neq 0$, i and f are all different, suppose that $p \geq 2$ and that $q_s + q_i + q_f > Q(\{s, i, f\}, p - 2)$. In this case node i can be removed from G_{sf} because no feasible p-step exists that visits s, i and f. Similarly, for $(i, j) \in A$, such that $s \neq 0$, i, j and f are all different, if $p \geq 3$ and $q_s + q_i + q_j + q_f > Q(\{s, i, j, f\}, p - 3)$ then arc (i, j) can be removed from G_{sf}. Observe that also the removal of nodes and arcs can be done in a preprocessing phase.

Observe that, for $p \geq 2$, $q_s + q_f > Q(\{s, f\}, p-1)$ implies $q_s + q_i + q_f > Q(\{s, i, f\}, p-2)$ for all $i \in N'$. Furthermore, for $p \geq 3$, $q_s + q_i + q_f > Q(\{s, i, f\}, p - 2)$ implies $q_s + q_i + q_j + q_f > Q(\{s, i, j, f\}, p - 3)$ for all $(i, j) \in A$ and $(j, i) \in A$. As a result, preprocessing might be most efficient if first (s, f)-pricing problems are eliminated, then nodes are removed and finally arcs are removed.

4.4 Symmetric CVRP

The symmetric CVRP is a subclass of CVRP in which the costs to traverse an arc are symmetric, i.e, $c_{ij} = c_{ji}$ for all $(i, j) \in A$. Next, we illustrate that for the symmetric CVRP we only need to consider one pricing problem of every pair consisting of the (s, f) pricing problem and the (f, s) pricing problem. Consider a p-step (P_r, d_r) such that P_r starts at s and ends at f. Also consider a p-step $(P_{r'}, d_{r'})$, such that $P_{r'}$ is the reverse of P_r. Assume without loss of generality that $\nu_f \geq \nu_s$. It follows that the p-steps r and r' have the lowest reduced costs when $d_r = 0$ and $d_{r'} = Q - q(r)$. The difference in reduced costs is the following:

$$RC(r) - RC(r') = -2\mu_s + 2\mu_f - \nu_s q_s + \nu_f q_f + (\nu_f - \nu_s)\, Q$$

If $RC(r) - RC(r') \leq 0$, we conclude that the (s, f) pricing problem yields a p-step with lower reduced costs than the (f, s) pricing problem, otherwise the (f, s) pricing problem yields a p-step with lower reduced costs. Therefore, when solving an instance of the symmetric CVRP, in our algorithm we only consider the most promising of the two pricing problems. This halves the number of pricing problems to be considered of those which start and end in N'.

4.5 Branch-price-and-cut

Next, we summarize the branch-price-and-cut algorithm which we use in our experiments. The details can be found in Appendix F. Our algorithm resembles any common branch-price-and-cut algorithm for vehicle routing problems, see Costa et al. (2019). We use a column generation algorithm to compute the LP bound at each node in the branching tree. We strengthen the bounds by adding rounded capacity, framed capacity, homogeneous multistar, strengthened comb and 2 edge hypotour inequalities using the package of heuristic separation algorithms of Lysgaard (2003), see also Lysgaard et al. (2004). To branch, we likewise use a procedure in the package of Lysgaard (2003).

4.6 Remarks on our algorithm

For our experiments, it seems sensible to use as much as possible conventional, tried-and-tested methodology developed for the CVRP. However, we omit some techniques that are commonly part of the fastest algorithm.

The algorithm by Fukasawa et al. (2006) dynamically decides between using an arc-based and path-based formulation, and numerical experiments show that depending on the instance either can result in the fastest overall computation time. Since the work of Fukasawa et al. (2006), lower computation times have been predominantly achieved using algorithms that require strong lower bounds. Strong lower bounds can be exploited by arc fixing (Irnich et al. 2010) and route enumeration (Baldacci et al. 2008), which also require a strong upper bound. To acquire strong lower bounds, the set partitioning bound has been further strengthened using subset row inequalities (Jepsen et al. 2008), or the more computationally appealing limited memory subset row inequalities (Pecin et al. 2017). Also the ng-route relaxation has been introduced by Baldacci et al. (2011) allowing for significantly faster computation while not severely weakening the LP bounds. These techniques are combined in fast algorithms like that of Pecin et al. (2017).

We do not use any route relaxation, because in our experiments we particularly want to demonstrate the differences between the p-step formulation for different values of p, and we do not want to obscure the results by using cyclic paths instead of elementary paths. Note that preliminary experiments with ng-route relaxations suggest typical behavior, i.e., lower computation times of the LP bounds and a small decrease of these LP bounds.

We also do not use so-called non-robust cuts, like the subset row inequalities (Jepsen et al. 2008). These are valid inequalities for the set partitioning formulation which alter the structure of the pricing problem. Unlike the robust cuts used in our algorithm, they cannot be immediately applied for the p-step formulation. However, this might be done by a careful modification of the inequalities, and we leave this for future investigation.

Arc fixing and route enumeration are most successful when good upper and lower bounds are available. In the context of a p-step formulation, we point out two counteracting effects. On the one hand, for smaller values of p the bounds are lower, making these strategies less successful. On the other hand, particularly route enumeration suffers from lengthy paths used in the formulation. When considering the p-step formulation, only paths of a limited length need to be enumerated, which could yield a computational advantage. Further investigation is required which we consider beyond the scope of this paper.

Observe that arc-based and path-based formulations have been the object of study for decades. A competitive algorithm using a p-step formulation might require a similar amount of effort into algorithmic development. It follows that our branch-price-and-cut algorithm is not competitive in terms of computation times with the fastest algorithms reported in the literature. Our purpose is to investigate whether there is sufficient potential for computational advantages, to merit further algorithmic development.

5 Improvements

In this section we present ways of strengthening the p-step formulation without increasing the number of variables and constraints. In Section 5.1, we explain that the well-known 2-cycle elimination constraints can be included by actually decreasing the number of variables and constraints. In Section 5.2, we show how the domain of the prior demand d_r of a feasible p-step can be limited to strengthen the formulation, and to potentially reduce the number of variables. Furthermore, in Section 5.3, we provide a specialized version of limiting the domain of the prior demand, which allows for computing the LP bounds using our column generation algorithm. In Section 5.4, we investigate the LP bounds of the improved p-step formulations and compare the improved 2-step formulation with F3. Finally, in Section 5.5 we provide new strongest known formulations with the number of variables and constraints limited by a polynomial of fixed degree three and higher.

5.1 2-cycle elimination

We next explain how the inclusion of 2-cycle elimination constraints is achieved by reducing the number of variables and constraints. We replace the variables θ_{ij} for the directed set of arcs A by their undirected counterparts, corresponding to the set of edges E. This limits the number of variables θ_{ij} and the number of constraints (7). Technically, this also requires a redefinition of b_{ij}^r. Now, for $(i,j) \in E$, let b_{ij}^r be 1 if arc $(i,j) \in A$ or $(j,i) \in A$ is used by p-step r. Observe that we do not redefine the p-steps, they still correspond to paths using the directed arcs A. The new undirected p-step formulation is given by (1)-(3), (6), (8) and

$$\sum_{r \in R^p} b_{ij}^r x_r = \theta_{ij} \qquad \forall (i,j) \in E \tag{10}$$

$$\theta_{ij} \in \{0,1\} \qquad \forall (i,j) \in E. \tag{11}$$

Although this redefinition of variables is common when working with formulations of the symmetric CVRP, it might not be obvious that it is valid in the asymmetric case. Therefore, we state this as a proposition for which we provide a proof in Appendix G.

Proposition 12. *The undirected p-step formulation is valid for the CVRP.*

Observe that the undirected p-step formulation is also stronger than the directed version provided by (1)-(3), (6)-(9). By (10) and (11) we implicitly impose the 2-cycle elimination constraints $\theta_{ij} + \theta_{ji} \leq 1$, for $i, j \in N'$, which are not implied by the directed p-step formulation.

5.2 A strong p-step formulation

The p-step formulation can be further strengthened by making use of the following observation. Note that any p-step that does not end at the depot, needs to be followed by at least one more p-step. Using this observation, we can identify p-steps that can never be selected in a solution. Indeed, if the total load of a p-step r is too high, visiting $p-1$ customers after visiting the last customer on r would result in violating the vehicle capacity. Therefore, although such a p-step r might be selected in a solution to the LP relaxation, it is not required for a p-step formulation to be valid. As a consequence, we can restrict the domain of the prior demand d_r corresponding to feasible p-steps, yielding a stronger formulation.

To do so, we make use of the remaining capacity function $Q(Y, y)$, as defined in Section 4.3. Like R^p, let the set of feasible p-steps R^p_{Strong} include all p-steps r ending at the depot for which the corresponding prior demand d_r satisfies $0 \leq d_r \leq Q - q(r)$. Moreover, for any p-step r which visits locations L and does not end at the depot, the prior demand d_r satisfies $0 \leq d_r \leq Q(L, p-1) - q(r)$.

Clearly $R^p_{\text{Strong}} \subseteq R^p$, if we ignore the compactification of R^p described in Proposition 1. Hence, using R^p_{Strong} instead of R^p in the p-step formulation (1)-(5) yields a non-weaker formulation. In Section 6, we show that the strengthened p-step formulation is really stronger, and not equally strong, by providing examples in which the LP bounds are strictly better. However, our column generation algorithm is not suited to compute these LP bounds. Instead, we compute the bounds as follows. Note that like in Proposition 1, a compact formulation can also be obtained using R^p_{Strong}. In our numerical experiments we generate a priori all p-steps in R^p_{Strong} required for constructing this compact formulation, and solve the resulting linear programming problem. This is only tractable for small values of p.

Note that for an instance in which all customer demands fit in one vehicle, i.e., $\sum_{i \in N'} q_i \leq Q$, it holds that $R^p = R^p_{\text{Strong}}$, so all LP bounds are the same. In this case, the instance can be modified without changing the solution, by setting $Q = n$ and $q_i = 1$ for all $i \in N'$. Defining R^p_{Strong} as if dealing with the modified instance, yields a stronger formulation than R^p also for these instances.

5.3 A specialized p-step formulation

We present a third set of p-steps which includes more p-steps than R^p_{Strong} but less than R^p. We introduce this specialized set to provide stronger LP bounds than those using R^p, while simultaneously attempting to speed up the column generation algorithm. That is, it is a specialized version specifically designed in conjunction with our column generation algorithm.

Let R^p_{CG} be a set of feasible p-steps defined similar to R^p_{Strong}. Like R^p, it also includes all p-steps r which end at the depot and for which the prior demand d_r satisfies $0 \leq d_r \leq Q - q(r)$. Moreover, for any other p-step r, starting at $s \in N$ and ending at $f \in N'$, the prior demand d_r satisfies $0 \leq d_r \leq Q(\{s, f\}, p-1) - q(r)$.

It follows that $R^p_{\text{Strong}} \subseteq R^p_{\text{CG}} \subseteq R^p$ and also for R^p_{CG} a compact formulation can be obtained like in Proposition 1. To compute z^{CG}_p, we modify the column generation algorithm presented in Section 4. In particular, we modify the capacity of the (s, f) pricing problems. Instead of a capacity Q, we impose the lower capacity $Q(\{s, f\}, p-1)$, if $f \in N'$. Observe that these capacities can be computed in a preprocessing stage, and are only dependent on the starting node and final node of the path. Note that this is not possible in the case of R^p_{Strong}.

Additionally, the preprocessing procedure described in Section 4.3 is modified to take these alternate capacities into account. That is, for $s, f \in N'$ the (s, f) pricing problem is now eliminated if $q_s + q_f > Q(\{s, f\}, 2p-2)$. Node $i \in N'$, such that s, i and f are all different, is now removed from G_{sf} if $q_s + q_i + q_f > Q(\{s, i, f\}, 2p-3)$. Finally, arc (i, j), for $i, j \in N'$, such that is s, i, j and f are all different, is now removed from G_{sf} if $q_s + q_i + q_j + q_f > Q(\{s, i, f\}, 2p-4)$.

5.4 LP bounds of the improved formulations

Denoting the LP bounds corresponding to R^p_{Strong} and R^p_{CG} while also including the 2-cycle elimination constraints, by z^{Strong}_p and z^{CG}_p respectively, we find $z_p \leq z^{\text{CG}}_p \leq z^{\text{Strong}}_p$, since $R^p_{\text{Strong}} \subseteq R^p_{\text{CG}} \subseteq R^p$. Note that the progression of LP bounds, as described by Propositions 2 through 7 and Corollary 3, equivalently apply to z^{Strong}_p and z^{CG}_p.

Next, we revisit the comparison of p-step formulations with other formulations. When $p = 1$ or $p = n + 1$, then $R^p = R^p_{\text{Strong}} = R^p_{\text{CG}}$ and the corresponding LP bounds are the same. Hence, z^{Strong}_1 and z^{CG}_1 are equal to the LP bound of SCF with 2-cycle elimination constraints, and z^{Strong}_{n+1} and z^{CG}_{n+1} are equal to the set partitioning bound.

Recall that F3 by Leggieri and Haouari (2017) is, to the best of our knowledge, currently the strongest compact formulation with a cubic number of variables and a cubic number of constraints. We show numerically that this is no longer the case. Consider an instance with 5 customers located on a regular pentagon with sides of length 8, and a single depot at the center. Hence, rounded to the nearest integer, the distance between the depot and all other customers is 7, and the distance between two customers that are not adjacent on the pentagon is 13. All customers have demand 2, while the vehicle capacity is 7. The LP bound of this instance provided by F3 is 48.5714 while $z^{\text{CG}}_2 = z^{\text{Strong}}_2 = 50$.

Note that F3 is also not weaker. Consider for example again instance E-n13-k4 of the CVRP introduced by Christofides and Eilon (1969). The LP bound of this instance provided by F3 is 240.972, while $z_2^{\text{CG}} = z_2^{\text{Strong}} = 239.711$. So neither F3 nor the 2-step formulation dominates. Moreover, the combination of these formulations is stronger. For example, the LP bounds of the combination of F3 and the 2-step formulation using R^p_{CG} or R^p_{Strong} are both 240.988 for E-n13-k4. We conclude that the combination of F3 and the p-step formulation using R^p_{Strong} is the new strongest known compact formulation with a cubic number of variables and a cubic number of constraints.

5.5 New strongest known formulations of fixed size

We present new strongest known formulations with the number of variables and constraints limited by a polynomial of fixed degree three and higher. We make use of the p-step formulation with 2-cycle elimination and using R^p_{Strong}, with LP bound z_p^{Strong}. Consider the combination of the 1-step through p-step formulations. One way of formulating this combination is by using separate k-step variables for k is 1 through p, while using a single set of arc-flow variables θ to link them, and expressing the objective function in terms of θ.

Denoting the LP bound by z'_p it follows that $z'_p \leq z'_{p+1}$. Note that the LP bounds are monotonically increasing. Moreover, it follows that $\max_{k\in\{1,\dots,p\}}\{z_k^{\text{Strong}}\} \leq z'_p$, showing that the LP bounds of these combined formulations are at least as strong as those of the p-step formulations. Moreover, for $p \geq 2$, also combining this formulation with F3, results in the new strongest known compact formulation of the CVRP with the number of variables of order n^{p+1}, and a cubic number of constraints.

It is unclear how to efficiently compute the LP bound of this formulation. However, note that not all formulations from 1 through p are required in this combination. Consider the combination of formulations only for a set of indices I constructed as follows. Initialize $I = \emptyset$, and iterate backwards from $k = p$ to $k = 1$. Include k in I only if I contains no integer multiple of k. It follows by Proposition 2, that the LP bounds remain the same, although the size of the formulation is reduced.

6 Computational experiments

In this section, we present the results of computational experiments with the p-step formulation. In Section 6.1, we demonstrate the LP bounds of the p-step formulation using various values of p, for benchmark instances from the literature. In Section 6.2, we discuss the corresponding computation times. Finally, in Section 6.3 we present experiments in which we use our branch-price-and-cut algorithm to find an optimal solution to the CVRP.

Our algorithms are implemented in C++ and compiled in Visual Studio 2015 (Platform Toolset v140). We use CPLEX 12.8 to solve the RMP at each iteration of the column generation algorithm. Furthermore, we use CPLEX 12.8 to compute the LP bounds of the p-step formulation defined by R^p_{Strong}, since our column generation algorithm cannot be applied for this purpose. All experiments are run on an Intel(R) Xeon(R) CPU E5-1620 v3 processor, with 16 GB of RAM. In some experiments we run pricing algorithms

Table 1: LP bounds using 2-cycle elimination.

Instance	p-steps	p :	2	3	4
E-n13-k4	R^p		239.39	241.65	247.00
	R^p_{CG}		239.71	243.16	247.00
	R^p_{Strong}		239.71	243.16	247.00
E-n22-k4	R^p		350.06	354.38	359.56
	R^p_{CG}		350.52	354.49	359.73
	R^p_{Strong}		350.52	354.49	259.73
E-n23-k3	R^p		531.68	535.87	535.57
	R^p_{CG}		531.76	535.87	538.82
	R^p_{Strong}		531.76	535.87	538.82
E-n30-k3	R^p		449.71	450.60	454.73
	R^p_{CG}		449.80	451.30	454.78
	R^p_{Strong}		449.80	451.30	OoM
E-n31-k7	R^p		363.52	368.27	370.17
	R^p_{CG}		363.57	368.35	370.18
	R^p_{Strong}		363.57	368.35	370.18
E-n33-k4	R^p		785.63	793.35	798.07
	R^p_{CG}		785.81	793.50	798.18
	R^p_{Strong}		785.82	793.50	-

in parallel using up to 8 parallel threads. In each of our experiments, we use a time limit of 3600 seconds of wall clock time. We use the following values for the parameters mentioned in Appendix F: $X_{\text{Arcs}} = 10$, $X_{\text{ColAdd}} = 100$ and $X_{\text{ColPool}} = 1000$.

6.1 LP bounds

We provide a brief illustration of the LP bounds of the p-step formulation presented in this paper. Because we can include 2-cycle elimination constraints by reducing the number of variables and constraints, we do so in all experiments.

Table 1 shows the LP bounds of the p-step formulation for R^p, R^p_{CG} and R^p_{Strong}, for a selection of the instances. In Table 1, we have included some of the smaller E instances (Christofides and Eilon 1969) of the CVRP as indicated by the first column, for low values of p as indicated by the last 4 columns. This allows us to also compute the LP bounds within the time limit, specifically using R^p_{Strong}. We do not include the case $p = 1$, since the LP bounds are equal for all variants. The second column indicates which variant of the p-step formulation is considered. Note that the problem corresponding to these instances are a version of the CVRP that additionally impose that precisely k vehicles should be used. This can easily be incorporated in the formulation by enforcing $\sum_{j \in N} \theta_{0j} = k$, which does not void the theoretical analysis in this paper, nor are modifications to the algorithm necessary.

The dash in Table 1 indicates that the algorithm did not terminate within the time limit, while OoM indicates that the computer went out of memory when generating p-steps. It is clear that using R^p_{CG} provides stronger bounds than using R^p, while given the nature of our algorithm these bounds are typically also computed faster. Furthermore, only for small values of p are we able to compute the LP bounds using R^p_{Strong}. Therefore, we only consider the p-step formulation using R^p_{CG} for the remainder of the computational experiments presented in this paper.

We have included the LP bounds of this formulation for all E instances and A instances (Augerat 1995) in Appendix H. Interestingly, the LP bounds that we can compute within the time limit, are monotonically increasing in p for all 27 A instances and 10 out of 13 E instances. A decrease is observed twice for instance E-n23-k3 and once for the instances E-n31-k7 and E-n51-k5. This illustrates that although a decrease can be observed as described in Proposition 5, this is not often the case for the benchmark instances.

6.2 Computation time of column generation

Next, we demonstrate the computational performance of our column generation algorithm used to compute LP bounds. When computing the LP bound of the p-step formulation, there are two counteracting effects which affect the computation time of our pricing algorithm. As p increases, the worst-case number of computations of our labeling algorithm increases exponentially, while at the same time the number of pricing problems decreases from quadratic in the number of customers to one. Observe that when all pricing problems are solved in parallel, the worst-case computation time to solve a pricing problem is expected to increase in p. This does not mean that it should necessarily be expected that the computation time of our column generation algorithm increases with p, as other factors like the number of iterations also play a role. Next, we illustrate the computational performance of our column generation algorithm on benchmark instances.

Table 2 provides the computation times for a selection of A instances provided in the first column, and values of p provided in the last seven columns. The largest A instances are chosen because the effect of parallelization is most pronounced there. The set partitioning bound is guaranteed for $p = 17$ for instance A-n61-k9, while for the other instances this is guaranteed only for larger p. Therefore, we have chosen to present result for values of p equally spread between 1 and 16, and additionally $p = n + 1$.

The second column of Table 2 indicates the implementation. Here, sequential means that all pricing problems are solved sequentially, while 8-threads means that the pricing problems are solved in parallel using up to 8 threads. Optimistic represents a lower bound on the computation time that might have been achieved when solving all pricing problems in parallel. It is obtained by using the sequential implementation, but instead of using the cumulative computation time of the pricing problems, only the maximum at each iteration is used to compute the computation time. We consider this a lower bound on the computation time of full parallelization, as it does not include any overhead of a parallel implementation. Note that a dash indicates that the algorithm reached the time limit of 3600 seconds wall clock time. Because the optimistic implementation actually runs pricing problems sequentially, it happens that the 8-thread implementation finds a solution in time while optimistic does not.

We can see that indeed parallelization reduces computation time. Even in the optimistic case, the observed computation times do not become monotonically increasing in p for all instances, although it

Table 2: Computation times in seconds of computing LP bounds.

Instance	Implementation	p : 1	4	7	10	13	16	$n+1$
A-n61-k9	Sequential	0.35	11.70	119.23	286.23	1027.07	-	359.06
	8-threads	0.39	6.45	43.78	110.90	294.91	-	351.10
	Optimistic	0.37	4.08	13.10	75.28	125.09	-	352.63
A-n62-k8	Sequential	0.38	10.23	89.79	619.75	263.86	681.42	212.23
	8-threads	0.38	6.33	34.80	236.93	62.15	240.92	145.64
	Optimistic	0.36	3.72	7.24	12.10	15.87	69.25	209.77
A-n63-k9	Sequential	0.41	13.29	199.47	442.23	760.62	2724.58	50.85
	8-threads	0.38	6.95	52.60	122.96	177.35	771.82	41.10
	Optimistic	0.41	4.06	7.28	35.30	54.56	140.69	49.72
A-n63-k10	Sequential	0.42	10.42	200.24	260.52	1010.16	2475.37	23.32
	8-threads	0.42	5.83	53.71	59.81	302.84	132.66	34.75
	Optimistic	0.39	3.74	8.82	24.14	55.75	258.62	22.91
A-n64-k9	Sequential	0.48	10.72	245.34	666.07	1271.17	-	1126.11
	8-threads	0.49	7.82	78.30	204.69	280.62	786.56	696.48
	Optimistic	0.46	5.16	19.50	46.70	147.87	-	1107.69
A-n65-k9	Sequential	0.48	15.79	338.29	293.34	767.62	2255.19	99.70
	8-threads	0.48	6.19	78.91	121.31	239.45	588.43	123.50
	Optimistic	0.47	5.16	21.35	39.46	61.87	146.97	96.73
A-n69-k9	Sequential	0.57	17.00	330.79	1912.05	1419.08	2966.77	418.51
	8-threads	0.57	10.35	82.89	687.58	508.49	608.27	309.22
	Optimistic	0.58	6.11	16.83	76.13	125.22	164.91	408.41
A-n80-k10	Sequential	1.06	41.34	587.82	-	-	-	-
	8-threads	1.03	20.30	167.06	2202.50	1384.88	1679.24	-
	Optimistic	1.01	11.55	29.54	-	-	-	-

does so for some instances. We remark that we have observed more variation in computation times for the 8-thread implementation compared to the sequential implementation.

Table 2 serves to give an impression of the potential gains of parallelization. For a complete picture of the computation times for varying values of p, we provide the computation times of the LP bounds for all values of p of the E and A instances in Appendix I, corresponding to the implementation using 8 threads.

6.3 Computation time of branch-price-and-cut

We provide empirical evidence of the computational potential of using the p-step formulation. Recall that the used p-step formulation corresponds to SCF for $p = 1$ and to the set partitioning formulation for $p = n + 1$, with 2-cycle elimination. Therefore, the p-step formulation is in particular new for intermediate values of p, i.e., values of p for which the p-step formulation does not correspond to an arc-based and path-based formulation. If there are instances for which intermediate values of p correspond to the lowest computation time, we consider this empirical evidence of computational potential.

In Appendix J, we provide the computation times of our algorithm for the E and A instances with not more than 50 customers. We identified 4 instances out of 22 for which the fastest computation time is found for an intermediate value of p: instance E-n31-k7 and $p = 8$, instance A-n37-k6 and $p = 12$, instance A-n44-k6 and $p = 10$, and instance A-n46-k7 and $p = 11$. Moreover, we construct new instances that are specifically designed to favor intermediate values of p. For these instances, our algorithm consistently has the lowest computation time for intermediate values of p. We believe this is sufficient evidence to conclude that there is potential in developing new algorithms specifically designed for the p-step formulation, with the goal of decreasing computation time.

We create new instances of the CVRP as follows. We generate C clusters of Q customers each. We assume unit demand so the customers in a single cluster fit precisely in a vehicle of capacity Q. Within the clusters, the customers are positioned at equal intervals on a circle of diameter R. The centers of the cluster are positioned at equal intervals on a circle of diameter 100 centered at the origin. We place the depot far away at coordinate (10000, 10000). The travel costs are Euclidean and rounded to two decimal places. Precisely C vehicles must be used.

In Table 3, the computation times in seconds are provided for eight instances and for the values of p from 1 through $Q + 1$, since Q is the maximum number of customers that can be visited on a route. A dash indicates that our algorithm did not terminate within the time limit. Table 3 demonstrates that for these instances our algorithm is fastest for intermediate values of p.

7 Conclusion and future research

In this paper we have introduced the p-step formulation of the CVRP. It can be considered a generalization of the traditional arc-based and path-based formulations. We use this formulation to show that there does not exist a strongest compact formulation for the CVRP, if $P \neq NP$. We have also shown that computing the set partitioning bound of the CVRP is NP-hard. This justifies why the best performing algorithms

Table 3: Computation times in seconds of branch-price-and-cut.

R	C	Q	p : 1	2	3	4	5	6	7	8	9	10
0.0	4	8	13.71	42.46	216.56	355.58	160.13	0.14	1.99	0.34	3.67	
		9	29.52	107.26	367.02	1078.37	0.13	54.93	16.84	3.66	0.64	12.54
	5	8	33.28	658.82	-	234.40	0.21	0.16	0.12	0.92	11.74	
		9	396.98	-	-	-	-	-	0.18	0.17	3.89	65.85
0.1	4	8	0.24	0.51	1.92	1.06	0.16	0.18	0.08	0.51	1.47	
		9	0.12	0.25	0.89	1.44	0.34	0.26	0.18	0.11	1.36	865.55
	5	8	0.99	1.16	6.08	1.58	0.15	0.15	0.14	1.48	193.25	
		9	0.24	0.33	1.88	2.23	0.45	0.45	0.37	0.22	5.35	2165.89

in the literature do not use this bound, but typically resort to weaker bounds, e.g., obtained by route relaxations.

Although the search for the strongest compact formulation is over, we think it is relevant to develop stronger formulations with the number of variables and constraints limited by a polynomial of fixed degree. It is not clear, for instance, what the strongest formulation of the CVRP is using a quadratic number of variables and constraints. In this paper we have provided new strongest known formulations for the CVRP with the number of variables of fixed polynomial degree three and higher. Nonetheless, it remains an open question whether the set partitioning bound can be computed in pseudo-polynomial time, or also indeed what the strongest pseudo-compact formulation of the CVRP would be. In light of the p-step formulation, we conjecture that no strongest pseudo-compact formulation exists for the CVRP.

The experiments presented in this paper provide empirical evidence that the p-step formulation could potentially contribute to faster algorithms for the CVRP. To pursue the computational potential, the use of route relaxation, non-robust cuts, arc fixing and route enumeration can be further investigated. Moreover, we emphasize that in our current algorithm we decompose the pricing problem in a substantial number of familiar problems. Other approaches might be devised to deal with the pricing problem.

The p-step formulation can be extended to problems other than the CVRP. The simplest way to achieve this, is by using the arc-flow variables θ to model additional constraints. For example, time window constraints can now be modeled resulting in a formulation of the VRPTW. Moreover, we can also model capacity constraints differently, to avoid the use of the prior demand d_r. One could model capacity constraints through the arc-flow variables using for instance the Miller-Tucker-Zemlin constraints (Desrochers and Laporte 1991). The effect is a reduction in LP bounds, but also a potential reduction of the complexity of the pricing problem.

References

Augerat P (1995) Approche polyèdrale du problème de tournées de véhicules. *PhD thesis* Institut Nationale Polytechnique de Grenoble.

Baldacci R, Hadjiconstantinou E, Mingozzi A (2004) An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research* 52(5):723–738.

Baldacci R, Christofides N, Mingozzi A (2008) An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming* 115(2):351–385.

Baldacci R, Mingozzi A, Roberti R (2011) New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* 59(5):1269–1283.

Balinski M, Quandt E (1969) On an integer program for the delivery problem. *Operations Research* 12(2):300–304.

Christofides N, Eilon S (1969) An Algorithm for the Vehicle Dispatching Problem. *Journal of the Operational Research Society* 20(3):309–318.

Costa L, Contardo C, Desaulniers G (2019) Exact Branch-Price-and-Cut Algorithms for Vehicle Routing *Transportation Science* published online in Articles in Advance 28 June 2019.

Desrochers M, Laporte G (1991) Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters* 10(1):27–36.

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to some Vehicle Routing Problems. *Networks* 40(3):216–229.

Fukasawa R, Longo H, Lysgaard J, Poggi de Aragão M, Reis M, Uchoa E, Werneck R (2006) Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming* 106(3):491–511.

Garey M, Johnson D (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (WH Freeman & Co, New York).

Gavish B (1984) The delivery problem: new cutting plane procedures. *Presented at the TIMS XXVI conference, Copenhagen.*

Gouveia L (1995) A result on projection for the vehicle routing problem. *European Journal of Operational Research* 85(3):610–624.

Irnich S, Desaulniers G, Desrosiers J, Hadjar, A (2010) Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing* 22(2):297–313.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research* 56(2):391–406.

Jepsen M, Petersen B (2009) Partial Path Column Generation for the Vehicle Routing Problem. *Kgs. Lyngby: DTU Management No. 12.*

Karp R (1972) Reducibility among combinatorial problems. *Complexity of computer computations*, (Springer, Boston) 85–103.

Laporte G, Nobert Y (1987) Exact algorithms for the vehicle routing problem. *North-Holland Mathematics Studies* 132:147–184.

Leggieri V, Haouari M (2017) Lifted polynomial size formulations for the homogenous and heterogeneous vehicle routing problems. *European Journal of Operational Research* 263(3):755–767.

Letchford A, Salazar-González J (2015) Stronger multi-commodity flow formulations of the Capacitated Vehicle Routing Problem. *European Journal of Operational Research* 244(3):730–738.

Lysgaard J (2003) CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. *Technical report*, Department of Management Science and Logistics, Aarhus School of Business, Aarhus, Denmark.

Lysgaard J, Letchford A, Eglese R (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming* 100(2):423–445.

Martinelli R, Pecin D, Poggi M (2014) Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research* 239(1):102–111.

Pecin D, Pessoa A, Poggi M, Uchoa E. (2017) Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation* 9(1):61-101.

Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization* 3(3):255–273.

Righini G, Salani M (2008) New Dynamic Programming Algorithms for the Resource Constrained Elementary Shortest Path Problem. *Networks* 51(3):155–170.

Toth P, Vigo D (2014) Vehicle Routing, Problems, Methods, and Applications. *(Society for Industrial and Applied mathematics)*, 2nd.

A Proof of Proposition 5

Before presenting a proof of Proposition 5, we first provide another result which will be used in our proof. The following lemma is similar to what Gouveia (1995) showed for the single-commodity flow formulation of the CVRP. It states that the flow out of and into the depot is at least the total demand divided by the vehicle capacity.

Lemma 13. *Denote by R_0 and R_{n+1} the p-steps connected to the starting and ending depot respectively, and let x be a feasible solution of the LP relaxation of the p-step formulation. Then*

$$\sum_{r\in R_0} x_r = \sum_{r\in R_{n+1}} x_r \geq \frac{1}{Q}\sum_{i\in N'} q_i.$$

Proof. By (3), for any feasible solution, the amount of selected p-steps starting and ending at any location $i \in N'$ is equal. We can rewrite these equalities using the following notation. For every $i \in N$, let R_i^+ be the collection of p-steps that start at i, and let R_i^- be the collection of p-steps ending at i. Note that

$R_0^+ = R_0$ and $R_{n+1}^- = R_{n+1}$. Using this notation and (3), it follows that

$$\begin{aligned}\sum_{r\in R_i^+} x_r &= \sum_{r\in R_i^+} x_r - \sum_{r\in R^p} e_r^i x_r \\ &= \sum_{r\in R_i^+} \left(1-e_r^i\right) x_r + \sum_{r\in R_i^-} \left(-e_r^i\right) x_r \\ &= \sum_{r\in R_i^-} x_r.\end{aligned}$$

To show that the flow out of and into the depot is also the same, we first derive

$$\begin{aligned}\sum_{r\in R_0} x_r + \sum_{i\in N'} \sum_{i\in R_i^+} x_r &= \sum_{i\in N} \sum_{r\in R_i^+} x_r \\ &= \sum_{r\in R^p} x_r \\ &= \sum_{i\in N} \sum_{r\in R_i^-} x_r \\ &= \sum_{r\in R_{n+1}} x_r + \sum_{i\in N'} \sum_{i\in R_i^-} x_r.\end{aligned}$$

Combining the above shows that

$$\sum_{r\in R_0} x_r = \sum_{r\in R_{n+1}} x_r.$$

To derive a bound on the flow out of or into the depot, we proceed by relating the demand to the p-step variables. We use q_r^+ to denote the demand of the first location on p-step r. Note in particular that we define $q_r^+ = 0$ for all p-steps starting at the depot. Using (2), (3) and (4) or (6), it follows that

$$\begin{aligned}\sum_{i\in N'} q_i &\leq \sum_{i\in N'} \left(\frac{1}{2}\sum_{r\in R^p} a_r^i x_r - \frac{1}{2}\sum_{r\in R^p} e_r^i x_r\right) q_i + \sum_{i\in N'} \sum_{r\in R^p} q_r^i x_r \\ &= \sum_{r\in R^p} \sum_{i\in N'} q_i \frac{1}{2}\left(a_r^i - e_r^i\right) x_r + \sum_{r\in R^p} \sum_{i\in N'} q_r^i x_r \\ &= \sum_{r\in R^p} \left(q(r) - q_r^+ + \sum_{i\in N'} q_r^i\right) x_r.\end{aligned}$$

Note that the first inequality is valid both for (4) and for (6). In case (4) is considered, the inequality can be replaced by an equality. Observe that for every $r \in R^p \setminus (R_0 \cup R_{n+1})$, that is, for every p-step that does not start nor end at the depot, q_r^i is nonzero for exactly two customers $i \in N'$. In particular, it holds that

$$\begin{aligned}q(r) - q_r^+ + \sum_{i\in N'} q_r^i &= q(r) - q_r^+ + \left(d_r + q_r^+\right) - \left(d_r + q(r)\right) \\ &= 0\end{aligned}$$

We now obtain

$$\begin{aligned}\sum_{i\in N'} q_i &\leq \sum_{r\in R^p} \left(q(r) - q_r^+ + \sum_{i\in N'} q_r^i \right) x_r \\ &= \sum_{r\in R_0\setminus R_{n+1}} (-d_r - q_r^+) x_r + \sum_{r\in R_{n+1}\setminus R_0} (d_r + q(r)) x_r + \sum_{r\in R_0\cap R_{n+1}} (q(r) - q_r^+) x_r \\ &\leq \sum_{r\in R_{n+1}} Q x_r,\end{aligned}$$

where we use $0 \leq d_r + q_r^+$ and $q(r) - q_r^+ \leq d_r + q(r) \leq Q$ to derive the final inequality. This shows that $\sum_{r\in R_{n+1}} x_r \geq \frac{1}{Q} \sum_{i\in N'} q_i$. □

We now repeat Proposition 5 and provide a proof.

Proposition 5. *For any $p, q \in \mathbb{N}_{>0}$ such that $p < q$ and q is not an integer multiple of p, there exists an instance of the CVRP for which $z_p > z_q$.*

Proof. In this proof, we first present an instance of the CVRP. Then, we provide a feasible solution to the LP relaxation of the q-step formulation for this instance. Finally, we show that any feasible solution to the LP relaxation of the p-step formulation for this instance has a higher cost, showing $z_p > z_q$.

As q is not an integer multiple of p, there exist integers $m \geq 1$ and $k \in \{1, \ldots, p-1\}$ such that $q = pm + k$. Consider the following instance with $n = (m+1)\,p$ customers. In this instance, the travel costs are equal to the Euclidean distances between locations. The customers appear in $m+1$ clusters of p customers. The clusters are located on the vertices of a regular convex $(m+1)$-polygon with edges of length 1. Within a cluster the travel costs are 0 (or negligibly small), while traveling from a customer to a customer in a neighbouring cluster has cost 1 (or negligibly close to 1). Finally, the depot is located far away, and travel from any customer to the depot has a very large cost which is (negligibly close to) $C > 1$. All customers have unit demand and the vehicle capacity is $Q = p + \frac{pq}{k}$. By construction, $Q > p + q$ holds because $k < p$.

For ease of notation, let the clusters be numbered from 1 to $m+1$ as they appear along the boundary of the polygon. As a result, the cost of traveling between a customer in cluster c and $c+1$ is 1, for $1 \leq c \leq m$, as is the cost of travel between clusters $m+1$ and 1. Similarly let the customers be numbered such that customers $(c-1)p + 1$ to cp are in cluster c.

Next, we construct a feasible solution for the LP relaxation of the q-step formulation of this instance. To do so, first consider the following path P_c^t visiting all customers in cluster c in order of numbering, but cyclically permuted t times, for $0 \leq t < p$.

$$P_c^t = (c-1)p + 1 + (t \bmod p) \rightarrow \ldots \rightarrow (c-1)p + 1 + ((t + p - 1) \bmod p)$$

For example, if we consider cluster $c = 2$ in case each cluster contains $p = 4$ customers, and the customer are cyclically permuted $t = 2$ times, then $P_c^t = 7 \rightarrow 8 \rightarrow 5 \rightarrow 6$. Next, we concatenate these paths for all clusters, starting with cluster c and obtain the path $P(c, t)$, for $1 \leq c \leq m+1$ and $0 \leq t < p$.

$$P(c,t) = P^t_{1+(c-1) \bmod (m+1)} \to P^t_{1+(c) \bmod (m+1)} \to \dots \to P^t_{1+(c+m-1) \bmod (m+1)}$$

For example, if $m+1=3$ and $p=3$, $P(2,1) = 5 \to 6 \to 4 \to 8 \to 9 \to 7 \to 2 \to 3 \to 1$. Note that the cost of traversing all arcs in any path $P(c,t)$ is m.

Finally, we truncate path $P(c,t)$ by removing the last arcs, to construct a path $P(c,t,q)$ traversing exactly q arcs. Note that also the cost of traversing all arcs on any path $P(c,t,q)$ is m. Moreover, observe that for a given position between 1 and $q+1$, every customer $i \in V'$ appears at that position on exactly one path $P(c,t,q)$, among all $1 \le c \le m+1$ and $0 \le t < p$. Similarly, we introduce truncated paths $P(c,t,q-1)$ traversing the first $q-1$ arcs of $P(c,t)$. Note that the total arc costs of these paths are also m.

Using the above paths, we now present a feasible solution to the LP relaxation of the q-step formulation. It selects all of the following q-steps with a positive value, for all $1 \le c \le m+1$ and $0 \le t < p$:

Type 1 $(0 \to P^t_c, 0)$ with total demand p,

Type 2 $(P(c,t,q), 0)$ with total demand $q+1$,

Type 3 $(P(c,t,q-1) \to n+1, Q-q)$ with total demand q.

Observe that given the value of Q, each selected q-step r is feasible since $0 \le d_r + q(r) \le Q$. The q-steps of type 1 and 3 are selected with value $\frac{1}{Q}$ each, and the q-steps of type 2 are selected with value $\frac{Q-p-q+1}{Qq}$. We verify that this is indeed a feasible solution.

First observe that the selection of p-steps is non-negative. Next, consider (2). Observe that every customer $i \in N'$ appears once as the last node of exactly one q-step of type 1, once as the first and once as the last node of a q-step of type 2, and once as the first node of a q-step of type 3. In these cases, $a^i_r = 1$. Moreover, every customer $i \in N'$ appears as not the first or last node for $p-1$ times on a q-step of type 1, for $q-1$ times on a q-step of type 2 and $q-1$ times on a q-step of type 3. Here, $a^i_r = 2$. Therefore, it follows for all $i \in N'$ that

$$\begin{aligned}\sum_{r \in R^p} a^i_r x_r &= \frac{1}{Q} + \frac{Q-p-q+1}{Qq} + \frac{Q-p-q+1}{Qq} + \frac{1}{Q} \\ &\quad + 2\left((p-1)\frac{1}{Q} + (q-1)\frac{Q-p-q+1}{Qq} + (q-1)\frac{1}{Q}\right) \\ &= 2.\end{aligned}$$

Hence, the suggested solution satisfies (2). Similarly, we verify that also (3) are satisfied. Since every customer $i \in N'$ appears as the last node of one q-step of type 1 and one of type 2, and as the first node of one q-step of type 2 and one of type 3, it follows that

$$\begin{aligned}\sum_{r\in R^p} e_r^i x_r &= -\frac{1}{Q} - \frac{Q-p-q+1}{Qq} + \frac{Q-p-q+1}{Qq} + \frac{1}{Q}\\ &= 0.\end{aligned}$$

Finally, we verify that (4), and also (6), are satisfied. Note that for a q-step of type 1 with customer $i \in N'$ as last location it holds that $q_r^i = -p$. For a q-step of type 2 with i as first location $q_r^i = 1$, while if i is the last location $q_r^i = -(q+1)$. Moreover, for a q-step of type 3 with i as first location $q_r^i = (Q-q+1)$. It follows that

$$\begin{aligned}\sum_{r\in R^p} q_r^i x_r &= \frac{-p}{Q} + \frac{Q-p-q+1}{Qq} - (q+1)\frac{Q-p-q+1}{Qq} + \frac{Q-q+1}{Q}\\ &= 0.\end{aligned}$$

We conclude that the suggested solution is feasible for the LP relaxation of the q-step formulation. Moreover, observe that the cost of a q-step of type 1, 2 and 3 is C, m and $C+m$ respectively, and n of each type are selected in the solution. As a result, the cost of the provided solution is $\frac{2n}{Q}C + \frac{n(Q-p+1)}{Qq}m$. Hence, $z_q \leq \frac{2n}{Q}C + \frac{n(Q-p+1)}{Qq}m$.

We complete this proof by showing that any feasible solution to the LP relaxation of the p-step formulation for this instance has a higher cost than the presented solution to the q-step formulation. Let us first separately consider the p-steps connected to a depot. Denote by R_0 the p-steps connected to the starting depot 0, and denote by R_{n+1} the p-steps connected to the ending depot $n+1$. Observe that $R_0 \cap R_{n+1}$ contains all p-steps that start at 0 and end at $n+1$. For such p-steps $r \in R_0 \cap R_{n+1}$ it holds that $c_r \geq 2C$, and for the p-steps $r \in R_0 \cup R_{n+1} \setminus R_0 \cap R_{n+1}$ it holds that $c_r \geq C$. This allows us to derive the following bound on the costs of the selected p-steps in any feasible solution to the LP relaxation of the p-step formulation.

$$\sum_{r\in R_0\cup R_{n+1}} c_r x_r \geq \sum_{r\in R_0} C x_r + \sum_{r\in R_{n+1}} C x_r$$

Next, we derive a bound on $\sum_{r\in R^p} x_r$, which could be interpreted as the amount of selected p-steps, by relating it to the degree of each node in a solution. For every p-step $r \in R^p \setminus (R_0 \cup R_{n+1})$, which are the p-steps not connected to a depot, it holds that exactly p arcs are traversed. Observe that for such a p-steps r it holds that $\sum_{i\in N'} a_r^i = 2p$, and equivalently $\frac{1}{2p}\sum_{i\in N'} a_r^i = 1$. Similarly, for all $r \in R_{n+1} \setminus R_0$ it holds that $1 + \sum_{i\in N'} a_r^i = 2p$. For all $r \in R_0$, we know that at most p arcs are traversed. Hence, for all $r \in R_0 \setminus R_{n+1}$ it holds that $1 + \sum_{i\in N'} a_r^i \leq 2p$, and for $r \in R_0 \cap R_{n+1}$ it holds that $2 + \sum_{i\in N'} a_r^i \leq 2p$. Combining these observations, and using (2) yields

$$\begin{aligned}\sum_{r\in R^p} x_r &\geq \frac{1}{2p}\left(\sum_{r\in R^p}\sum_{i\in N'} a_r^i x_r + \sum_{r\in R_0} x_r + \sum_{r\in R_{n+1}} x_r\right)\\ &= \frac{n}{p} + \frac{1}{2p}\left(\sum_{r\in R_0} x_r + \sum_{r\in R_{n+1}} x_r\right).\end{aligned}$$

Finally, we derive a bound on the solution value of any feasible solution to the p-step formulation. Using that for any p-step $r \in R^p \setminus R_0 \cup R_{n+1}$ it holds that $c_r \geq 1$, it follows that

$$\begin{aligned}\sum_{r\in R^p} c_r x_r &= \sum_{r\in R^p\setminus(R_0\cup R_{n+1})} c_r x_r + \sum_{r\in R_0\cup R_{n+1}} c_r x_r\\ &\geq \sum_{r\in R^p\setminus(R_0\cup R_{n+1})} x_r + \sum_{r\in R_0} C x_r + \sum_{r\in R_{n+1}} C x_r\\ &\geq \sum_{r\in R^p} x_r + \sum_{r\in R_0} (C-1)x_r + \sum_{r\in R_{n+1}} (C-1)x_r\\ &\geq \frac{n}{p} + \left(C - 1 + \frac{1}{2p}\right)\left(\sum_{r\in R_0} x_r + \sum_{r\in R_{n+1}} x_r\right)\\ &\geq \frac{n}{p} + \left(C - 1 + \frac{1}{2p}\right)\frac{2n}{Q}\\ &= \frac{2n}{Q}C + \frac{n(Q-2p+1)}{Qp},\end{aligned}$$

where we used Lemma 13 in the last inequality. As this holds for all feasible solutions, the last expression is a lower bound on z_p.

Comparing the LP bounds of the q-step formulation and the p-step formulation, we observe

$$\begin{aligned}z_p - z_q &\geq \left(\frac{2n}{Q}C + \frac{n(Q-2p+1)}{Qp}\right) - \left(\frac{2n}{Q}C + \frac{n(Q-p+1)}{Qq}m\right)\\ &= \frac{n\left(Q(q-mp) + (p-1)(mp-q) - pq\right)}{Qpq}\\ &= \frac{n\left(Qk - pk - pq + k\right)}{Qpq}.\end{aligned}$$

Because $Q = p + \frac{pq}{k}$, we find for this instance that

$$\begin{aligned}z_p - z_q &\geq \frac{nk}{Qpq}\\ &> 0.\end{aligned}$$

This proves the claim. $\square$

B Proof of Proposition 7

We repeat Proposition 7 and provide a proof.

Proposition 7. *For any $p, q \in \mathbb{N}_{>0}$ such that $p < q$, there exists an instance of the CVRP for which $z_p < z_q$.*

Proof. Consider an instance with $n = p + 1$ customers with unit demand. In this instance, the travel costs are equal to the Euclidean distances between locations. The customers are located at the vertices of a negligibly small regular convex $(p + 1)$-polygon. The depot is located far away, such that the distance between the depot and each of the customers is (negligibly close to) C. Finally, the vehicle has capacity $2(p + 1)$.

Next, we construct a feasible solution to the LP relaxation of the p-step formulation. Let P^t be the path starting with an arbitrary but fixed customer, visiting all other customers in order of appearance along the edges of the polygon, cyclically permuted t times, for $0 \leq t \leq p$. Furthermore, let $P(t, p-1)$ be path P^t truncated after traversing the first $p - 1$ arcs. Using these paths, we now present a solution to the LP relaxation of the p-step formulation. It selects all of the following p-steps with a positive value, for all $0 \leq t \leq p$:

Type 1 $(0 \to P(t, p-1), 0)$ with total demand p,

Type 2 $(P^t, 0)$ with total demand $p + 1$,

Type 3 $(P(t, p-1) \to n + 1, p + 2)$ with total demand p.

The p-steps of type 1 and 3 are selected with value $\frac{1}{2(p+1)}$ each, and the p-steps of type 2 are selected with value $\frac{3}{2p(p+1)}$. Similar to the proof of Proposition 5 found in Appendix A, one can verify that this is a feasible solution. Moreover, observe that the cost of a p-step of type 1, 2 and 3 is C, 0 and C respectively. As a result, the cost of the provided solution is $\frac{2(p+1)C}{2(p+1)} = C$.

Given that $q \geq p + 1 = n$, all q-steps start or end at the depot and visit at most n customers. It follows that $c_r \geq C$ and that

$$\sum_{i \in N'} a_i^r < 2n$$

for all q-steps $r \in R^q$. For any solution x_r to the LP-relaxation of the q-step formulation, it follows from (2) that

$$\begin{aligned} 2n &= \sum_{i \in N'} \sum_{r \in R^q} a_i^r x_r \\ &= \sum_{r \in R^q} \sum_{i \in N'} a_i^r x_r \\ &< \frac{2n}{C} \sum_{r \in R^q} C x_r \\ &\leq \frac{2n}{C} \sum_{r \in R^q} c_r x_r. \end{aligned}$$

This shows that $z_q > C \geq z_p$. The claim follows.

□

C Proof of Proposition 8

We repeat Proposition 8 and provide a proof.

Proposition 8. *Computing the set partitioning bound of the CVRP is NP-hard.*

Proof. We provide a polynomial time reduction of the partition problem, which is known to be NP-hard (Karp 1972). In the partition problem, a set of integer numbers $D = \{d_1, \dots, d_k\}$ is provided. The problem is to determine whether a subset S of these numbers exist such that $\sum_{i \in S} d_i = \frac{1}{2} \sum_{i \in D} d_i$, yes or no.

We create an instance of the LP relaxation of the set partitioning formulation of the CVRP as follows. Introduce a customer for each number, i.e., $N' = \{1, \dots, k\}$, with demand $q_i = d_i$ for all $i \in N'$. Let the vehicle capacity be $Q = \frac{1}{2} \sum_{i=1}^{k} d_i$. Define the travel cost between the depot and any customer as 1, i.e., $c_{0i} = c_{i(n+1)} = 1$ for $i \in N'$. All other travel costs are 0.

For an instance of which the answer to the partition problem is yes, a set S exists such that $\sum_{i \in S} q_i = Q$, and also for $\bar{S} = D \backslash S$ it holds that $\sum_{i \in \bar{S}} q_i = Q$. Hence a feasible solution to the LP relaxation of the set partitioning formulation of the CVRP can be found, by selecting one route visiting all customers in S and another to visit the customers in $\bar{S}$. The total cost of this solution is 4, hence the set partitioning bound of the CVRP is in this case 4 or less. Noting that the solution we have constructed is a feasible solution to the CVRP, we conclude that actually any lower bound on the optimal solution value is at most 4 for this instance.

For an instance of which the answer to the partition problem is no, there does not exist a set S such that $\sum_{i \in S} q_i = Q$. Hence, for any feasible route visiting a subset of customers S, it holds that $\sum_{i \in S} q_i \leq Q - 1$ since all demands are integer. Therefore, the solution to this instance is the same as that for a related instance, which we construct by changing the vehicle capacity to $Q' = Q - 1$. By Lemma 13 found in Appendix A we know that the flow out of and into the depot is at least $\frac{1}{Q'} \sum_{i \in N'} q_i > 2$. Hence, the total cost of any feasible solution is strictly larger than 4.

This shows that the partition problem can be solved by computing the set partitioning bound of the CVRP. Indeed if the set partitioning bound is strictly larger than 4, the answer is no, and otherwise the answer is yes. We conclude that computing the set partitioning bound of the CVRP is NP-hard. □

D Proof of Proposition 11

Before proving Proposition 11, we first provide SCF. This mixed integer programming formulation makes use of the continuous variables f_{ij} which can be interpreted as the total load carried by the vehicle traversing arc $(i, j) \in A$. The formulation is the following.

$$(SCF) \quad \min \sum_{(i,j)\in A} c_{ij}\theta_{ij} \tag{12}$$

$$\sum_{j\in N} \theta_{ji} = 1 \qquad \forall i \in N' \tag{13}$$

$$\sum_{j\in N} \theta_{ij} = 1 \qquad \forall i \in N' \tag{14}$$

$$\sum_{j\in N} f_{ji} - \sum_{j\in N} f_{ij} = q_i \qquad \forall i \in N' \tag{15}$$

$$q_j\theta_{ij} \leq f_{ij} \qquad \forall (i,j) \in A \tag{16}$$

$$f_{ij} \leq (Q - q_i)\,\theta_{ij} \qquad \forall (i,j) \in A \tag{17}$$

$$\theta_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{18}$$

The objective (12) is to minimize the total costs. Constraints (13) and (14) are the indegree and outdegree constraints. Constraints (15) ensure that after visiting a customer the vehicle load is reduced by the demand. Constraints (16) specify that when visiting a customer, a vehicle should carry at least the demand. Furthermore, constraints (17) ensure that the vehicle load does not exceed the capacity. Finally, (18) specify the domains of the arc-flow variables.

Before presenting a proof of Proposition 11, stating that the 1-step formulation has the same LP bound as SCF, we first show a stronger result. We show that actually for any choice of p, the p-step formulation is at least as strong as SCF.

Lemma 14. *For any $p \in \mathbb{N}_{>0}$, the p-step formulation is at least as strong as SCF.*

Proof. Consider any feasible solution (x, θ) to the LP relaxation of the p-step formulation (1)-(5). We show that there exists a corresponding solution to the LP relaxation of SCF with the same objective value, proving the p-step formulation is at least as strong as SCF.

Define $q(r,i)$ as the sum of the demands of the customers that are visited on p-step $r \in R^p$ up to and including location $i \in N$, if i is visited by r, and set

$$f_{ij} = \sum_{r\in R^p} b^r_{ij}(Q - d_r - q(r,i))x_r.$$

Next, we show that (θ, f) is a feasible solution of the LP relaxation of SCF with the same objective value as (x, θ).

First consider the objective value. By definition, the objective value of (θ, f) is given by

$$\begin{aligned}\sum_{(i,j)\in A} c_{ij}\theta_{ij} &= \sum_{(i,j)\in A} c_{ij} \sum_{r\in R^p} b^r_{ij} x_r \\ &= \sum_{r\in R^p} \left(\sum_{(i,j)\in A} c_{ij} b^r_{ij} \right) x_r \\ &= \sum_{r\in R^p} c_r x_r.\end{aligned}$$

Next, we show that the solution satisfies all constraints of the LP relaxation of SCF. Like in the proof of Lemma 13, let R_i^+ be the collection of p-steps that start at i, and let R_i^- be the collection of p-steps that end at i. Furthermore, let $R_i^=$ be the collection of p-steps that include i but do not start nor end at i. Note that in the proof of Lemma 13, we demonstrate that $\sum_{r\in R_i^+} x_r = \sum_{r\in R_i^-} x_r$ for any $i \in N'$. Using this and (2), we derive for all $i \in N'$ that

$$\begin{aligned}1 &= \frac{1}{2} \sum_{r\in R^p} a^i_r x_r \\ &= \frac{1}{2} \sum_{r\in R_i^+} x_r + \sum_{r\in R_i^=} x_r + \frac{1}{2} \sum_{r\in R_i^-} x_r \\ &= \sum_{r\in R_i^+} x_r + \sum_{r\in R_i^=} x_r \\ &= \sum_{r\in R_i^=} x_r + \sum_{r\in R_i^-} x_r.\end{aligned}$$

We now obtain for all $i \in N'$,

$$\begin{aligned}\sum_{j\in N} \theta_{ji} &= \sum_{j\in N} \sum_{r\in R^p} b^r_{ji} x_r \\ &= \sum_{r\in R^p} \left(\sum_{j\in N} b^r_{ji} \right) x_r \\ &= \sum_{r\in R_i^= \cup R_i^-} x_r \\ &= 1,\end{aligned}$$

showing that (13) are satisfied. Similarly,

$$\begin{aligned}\sum_{j\in N} \theta_{ij} &= \sum_{j\in N} \sum_{r\in R^p} b^r_{ij} x_r \\ &= \sum_{r\in R^p} \left(\sum_{j\in N} b^r_{ij} \right) x_r \\ &= \sum_{r\in R_i^+ \cup R_i^=} x_r \\ &= 1,\end{aligned}$$

showing that (14) are satisfied. To verify that (15) are satisfied, we derive for $i \in N'$ that

$$\begin{aligned}
\sum_{j\in N} f_{ji} - \sum_{j\in N} f_{ij} =& \sum_{j\in N}\sum_{r\in R^p} \left(b^r_{ji}(Q-d_r-q(r,j)) - b^r_{ij}(Q-d_r-q(r,i))\right) x_r \\
=& \sum_{r\in R^+_i} x_r \sum_{j\in N}\left(-b^r_{ij}(Q-d_r-q_i)\right) \\
&+ \sum_{r\in R^=_i} q_i x_r + \sum_{r\in R^-_i} x_r \sum_{j\in N}\left(b^r_{ji}(Q-d_r-q(r,j))\right) \\
=& Q\left(-\sum_{r\in R^+_i} x_r + \sum_{r\in R^-_i} x_r\right) \\
&+ \sum_{r\in R^+_i} q^i_r x_r + \sum_{r\in R^=_i} q_i x_r + \sum_{r\in R^-_i} x_r\left(q^i_r + q_i\right) \\
=& q_i\left(\sum_{r\in R^=_i\cup R^-_i} x_r\right) + \sum_{r\in R^+_i\cup R^-_i} q^i_r x_r \\
=& q_i,
\end{aligned}$$

where we use (4) to derive the final equality by noting that $q^r_i = 0$ for all $r \in R^=_i$. This shows that (15) are satisfied.

Finally, observe that for any p-step $r \in R^p$, it holds that $d_r + q(r) \le Q$. Furthermore, if p-step r includes arc $(i,j) \in A$, then $d_r + q(r,i) + q_j \le d_r + q(r) \le Q$. Hence,

$$\begin{aligned}
q_j\theta_{ij} &= \sum_{r\in R^p} b^r_{ij} x_r q_j \\
&\le \sum_{r\in R^p} b^r_{ij} x_r (Q-d_r-q(r,i)) \\
&= f_{ij},
\end{aligned}$$

showing that (16) are satisfied. Similarly,

$$\begin{aligned}
(Q-q_i)\theta_{ij} &= \sum_{r\in R^p} b^r_{ij} x_r (Q-q_i) \\
&\ge \sum_{r\in R^p} b^r_{ij} x_r (Q-d_r-q(r,i)) \\
&= f_{ij},
\end{aligned}$$

showing that (17) are satisfied. We conclude that the solution (θ, f) is a feasible solution for the LP relaxation of SCF. From this, it follows that the p-step formulation is at least as strong as SCF. □

We repeat Proposition 11 and provide a proof.

Proposition 11. *The* 1*-step formulation is equally strong as SCF.*

Proof. By Lemma 14 it holds that the 1-step formulation is at least as strong as SCF. What remains to be shown is that SCF is at least as strong as the 1-step formulation, by demonstrating that any feasible solution to the LP relaxation of SCF corresponds to a feasible solution to the LP relaxation of the 1-step formulation, with the same objective value. Consider a solution to the LP relaxation of SCF. For all $(i,j) \in A$ such that $\theta_{ij} > 0$, define the 1-step

$$r(i,j) = \left((i,j), Q - \frac{f_{ij}}{\theta_{ij}} - q_i\right)$$

and let $x_{r(i,j)} = \theta_{ij}$. From (16), it follows that

$$\begin{aligned} d_r &= Q - \frac{f_{ij}}{\theta_{ij}} - q_i \\ &\le Q - \frac{q_j \theta_{ij}}{\theta_{ij}} - q_i \\ &= Q - q_i - q_j. \end{aligned}$$

From (17), it follows that

$$\begin{aligned} d_r &= Q - \frac{f_{ij}}{\theta_{ij}} - q_i \\ &\ge Q - \frac{(Q - q_i)\theta_{ij}}{\theta_{ij}} - q_i \\ &= 0. \end{aligned}$$

This shows that the 1-step $((i,j), d_r)$ is feasible.

It is a standard result that (2) and (3) coincide with (13) and (14). It remains to be shown that the solution satisfies (4). In order to do so, note that $q_r^i \neq 0$ only if i is visited by p-step r, that is, if $P_r = (i,j)$ or if $P_r = (j,i)$ for some $j \in N$. Also note that

$$q^i_{r(i,j)} = Q - \frac{f_{ij}}{\theta_{ij}}$$

and

$$\begin{aligned} q^i_{r(j,i)} &= -\left(Q - \frac{f_{ji}}{\theta_{ji}} - q_j\right) - q_i - q_j \\ &= -\left(Q - \frac{f_{ji}}{\theta_{ji}} + q_i\right). \end{aligned}$$

This allows us to derive

$$\begin{aligned} \sum_{r \in R^p} q^i_r x_r &= \sum_{j \in N} q^i_{r(i,j)} x_{r(i,j)} + \sum_{j \in N} q^i_{r(j,i)} x_{r(j,i)} \\ &= \sum_{j \in N} \left(Q - \frac{f_{ij}}{\theta_{ij}}\right)\theta_{ij} - \sum_{j \in N}\left(Q - \frac{f_{ji}}{\theta_{ji}} + q_i\right)\theta_{ji} \\ &= Q\sum_{j \in N}\theta_{ij} - Q\sum_{j \in N}\theta_{ji} - q_i \sum_{j \in N}\theta_{ji} + \sum_{j \in N} f_{ji} - \sum_{j \in N} f_{ij} \\ &= 0, \end{aligned}$$

where the last equation follows from (13)-(15). Thus, the solution satisfies (4). Since the corresponding solution is feasible and clearly has the same solution value as the original, SCF is at least as strong as the 1-step formulation.

We conclude that the 1-step formulation and SCF are equally strong. $\square$

E Exact labeling algorithm

Next, we describe a labeling algorithm used to solve the (s, f) pricing problem for a given choice of the prior demand. A label L corresponds to a partial path P_L such that $N(L)$ is the terminal node, $A(L)$ is the number of arcs on P_L, $q(L)$ is the cumulative demand of all nodes on P_L and the cumulative reduced cost of the traversed arcs is $c(L)$. Moreover, similar to Feillet et al. (2004) we use a vector $U(L) \in \mathbb{B}^{|N|}$, such that $U_i(L)$ indicates whether node $i \in N$ is unreachable. That is, if $U_i(L) = 1$, P_L cannot be extended to node i because i is either already visited or a visit would result in a violation of the capacity constraint because $q(L) + q_i > Q$.

We distinguish between forward labels, corresponding to partial paths from s to $N(L)$, and backward labels, corresponding to partial paths from $N(L)$ to f. We initialize the labeling algorithm with a forward label L corresponding to a partial path starting at s, such that $N(L) = s$, $A(L) = 0$, $q(L) = q_s$, $U_s(L) = 1$ and $U_i(L) = 0$ for all $i \in N\backslash\{s\}$, and the cumulative reduced cost is initialized as the reduced costs constant, i.e., $c(L) = C_{sf}$. Similarly we initialize a backward label corresponding to a partial path ending at f, although we set the reduced cost of this path to 0. Note that including the reduced cost constant in the forward label is an arbitrary choice.

Next, iteratively a label L, which has not yet been extended, is selected and extended. A partial path P_L corresponding to a forward label L from s to $N(L)$ is extended along an arc $(N(L), j) \in A$ to create a new label L', corresponding to a new partial path $P_{L'}$ from s through $N(L)$ to j. To describe the extension procedure, we define the function $F_i(L)$ for $i \in N$, that takes value 1 if $q(L) + q_i > Q$. The following extension functions are used to extend partial path P_L, corresponding to forward label L, along $(N(L), j) \in A$.

$$c(L') = c(L) + c'_{N(L)j} \tag{19}$$

$$q(L') = q(L) + q_j \tag{20}$$

$$N(L') = j \tag{21}$$

$$A(L') = A(L) + 1 \tag{22}$$

$$U_i(L') = \begin{cases} 1 & \text{if } i = j \\ \max\{U_i(L), F_i(L')\} & \text{otherwise} \end{cases} \qquad \forall i \in N \tag{23}$$

Analogously, a backward label L is extended in the reverse direction of an arc $(j, N(L)) \in A$.

The bi-directional labeling algorithm extends a forward label L for which $A(L) < p - \left\lfloor \frac{p-1}{2} \right\rfloor - 1$ along all arcs $(N(L), j) \in A$ such that $U_j(L) = 0$ and $j \neq f$, $j \neq n+1$. An exception to this is that a forward label L is extended to f, even if $f = n+1$, in case the source s is the depot, because in this case a path may consist of less than p arcs. A backward label L for which $A(L) < \lfloor \frac{p-1}{2} \rfloor$ is extended in the reverse direction of each arc $(j, N(L)) \in A$ such that $U_j(L) = 0$ and $j \neq s$, $j \neq 0$. Observe that the limits we impose on $A(L)$ for considering the extension of L, ensures that the longest partial paths corresponding to forward and backward labels differ in length by at most one arc. This ensures that the amount of generated forward and backward labels are roughly the same.

When no labels are left for extension, forward and backward labels are merged to construct full paths. By merging a forward label L for which $A(L) = p - \lfloor \frac{p-1}{2} \rfloor - 1$ and a backward label L' for which $A(L') = \lfloor \frac{p-1}{2} \rfloor$, a feasible path from s to f is created consisting of precisely p arcs, when $(N(L), N(L')) \in A$, $q(L) + q(L') \leq Q_{sf}$ and L and L' have not visited any common nodes. In our implementation, for the purpose of assessing whether L and L' have visits in common we also store for every label L a vector $V(L)$, for which $V_i(L)$ indicates whether $i \in N$ is visited or not. Note that $U(L)$ cannot be used for this purpose, but a backtracking procedure can alternatively be used if one does not wish to store $V(L)$ for each label. Furthermore, in case the start node s is the depot, the path need not have exactly p arcs. In this case, all forward labels are considered for merging, so also each forward label L for which $A(L) < p - \lfloor \frac{p-1}{2} \rfloor - 1$. Note that we need not consider more backward labels.

To avoid enumerating all labels, a dominance procedure is used to eliminate labels. A label L dominates label L' if every feasible extension of L' is also feasible for L, while the extension of L results in lower costs. In our labeling algorithm we apply the following sufficient conditions for dominance of L over L'. If $N(L) = N(L')$, $q(L) \leq q(L')$, $U(L) \leq U(L')$, $c(L) \leq c(L')$ and $A(L) = A(L')$ we eliminate label L'. We particularly want to emphasize the condition $A(L) = A(L')$, which is needed due to the requirement on the number of arcs on a path. After the construction of a new label L by extension, we perform a dominance check. In principle, this dominance check is performed of L over any other label, and of any other label over L. However, we can limit the amount of dominance checks as explained next.

We store the forward and backward labels in a separate dynamic programming matrix with n rows and $p+1$ columns. Row i column j of the forward matrix corresponds to a list of all labels with terminal node i and using $j-1$ arcs, and the backward matrix has the same interpretation for backward labels. This allows us to implicitly keep track of $N(L)$ and $A(L)$ which now no longer have to be stored separately per label. Also, $N(L)$ and $A(L)$ no longer have to be checked explicitly during the dominance check, since we only need to compare labels from the same entry in the matrix. Furthermore, we keep the lists of labels sorted in order of reduced costs, which helps us limit the number of dominance checks we need to perform if a new label is created. It also helps us limit the amount of merges to attempt, since we are only interested in a merge if the resulting partial path has negative reduced costs.

We apply decremental state space relaxation which is summarized as follows. Initially, we relax the pricing problem by allowing any cyclic path. If the optimal path does not contain a cycle, it is also optimal for the original pricing problem. Otherwise we prevent the generation of the same cycle and resolve. We prevent the generation of the given cycle as follows. For each node $i \in N$ we generate a neighborhood

NG_i which is initially empty. When a cycle C is observed in an optimal path, we identify the first location j on the path which is in the cycle and add j to the neighborhoods NG_i for all $i \in C \backslash \{j\}$. Furthermore, we modify the extension procedure by replacing (23) by

$$U_i(L') = \begin{cases} 1 & \text{if } i = j \\ \max\{U_i(L), F_i(L')\} & \text{if } i \in NG_j, \\ 0 & \text{otherwise.} \end{cases}$$

This prevents the generation of the same cycle in the next iteration of the decremental state space procedure. This particular way of cycle elimination results in the generation of ng-paths, as introduced by Baldacci et al. (2011), where we have neighborhoods of increasing size in each iteration of the decremental state space algorithm.

Finally, we apply completion bounds to eliminate even more labels. For every entry of the dynamic programming matrix, we find a lower bound on the additional reduced cost required for completing the path. When creating a label such that the corresponding reduced cost plus the lower bound is non-negative, the label cannot result in a path with a negative reduced cost, and we eliminate it. We obtain lower bounds after each iteration of the decremental state space relaxation, by completing the extension of backward labels to s and forward labels to f. The lowest reduced cost among the backward labels in the dynamic programming matrix row i column j, serves as a completion bound for a forward label in the dynamic programming matrix row i column $p + 2 - j$, and vice versa.

F Branch-price-and-cut details

We initialize our branch-price-and-cut algorithm with the preprocessing procedure described in Section 4.3 and by generating a limited set of p-steps. We generate these p-steps in two ways. First, we generate a feasible solution for the CVRP instance using a greedy insertion procedure, which we decompose into p-steps. Secondly, for each (s, f) pricing problem, we generate p-steps using a greedy insertion procedure as well. Specifically, in accordance with Proposition 1, if $s = 0$ and $f = n+1$ we create p-steps of length 2 through p with prior demand 0. If $s = 0$ and $f \neq n + 1$, we create p-steps of length 1 through p with prior demand 0. If $s \neq 0$ and $f = n + 1$, we create a p-step of length p with prior demand $Q - q(r)$. Finally, if $s \neq 0$ and $f \neq n + 1$, we create a p-step of length p with prior demand 0 and one with prior demand $Q - q(r)$. Observe that this way, for $p = 1$ all relevant p-steps are included, and no new p-steps need to be generated anymore.

We use a column generation algorithm to compute the LP bound at each node in the branching tree. We solve every (s, f) pricing problem that has not been eliminated in preprocessing. For each (s, f) pricing, we first apply a local search heuristic, in which the best swap is performed at each iteration. When $s = 0$, the path length may vary and we additionally consider insert and remove operations. The local search heuristic is initialized by constructing a feasible p-step by a greedy insertion procedure. We apply the local search heuristic an additional amount of times, once for every p-step selected in the current solution. The heuristic terminates if a negative reduced cost column is found, in which case we continue with the column generation algorithm, or no improvement is found. In the latter case, we continue by applying a labeling heuristic. The labeling heuristic is similar to the exact labeling algorithm, but for every customer $i \in N'$ only the X_{Arcs} arcs $(i, j) \in A$ with the lowest cost c_{ij} are considered. Furthermore, in the labeling heuristic, unreachability is not considered when performing a dominance check.

If the labeling heuristic also fails to identify negative reduced cost columns, we first attempt to separate valid inequalities to strengthen the LP bound before using the exact labeling algorithm. In the root node of the branching tree, we separate the rounded capacity, framed capacity, homogeneous multistar, strengthened comb and 2 edge hypotour inequalities using the package of heuristic separation algorithms of Lysgaard (2003), see also Lysgaard et al. (2004). Similar to Lysgaard et al. (2004), we separate valid inequalities as follows. We first attempt to separate rounded capacity inequalities, if it fails we attempt to separate framed capacity inequalities, if that fails we attempt to separate, in revolving order per iteration, the homogeneous multistar, strengthened comb and 2 edge hypotour inequalities. In all other nodes of the branching tree, we only separate rounded capacity inequalities. When no violated valid inequalities are identified, we use the exact labeling algorithm, otherwise we move to the next iteration of the column generation algorithm.

Both when using the heuristic and exact labeling algorithm, only the first X_{ColAdd} found columns with negative reduced costs are added to the RMP. Note that the state space used for decremental state space relaxation is reset as empty only at each iteration of the column generation algorithm, so not after the heuristic labeling algorithm has failed. Furthermore, we apply column management as follows. At every iteration in which the solution value has strictly decreased, we remove columns in decreasing order of reduced costs, until all columns with positive reduced costs are removed or only X_{ColPool} columns are left. When generating columns, first the pool of removed columns is checked for negative reduced cost columns. If they are found they are added to the RMP, and otherwise we proceed with the pricing heuristics.

If a fractional solution is found after termination of the column generation algorithm, we branch. We use the branching procedure in the package of Lysgaard (2003) to create two child nodes. In each iteration of the branching algorithm, we select a child of the node with the lowest lower bound to be processed next. In our experiments, we only consider instances with integer values of c_{ij} for all $(i, j) \in A$. Therefore, we terminate the branch-cut-and-price algorithm if the difference in value of the lower and upper bound is strictly less than 1, since this proves optimality.

G Proof of Proposition 12

We repeat Proposition 12 and provide a proof.

Proposition 12. *The undirected p-step formulation is valid for the CVRP.*

Proof. In this proof we demonstrate that an optimal solution to (1)-(3), (6), (8), and (10)-(11), is an optimal solution to the CVRP. It is easy to verify that every feasible solution of the CVRP can be transformed into an edge flow that is allowed by this formulation. Therefore, we limit ourselves to demonstrating that a binary edge flow, as described by the variables θ, corresponds to a feasible solution. The claim easily follows from these observations.

By (1), the degree of every node in N' is 2, so it is connected to precisely 2 edges. Hence, the edge flow can be decomposed in elementary paths from the starting depot 0 to the ending depot $n+1$, and potentially simple cycles not including the depot. Observe that an elementary path from 0 to $n+1$ can only be formed if p-steps are selected which follow that path in the direction from 0 to $n+1$, and no p-steps in the reverse direction can be selected. This is due to (3) and the fact no p-steps terminate at 0, or start at $n+1$. By (6), the capacity constraints are satisfied and such paths correspond to feasible routes.

In case of a simple cycle, let $C \subseteq N'$ be the nodes visited by the cycle. Observe that any selected p-step that is part of this cycle has its start and end point on the cycle. Using (6) this allows us to derive

$$\begin{aligned} 0 &\leq \sum_{i\in C}\sum_{r\in R^p} q_r^i x_r \\ &= \sum_{r\in R^p} \left(d_r + q_i - d_r - q(r)\right) x_r. \end{aligned}$$

Because $q_i - q(r) < 0$, this shows that such a cycle cannot exist. □

H LP bounds for benchmark instances

Tables 4 and 5 provide the LP bounds of the p-step formulation using R^p_{CG} and 2-cycle elimination for benchmark instances. For each instance, we have computed the maximum number of customers that would fit in one vehicle without violating the capacity constraint. The LP bounds are computed for all values of p from 1 through that maximum plus one. For higher values of p the LP bound remains constant, and equal to the set partitioning bound. Per instance, for higher values of p the corresponding cells in the tables are empty. Table 4 shows the results for p is 1 through 16, while Table 5 shows the results for p is 17 through 35. Note that a dash indicates that the LP bound could not be computed within the time limit of 3600 seconds using our algorithm.

I Computation times of LP bounds

Tables 6 and 7 provide the computation time in seconds of computing the LP bounds of the p-step formulation using R^p_{CG} and 2-cycle elimination for benchmark instances. The times correspond to our column generation algorithm when solving the pricing problems in parallel using up to 8 threads. Like in Appendix H, per instance we only consider the values of p from 1 through the maximum number of customers that fit in a vehicle plus one. Per instance, for higher values of p the corresponding cells in the tables are empty. Table 6 shows the results for p is 1 through 16, and Table 7 shows the results for p is

Table 4: LP bounds using R^p_{CG} and 2-cycle elimination (part 1).

Instance \ p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
E-n13-k4	239.28	239.71	243.16	247.00	247.00											
E-n22-k4	349.97	350.52	354.49	359.73	364.83	369.76	372.36	373.88	373.88	373.88	373.88					
E-n23-k3	529.88	531.76	535.87	538.82	543.12	540.37	544.64	544.99	546.62	550.55	550.55	552.53	557.20	559.20	558.71	559.19
E-n30-k3	448.74	449.80	451.30	454.78	457.74	458.56	459.00	460.53	463.92	468.26	471.35	474.09	478.26	484.27	484.27	484.27
E-n31-k7	363.30	363.57	368.35	370.18	368.22	373.29	377.49	378.27	378.29	378.29	378.29	378.29	378.29	378.29	378.29	378.29
E-n33-k4	784.44	785.81	793.50	798.18	802.29	809.07	811.80	813.61	816.88	817.75	819.42	821.81	822.62	822.67	822.77	822.77
E-n51-k5	499.43	499.64	500.52	504.12	503.92	506.25	507.69	509.79	511.88	514.26	515.16	516.72	517.13	517.14	517.14	517.14
E-n76-k7	644.30	644.75	646.81	649.02	651.32	653.91	655.13	656.28	657.22	658.64	660.67	662.37	663.95	665.03	665.45	665.58
E-n76-k8	690.47	691.87	695.60	698.90	703.04	706.44	708.55	711.30	712.97	715.13	716.76	717.89	718.50	718.78	718.78	718.78
E-n76-k10	771.77	773.74	778.71	786.30	792.33	799.30	802.50	804.43	807.75	810.39	812.44	812.47	812.47	812.47	812.47	812.47
E-n76-k14	933.81	938.38	947.28	965.72	981.03	991.19	998.77	1000.87	1002.77	1002.77	1002.77	1002.77	-	1002.77		
E-n101-k8	768.92	769.51	771.19	773.27	773.92	775.09	776.25	779.04	780.87	782.45	783.30	784.28	-	-	-	-
E-n101-k14	994.27	997.52	1005.93	1014.99	1023.81	1029.34	1036.16	1041.91	1045.91	1048.86	1050.08	1050.42	1050.42	1050.42	1050.42	1050.42
A-n32-k5	708.88	712.64	716.72	729.30	736.07	748.61	752.55	756.41	765.18	768.64	769.95	770.29	770.29	770.29	770.29	
A-n33-k5	596.71	600.70	608.58	619.52	626.99	631.90	644.38	651.44	655.37	656.21	656.21	656.21	656.21	656.21		
A-n33-k6	666.01	666.72	675.19	686.19	694.79	712.27	722.05	729.21	732.10	732.10	732.10	732.10	732.10			
A-n34-k5	683.50	686.61	695.96	709.86	717.16	724.12	733.96	736.93	748.25	748.96	748.96	748.96	748.96	748.96	748.96	
A-n36-k5	729.31	731.84	737.40	745.80	752.57	757.24	758.51	772.68	775.86	776.94	778.38	778.70	778.70	778.70	778.70	778.70
A-n37-k5	610.67	612.90	618.19	622.10	630.02	632.53	636.72	641.97	648.31	653.73	657.26	658.25	658.34	658.34	658.34	658.34
A-n37-k6	855.24	859.80	863.24	876.52	889.03	897.43	907.72	923.89	927.52	928.18	928.34	928.34	928.34	928.34	928.34	
A-n38-k5	644.53	647.64	654.01	660.93	663.28	672.80	682.28	690.54	694.70	697.19	699.17	699.17	699.17	699.17	699.17	699.17
A-n39-k5	744.18	750.15	759.86	763.93	771.28	775.59	778.34	793.08	799.05	801.31	801.56	801.56	801.56	801.56	801.56	801.56
A-n39-k6	745.89	747.97	757.32	764.63	768.72	777.25	785.14	798.76	807.17	809.40	809.44	809.44	809.44	809.44	809.44	809.44
A-n44-k6	846.81	855.58	874.46	883.74	890.88	900.50	911.07	918.14	924.19	927.62	927.64	927.64	927.64	927.64	927.64	927.64
A-n45-k6	827.67	833.70	841.93	854.43	866.79	886.75	900.26	912.13	924.32	930.74	932.00	932.00	932.00	932.00	932.00	-
A-n45-k7	1044.23	1054.72	1067.35	1075.73	1089.26	1098.27	1111.28	1120.44	1124.78	1124.78	1124.78	1124.78	1124.78	1124.78	1124.78	1124.78
A-n46-k7	837.80	844.23	851.62	860.86	867.41	875.10	889.79	901.71	905.47	907.73	907.84	907.84	907.84	907.84	907.84	907.84
A-n48-k7	976.60	983.65	994.14	1006.12	1023.23	1037.81	1046.03	1048.14	1050.47	1051.53	1052.01	1053.92	1053.92	1053.92	1053.92	1053.92
A-n53-k7	917.30	922.48	929.88	936.62	951.54	964.99	972.04	980.34	987.33	992.00	995.36	995.45	995.47	995.53	995.53	995.53
A-n54-k7	1050.67	1058.83	1069.39	1078.35	1088.39	1104.91	1115.24	1125.71	1134.27	1138.06	1141.35	1141.73	1141.73	1141.73	1141.73	1141.73
A-n55-k9	965.36	970.20	979.73	989.56	1000.70	1020.66	1036.03	1050.19	1058.82	1060.33	1060.33	1060.33	1060.33	1060.33	1060.33	
A-n60-k9	1211.70	1221.32	1230.73	1245.34	1260.53	1277.96	1295.34	1306.67	1319.52	1327.58	1327.72	1327.72	1327.72	1327.72	1327.72	1327.72
A-n61-k9	946.24	948.45	958.90	965.87	972.23	980.54	991.79	1003.59	1011.15	1012.40	1013.15	1013.15	1013.15	1013.15	-	-
A-n62-k8	1146.73	1155.47	1165.05	1172.87	1184.73	1194.88	1209.60	1229.44	1241.41	1246.91	1248.92	1251.65	1254.33	1254.83	1254.83	1254.83
A-n63-k9	1488.41	1497.92	1510.26	1521.71	1531.14	1542.91	1565.50	1577.20	1584.57	1586.02	1587.23	1588.16	1588.16	1588.16	1588.16	1588.16
A-n63-k10	1180.50	1186.27	1195.79	1214.52	1230.82	1252.87	1267.31	1275.89	1284.10	1286.53	1286.83	1286.83	1286.83	1286.83	1286.83	1286.83
A-n64-k9	1277.39	1283.63	1292.05	1308.15	1327.87	1348.47	1360.83	1367.46	1373.14	1374.47	1376.05	1376.54	1376.90	1376.90	1376.90	1376.90
A-n65-k9	1082.16	1085.01	1094.17	1101.40	1112.72	1124.52	1131.27	1140.05	1146.25	1150.09	1150.15	1150.15	1150.15	1150.15	1150.15	1150.15
A-n69-k9	1054.18	1060.52	1068.25	1076.94	1085.00	1103.66	1111.86	1117.05	1124.27	1126.10	1129.65	1130.82	1131.34	1131.34	1131.34	1131.34
A-n80-k10	1614.62	1625.59	1634.50	1651.53	1668.52	1685.26	1701.43	1716.98	1724.37	1727.00	1728.37	1729.25	1730.05	1730.59	1731.38	1731.58

Table 5: LP bounds using R^p_{CG} and 2-cycle elimination (part 2).

Instance \ p	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
E-n13-k4																			
E-n22-k4																			
E-n23-k3	563.57	565.32	565.32	565.32															
E-n30-k3	484.27	484.27	484.27	484.27	484.27	484.27	484.27												
E-n31-k7																			
E-n33-k4	822.77	822.77	822.77																
E-n51-k5	517.14	517.14	517.14	517.14															
E-n76-k7	665.58	665.58	665.58	-	665.58	665.58	665.58	665.58											
E-n76-k8	718.78	718.78	718.78	-	718.78														
E-n76-k10	812.47	812.47																	
E-n76-k14																			
E-n101-k8	790.97	790.99	790.99	790.99	790.99	790.99	790.99	790.99	790.99	790.99	790.99	790.99	-	790.99	-	790.99	790.99	-	790.99
E-n101-k14	1050.42	1050.42	1050.42	1050.42	1050.42	1050.42	1050.42	-	1050.46										
A-n32-k5																			
A-n33-k5																			
A-n33-k6																			
A-n34-k5																			
A-n36-k5																			
A-n37-k5	658.34	658.34	658.34	658.34	658.34														
A-n37-k6																			
A-n38-k5																			
A-n39-k5	801.56	801.56	801.56																
A-n39-k6	809.44																		
A-n44-k6																			
A-n45-k6	932.00																		
A-n45-k7																			
A-n46-k7	907.84	907.84																	
A-n48-k7	1053.92																		
A-n53-k7	995.53	995.53	995.53	-	995.53	995.53	-	995.56											
A-n54-k7	1141.73	1141.73	1141.73	1141.73	1141.73														
A-n55-k9																			
A-n60-k9	1327.72	1327.72	1327.72	1327.72	1327.72	1327.72													
A-n61-k9	1013.15																		
A-n62-k8	1254.83	1254.83	1254.83	1254.83	1254.83	1254.83	-	-	1254.83										
A-n63-k9	1588.16	1588.16	1588.16	1588.16	1588.16														
A-n63-k10	1286.83	1286.83	1286.83	1286.83															
A-n64-k9	1376.90	-	-	-	-	1376.90													
A-n65-k9	1150.15	1150.15	1150.15	1150.15															
A-n69-k9	1131.34	1131.34	1131.34	1131.34	1131.34	1131.34	1131.34												
A-n80-k10	1731.58	-	-	-	-	-	-	-	-	-	-								

17 through 35. Again, a dash indicates that the LP bound could not be computed within the time limit of 3600 seconds.

J Computation times of branch-price-and-cut

Tables 8 and 9 provide the computation time in seconds of computing the optimal solutions to the CVRP for benchmark instances. The computation times correspond to our branch-price-and-cut algorithm in which we use the p-step formulation with R^{p}_{CG} and 2-cycle elimination and in which we solve the pricing problems in parallel using up to 8 threads. Using our implementation, we are not able to solve all E and A instances for all values of p. Therefore, we only report the computation times for all values of p and all instances with no more than 51 nodes. For larger instances, the number of combinations of instances and values of p that we can solve within the time limit is too low to draw meaningful conclusions. Like in Appendix H, per instance we only consider the values of p from 1 through the maximum number of customers that fit in a vehicle plus one. Per instance, for higher values of p the corresponding cells in the tables are empty. Table 8 shows the results for p is 1 through 16, and Table 9 shows the results for p is 17 through 23. Again, a dash indicates that the LP bound could not be computed within the time limit of 3600 seconds.

Table 6: Computation times in seconds of LP bounds using R_{CG}^{p} and 2-cycle elimination with 8 threads (part 1).

Instance \ p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
E-n13-k4	0.01	0.02	0.02	0.02	0.02											
E-n22-k4	0.02	0.04	0.10	0.13	0.10	0.11	0.08	0.09	0.11	0.09	0.09					
E-n23-k3	0.02	0.04	0.09	0.14	0.17	0.35	0.52	0.48	1.49	0.67	0.73	0.68	1.02	1.41	0.98	8.69
E-n30-k3	0.04	0.08	0.27	0.41	0.48	1.04	1.08	1.85	2.44	3.27	2.43	1.11	3.00	11.87	33.48	38.20
E-n31-k7	0.06	0.12	0.41	0.77	0.93	1.12	1.34	0.58	0.49	0.98	0.68	0.71	0.56	0.64	0.42	0.35
E-n33-k4	0.06	0.12	0.44	0.76	0.87	1.28	1.78	1.94	2.74	1.60	2.65	5.12	8.06	9.24	14.33	15.97
E-n51-k5	0.23	0.54	2.10	2.85	4.08	5.86	9.15	12.93	9.40	4.25	4.94	7.89	8.59	23.02	20.10	20.56
E-n76-k7	0.94	1.68	10.12	18.11	31.12	46.73	128.93	171.28	250.52	486.80	678.46	137.04	303.55	420.20	575.96	739.57
E-n76-k8	1.33	2.82	18.75	27.14	38.69	71.59	155.83	316.80	372.38	124.96	227.77	244.77	408.87	665.88	1450.61	1841.61
E-n76-k10	1.38	3.15	24.70	31.21	52.98	86.85	161.80	170.31	105.64	195.73	270.91	611.11	647.19	645.47	966.83	1379.00
E-n76-k14	0.82	1.48	9.89	26.22	34.07	79.12	97.51	158.76	206.01	307.57	386.18	972.40	-	394.40		
E-n101-k8	2.84	3.98	36.68	94.11	120.58	310.45	373.61	659.20	1072.92	1490.73	2212.64	2914.25	-	-	-	-
E-n101-k14	2.16	3.82	36.14	86.25	149.92	293.12	558.55	1405.04	1969.44	2495.22	1613.71	957.98	776.80	1019.05	889.97	1195.84
A-n32-k5	0.06	0.10	0.29	0.52	0.73	1.09	1.56	0.47	0.79	0.95	1.24	2.31	2.75	2.22	0.97	
A-n33-k5	0.06	0.11	0.35	0.64	0.92	0.95	0.41	0.65	0.79	1.59	1.09	1.00	1.12	0.89		
A-n33-k6	0.06	0.11	0.36	0.55	0.70	0.86	0.36	0.61	0.92	1.40	1.49	1.28	0.59			
A-n34-k5	0.06	0.10	0.43	0.68	0.86	1.25	0.64	0.59	1.22	1.97	2.38	1.83	2.00	1.21	1.05	
A-n36-k5	0.09	0.23	0.55	0.82	1.15	1.86	2.21	1.16	1.19	1.67	2.62	5.34	4.97	7.97	2.72	4.92
A-n37-k5	0.09	0.20	0.62	0.82	1.33	2.04	3.51	4.01	5.29	3.42	2.15	5.23	4.46	4.10	5.55	8.18
A-n37-k6	0.09	0.19	0.60	0.91	1.13	1.34	1.19	1.36	2.13	3.12	5.38	4.96	5.40	2.98	2.35	
A-n38-k5	0.10	0.19	0.58	0.94	1.18	2.41	2.86	1.35	1.60	2.56	2.64	3.05	4.33	3.05	2.22	1.28
A-n39-k5	0.11	0.24	0.80	1.14	1.53	2.73	3.33	9.45	7.28	3.33	6.44	10.34	14.27	10.35	10.69	7.80
A-n39-k6	0.12	0.21	0.65	1.02	1.50	1.95	3.28	2.73	1.35	2.64	4.25	4.58	5.93	6.76	4.97	2.71
A-n44-k6	0.16	0.33	0.91	1.60	2.27	4.17	4.63	2.10	2.59	4.59	10.69	6.31	5.12	4.48	3.63	3.41
A-n45-k6	0.17	0.31	1.08	1.51	2.45	5.87	5.35	9.11	7.49	12.51	26.90	40.98	48.52	54.62	1108.33	-
A-n45-k7	0.16	0.34	1.17	1.70	2.60	3.74	4.99	2.27	3.19	7.14	8.70	7.48	7.14	10.31	4.87	3.57
A-n46-k7	0.16	0.32	1.23	1.60	2.82	5.00	10.35	7.11	5.18	8.39	23.05	19.11	12.83	16.45	13.84	16.90
A-n48-k7	0.18	0.40	1.40	2.42	3.21	6.17	8.31	5.93	3.22	6.31	13.62	20.90	34.91	37.00	51.35	40.92
A-n53-k7	0.23	0.51	2.02	3.46	4.88	8.63	18.02	25.90	47.62	55.31	44.27	23.00	30.21	36.20	50.54	55.31
A-n54-k7	0.25	0.57	2.21	3.59	4.47	8.52	15.51	29.20	60.07	33.53	20.06	24.37	47.42	88.38	79.00	69.81
A-n55-k9	0.26	0.52	2.19	3.23	4.53	10.96	7.07	4.58	7.04	10.26	12.24	18.84	13.46	8.11	5.14	
A-n60-k9	0.32	0.71	3.05	5.72	8.28	20.71	35.69	77.59	109.98	56.71	34.58	70.66	98.06	94.73	117.64	94.79
A-n61-k9	0.39	0.68	3.09	6.45	7.71	20.91	43.78	69.08	60.61	110.90	200.29	262.82	294.91	445.02	-	-
A-n62-k8	0.38	0.85	3.48	6.33	10.20	19.36	34.80	70.10	116.23	236.93	117.83	118.07	62.15	108.80	134.69	240.92
A-n63-k9	0.38	0.82	3.73	6.95	11.29	31.20	52.60	128.95	171.02	122.96	160.37	165.47	177.35	315.55	479.10	771.82
A-n63-k10	0.42	0.77	3.56	5.83	10.72	25.77	53.71	108.84	106.05	59.81	106.24	180.04	302.84	195.22	359.22	132.66
A-n64-k9	0.49	0.90	4.12	7.82	14.72	31.18	78.30	220.44	248.32	204.69	136.93	231.73	280.62	438.27	544.16	786.56
A-n65-k9	0.48	0.95	4.08	6.19	12.37	35.86	78.91	147.07	240.72	121.31	136.23	170.93	239.45	313.04	347.17	588.43
A-n69-k9	0.57	1.08	5.28	10.35	18.64	40.86	82.89	256.51	440.99	687.58	323.02	243.28	508.49	550.74	696.96	608.27
A-n80-k10	1.03	1.78	9.71	20.30	41.04	132.04	167.06	632.15	1051.68	2202.50	2268.17	2700.78	1384.88	939.51	1159.98	1679.24

Table 7: Computation times in seconds of LP bounds using R_{CG}^{p} and 2-cycle elimination with 8 threads (part 2).

Instance \ p	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
E-n13-k4																			
E-n22-k4																			
E-n23-k3	5.39	24.84	11.63	4.41															
E-n30-k3	43.67	26.71	25.58	34.86	47.47	19.35	28.57												
E-n31-k7																			
E-n33-k4	88.16	242.75	9.22																
E-n51-k5	19.69	43.15	14.83	6.14															
E-n76-k7	1478.78	2177.19	1262.61	-	1472.52	2161.07	1222.76	439.86											
E-n76-k8	1241.56	2560.38	2320.39	-	386.97														
E-n76-k10	141.62	749.05																	
E-n76-k14																			
E-n101-k8	2463.47	1806.59	1852.54	3064.48	2796.91	2415.07	3017.80	2684.73	2853.03	-	2827.61	3581.12	-	2436.06	-	2906.81	3235.36	-	2543.59
E-n101-k14	2332.06	1470.69	1403.35	1466.92	1027.94	746.43	433.62	-	608.96										
A-n32-k5																			
A-n33-k5																			
A-n33-k6																			
A-n34-k5																			
A-n36-k5																			
A-n37-k5	9.15	4.51	4.26	4.52	3.94														
A-n37-k6																			
A-n38-k5																			
A-n39-k5	5.97	9.03	7.80																
A-n39-k6	1.98																		
A-n44-k6																			
A-n45-k6	124.75																		
A-n45-k7																			
A-n46-k7	10.64	9.54																	
A-n48-k7	9.75																		
A-n53-k7	50.16	96.60	41.33	-	43.23	25.97	-	17.53											
A-n54-k7	50.98	131.53	46.96	37.78	22.38														
A-n55-k9																			
A-n60-k9	99.76	94.45	70.52	40.03	34.32	33.14													
A-n61-k9	351.10																		
A-n62-k8	295.79	789.52	257.61	489.24	147.22	216.97	-	-	145.64										
A-n63-k9	877.51	2751.53	335.94	53.78	41.10														
A-n63-k10	978.22	120.29	740.76	34.75															
A-n64-k9	2264.90	-	-	-	-	696.48													
A-n65-k9	671.47	810.99	936.01	123.50															
A-n69-k9	875.33	677.49	1174.22	857.18	675.00	488.63	309.22												
A-n80-k10	3574.48	-	-	-	-	-	-	-	-	-	-								

Table 8: Computation times in seconds of branch-price-and-cut (part 1).

Instance \ p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
E-n13-k4	0.39	0.15	0.11	0.04	0.04											
E-n22-k4	0.08	0.17	0.28	0.35	0.50	0.33	0.17	0.13	0.17	0.14	0.16					
E-n23-k3	0.07	0.12	0.23	0.25	0.33	0.41	0.82	1.05	1.97	0.98	0.61	0.54	0.67	1.04	1.51	2.58
E-n30-k3	1.35	3.22	9.23	11.15	19.25	16.44	-	135.86	66.97	421.90	27.83	58.58	35.24	35.41	1003.54	1131.43
E-n31-k7	2.69	11.45	27.52	26.12	35.37	10.15	4.26	2.09	2.13	12.97	5.30	4.52	32.56	21.79	93.05	4.11
E-n33-k4	2.16	7.47	30.46	32.63	29.71	111.59	370.44	1044.97	671.39	105.20	1658.25	78.06	1436.07	557.23	1976.00	-
E-n51-k5	46.04	160.90	652.99	1121.26	2263.88	-	-	-	3560.33	397.15	1123.63	-	-	288.09	-	-
A-n32-k5	1.91	4.43	9.95	15.49	90.83	132.67	250.25	185.39	160.34	59.25	4.84	6.78	5.12	17.80	2.46	
A-n33-k5	2.70	16.00	54.11	108.11	197.50	163.07	143.09	37.30	1.74	5.23	8.43	4.39	1.78	0.83		
A-n33-k6	9.02	51.73	152.55	266.96	720.17	382.09	212.33	545.49	90.38	179.72	176.88	100.41	30.42			
A-n34-k5	5.14	44.13	188.33	401.28	549.06	765.89	393.73	138.02	225.78	136.36	384.08	152.92	973.53	247.48	94.16	
A-n36-k5	11.30	66.34	117.40	786.80	930.24	628.48	3506.70	529.47	617.02	1613.89	369.60	724.05	328.13	-	-	1658.04
A-n37-k5	5.64	27.72	74.92	134.19	257.75	339.20	325.17	611.12	1055.77	375.02	122.14	574.05	532.05	-	-	-
A-n37-k6	-	-	-	-	-	-	-	-	-	-	-	421.14	-	-	-	
A-n38-k5	135.73	219.37	987.62	1150.05	-	-	-	-	-	-	1130.51	1620.59	-	1076.58	-	410.16
A-n39-k5	97.94	1421.04	2879.73	-	-	-	-	-	1787.95	658.26	526.07	-	-	-	-	-
A-n39-k6	35.85	236.86	1669.17	1016.01	-	-	2559.58	-	802.59	1312.36	1668.27	-	2937.94	2807.15	-	-
A-n44-k6	-	-	-	-	-	-	-	-	764.51	505.55	781.42	1065.11	2535.41	2774.15	-	573.32
A-n45-k6	185.66	2244.43	-	-	-	-	-	-	-	1508.49	2549.20	259.88	-	-	-	-
A-n45-k7	-	-	-	-	-	-	-	-	-	-	3314.18	-	-	-	-	1202.43
A-n46-k7	50.22	282.84	448.38	1188.93	1563.43	3211.19	1172.10	668.85	51.00	37.91	13.80	53.61	33.72	265.43	128.50	2407.50
A-n48-k7	449.07	-	-	-	-	-	-	-	-	637.19	544.41	886.87	1728.42	1991.22	-	-

Table 9: Computation times in seconds of branch-price-and-cut (part 2).

Instance \ p	17	18	19	20	21	22	23
E-n13-k4							
E-n22-k4							
E-n23-k3	2.69	14.69	11.25	3.32			
E-n30-k3	403.02	201.62	101.58	293.18	-	1620.32	98.88
E-n31-k7							
E-n33-k4	-	-	962.13				
E-n51-k5	-	-	-	1835.04			
A-n32-k5							
A-n33-k5							
A-n33-k6							
A-n34-k5							
A-n36-k5							
A-n37-k5	3258.86	-	-	95.06	-		
A-n37-k6							
A-n38-k5							
A-n39-k5	-	-	-				
A-n39-k6	785.67						
A-n44-k6							
A-n45-k6	1150.90						
A-n45-k7							
A-n46-k7	210.92	16.96					
A-n48-k7	2095.93						