Transportation Science

An Exact Algorithm for the Green Vehicle Routing Problem

Juho Andelmin, Enrico Bartolini

http://pubsonline.informs.org/journal/trsc/

TRANSPORTATION SCIENCE
Articles in Advance, pp. 1–16
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

An Exact Algorithm for the Green Vehicle Routing Problem

Juho Andelmin,[a] **Enrico Bartolini**[b]

[a] Department of Mathematics and Systems Analysis, Aalto University School of Science, 00076 Aalto, Finland; [b] School of Business and Economics, RWTH Aachen University, 52062 Aachen, Germany

Contact: juho.andelmin@aalto.fi (JA); bartolini@dpo.rwth-aachen.de (EB)

Received: November 3, 2015
Revised: August 2016
Accepted: September 21, 2016
Published Online in Articles in Advance: July 6, 2017

https://doi.org/10.1287/trsc.2016.0734

Copyright: © 2017 INFORMS

Abstract. We propose an exact algorithm for solving the green vehicle routing problem (G-VRP). The G-VRP models the optimal routing of an alternative fuel vehicle fleet to serve a set of geographically scattered customers. Vehicles' fuel autonomy and possible refueling stops en route are explicitly modeled and maximum duration constraints are imposed on each vehicle route. We model the G-VRP as a set partitioning problem in which columns represent feasible routes corresponding to simple circuits in a multigraph. Each node in the multigraph represents one customer and each arc between two customers represents a nondominated path through a set of refueling stations visited by a vehicle when traveling directly between the two customers. We strengthen the set partitioning formulation by adding valid inequalities including k-path cuts and describe a method for separating them. We provide computational results on benchmark instances showing that the algorithm can optimally solve instances with up to ~110 customers.

Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2016.0734.

Keywords: vehicle routing • column generation • k-path cuts • green logistics

1. Introduction

Vehicle routing problems (VRPs) constitute a major class of combinatorial optimization problems that are concerned with the design of least-cost delivery routes for a fleet of vehicles to service a set of geographically scattered customers.

Even though several VRP variants have been extensively studied in the past 60 years, most of the models that have been considered are best suited for gasoline- and diesel-powered vehicles with a long driving range, relatively short refueling time, and widespread fueling infrastructure. Issues related to fuel consumption, refueling delays, and their interplay with other operational constraints that characterize real-life applications have been mostly overlooked, and the majority of the VRP variants incorporate neither the locations of the refueling stations nor the refueling occurrences.

Recently, a growing interest toward "green transportation" and a strong tendency to reduce fossil-fuel use has spawned renewed attention to vehicles that use alternative fuel sources, such as biodiesel and electricity. However, as opposed to conventional internal combustion engine vehicles, alternative fuel vehicles (AFVs) typically have limited fuel autonomy, relatively long refueling delays, and must often rely on a limited fueling infrastructure. The adoption of AFVs poses new challenges to transportation planning and motivates the study of tailored routing models that capture the limitations arising in connection with their operation.

In this paper, we consider one such model called the green vehicle routing problem (G-VRP) introduced by Erdoğan and Miller-Hooks (2012), which captures some of these key issues. Given a set of customers, a set of refueling stations, and a fleet of AFVs located at a central depot, the G-VRP consists of determining a set of vehicle routes, each starting and ending at the depot, so that each customer is visited exactly once and the total distance traveled by the vehicles is minimized. The vehicles have a maximum driving range as a consequence of their limited fuel autonomy. A distinguishing feature of the G-VRP is the presence of *fuel constraints*. Such constraints impose that the total distance traveled by a vehicle can be extended over the maximum driving range only by stopping en route at any of the available refueling stations, but a constant refueling delay is incurred at each stop. The vehicle routes are further subject to maximum duration constraints that limit the total driving time of each vehicle. The model captures the trade-off between high fuel levels and short driving times.

The G-VRP may be considered a basic model for AFV routing. It was motivated by a VRP involving a fleet of vehicles running on biodiesel, but it is also relevant to vehicles based on other alternative fuels such as ethanol, hydrogen, and natural gas. Interestingly, the G-VRP can also be applied to the routing of battery electric vehicles (BEVs) with replaceable batteries where nearly depleted vehicle batteries can be replaced with fully charged ones (Avci, Girotra, and Netessine 2014). Each battery swap fully restores the vehicle driving range but incurs a fixed service time that corresponds precisely to the refueling delay in the G-VRP.

1.1. Literature Review and Related Problems

The G-VRP appears to be one of the first routing models that incorporates refueling stops as part of the route planning, although from a combinatorial point of view it is closely related to VRPs with satellite facilities (see, e.g., Bard et al. 1998; Crevier, Cordeau, and Laporte 2007). Early studies with a similar focus include the work of Ichimori, Ishii, and Nishida (1981, 1983) who concentrated on the routing of a single vehicle with a limited fuel autonomy initially located at a depot. More recently, Conrad and Figliozzi (2011) investigate the routing of BEVs and present the recharging VRP, which extends the VRP with time windows by allowing the BEVs to recharge at the customer locations. However, recharging is permitted only upon service, and the model is not applicable to more general cases where recharging stations are located apart from the customers and can be visited more than once. For recent surveys of different "green" VRPs and an analysis of past and future trends, we refer the reader to the works of Lin et al. (2014) and Pelletier, Jabali, and Laporte (2016).

Several heuristic solution methods have recently been proposed for solving the G-VRP or extensions of it, which incorporate refueling stops as part of the route planning.

Erdoğan and Miller-Hooks (2012) propose two heuristics for solving the G-VRP: a modified Clarke and Wright savings algorithm and a density-based clustering algorithm. Montoya et al. (2016) also study the G-VRP and present a heuristic that first generates traveling salesman tours using constructive heuristics, then partitions the tours into feasible vehicle routes, and finally constructs a solution by solving a set partitioning problem over the vehicle routes.

Schneider, Stenger, and Goeke (2014) introduce the electric VRP with time windows (EVRPTW), which extends the G-VRP by including time window and vehicle capacity constraints, but ignores the maximum route duration constraint. A new recharging scheme is also applied in which the recharging time depends on the battery level on arrival at a station. Felipe et al. (2014) investigate the G-VRP with multiple technologies and partial recharges, which generalizes the G-VRP by including vehicle capacities, partial recharges, and different (but constant) recharging speed. The objective is to minimize the total recharging cost, which consists of a fixed and a variable cost component based on the recharged amount. In a more recent study, Schneider, Stenger, and Hof (2015) present the VRP with intermediate stops, which includes both refueling stations and satellite facilities. The vehicles have a limited capacity and fuel autonomy, and both can be replenished en route by visiting a satellite facility or a refueling station, respectively. Goeke and Schneider (2015) study a VRP with a mixed fleet of electric and conventional vehicles in which the vehicles' energy consumption depends on their curb mass and the carried load, vehicle speed, and gradient of the terrain. Recharging visits for BEVs are also included in the model. Finally, Hiermann et al. (2016) introduce a further extension of the EVRPTW called the electric fleet size and mix vehicle routing problem with time windows and recharging stations (E-FSMFTW). Their model considers vehicles of different types with respect to transport capacity, battery size, and cost.

Exact methods for the G-VRP and its generalizations are rather scarce. Recently, Koç and Karaoglan (2016) presented a new mathematical formulation of the G-VRP based on the assumption that no more than one refueling stop is made by any vehicle when traveling between two customers (or the depot and one customer). They used it as a basis to develop a branch-and-cut algorithm that was computationally evaluated on instances with up to 20 customers. To our knowledge, Desaulniers et al. (2016) and Hiermann et al. (2016) are the only ones to develop exact methods for a class of VRPs similar to the G-VRP and to report computational results for larger instances. Desaulniers et al. (2016) study the EVRPTW and present a branch-price-and-cut algorithm for four variants of it. They report optimal solutions to problem instances with up to 100 customers and 21 recharging stations. Hiermann et al. (2016) develop an exact branch-and-price algorithm and a heuristic algorithm for the E-FSMFTW and report computational results on both E-FSMFTW and EVRPTW instances. However, neither Desaulniers et al. (2016) nor Hiermann et al. (2016) considered the G-VRP.

1.2. Contributions of This Paper

In this paper, we develop an exact algorithm for the G-VRP based on a set partitioning formulation in which columns correspond to feasible routes. We model the problem by using a multigraph in which nodes correspond to customers and each arc represents one possible sequence of refueling stations visited by a vehicle when traveling from one customer to another. We strengthen the formulation by adding three classes of valid inequalities: weak subset row inequalities, subset row inequalities, and k-path cuts. We show that a subset of the k-path cuts correspond to Chvátal–Gomory (CG) cuts of rank one. We use this observation to formulate the corresponding separation problem as a mixed-integer linear programming (MILP) problem and describe a method to incorporate such cuts within a cut-and-column generation algorithm. We report a computational evaluation of the exact algorithm on a set of instances introduced by Erdoğan and Miller-Hooks (2012) as well as new ones. Our results show that the incorporation of k-path cuts is crucial to the effectiveness of our algorithm and permits to optimally solve instances with up to ~110 customers and 28 refueling stations.

1.3. Organization of This Paper

The remainder of the paper is organized as follows. In Section 2 we describe the G-VRP, we introduce the main notation used in the paper, and we describe the multigraph we use to model the problem. In Section 3, we present the set partitioning formulation, and in Section 4 we describe the valid inequalities that we use to strengthen it. The exact algorithm consists of two phases detailed in Section 5. Phase I of the algorithm embeds three lower bounding procedures based on column generation that are executed in sequence to compute a lower bound on the optimal G-VRP cost. Section 6 describes a dynamic programming algorithm that is used by the lower bounding procedures and the exact algorithm to generate least cost G-VRP routes. Section 7 reports a computational evaluation of the exact algorithm, and concluding remarks are given in Section 8.

2. Description of the G-VRP

The G-VRP may be defined on a complete directed graph $G = (V, A)$, where the vertex set $V = N \cup F \cup \{0\}$ consists of a subset $N = \{1, \ldots, n\}$ of n customers, a subset $F = \{n+1, \ldots, n+s\}$ of s refueling stations, and one vertex 0 representing the depot. Each customer $i \in N$ is associated with a service time ς_i, and each arc $(i, j) \in A$ is associated with a distance c_{ij} and a travel time t_{ij}.

A homogeneous fleet of AFVs based at the depot 0 is to visit the n customers. Each vehicle has a maximum driving time of T minutes and a limited fuel autonomy that is expressed as a fuel capacity C. The rate of fuel consumption per distance unit of each vehicle is assumed to be a constant K; thus, the maximum distance that a vehicle can travel without refueling is defined as $Q = C/K$. The vehicles can, however, stop en route at any of the s refueling stations. After stopping at a station, the vehicle is assumed to be fully refueled, but each stop is assumed to incur a refueling delay of δ minutes (it is also assumed that each vehicle incurs such a delay before leaving the depot for the first time). Since refueling stations can be visited more than once, the set F is usually replaced by a larger set F' of "dummy" vertices, each one representing one potential refueling station visit rather than the station itself (Bard et al. 1998, Erdoğan and Miller-Hooks 2012, Schneider, Stenger, and Goeke 2014). In the resulting augmented graph G' each customer vertex $i \in N$ must be visited exactly once, whereas each vertex $i \in F'$ can be visited at most once by the same vehicle.

All vehicles are assumed to travel at a constant speed v and the arc travel times are derived accordingly from the arc distances. To simplify the notation, we include also the service and refueling times in the arc travel times. Thus, we assume that the travel times of the arcs are defined as $t_{ij} = c_{ij}/\mathrm{v} + \varsigma_i$ if $i \in N$ and $t_{ij} = c_{ij}/\mathrm{v} + \delta$ if $i \in F \cup \{0\}$. Each vehicle can be assigned a *route* representing a vehicle trip starting and ending at the depot and visiting a subset of the customers. A route corresponds to a simple circuit in the augmented graph G' starting and ending at the depot 0 and satisfying the following constraints: (i) the sum of the travel times of the arcs traversed by the route does not exceed T, and (ii) any path with total distance (i.e., sum of distances of its arcs) bigger than Q that belongs to the route visits at least one station.

The G-VRP may then be defined as the problem of finding a set of routes of minimum total distance such that each customer $i \in N$ is visited by exactly one route. It is worth noting that although Erdoğan and Miller-Hooks (2012) defined the G-VRP for a limited vehicle fleet, their experiments were all conducted on instances with an unlimited vehicle fleet, and the same instances have been used in all subsequent studies. Therefore, we do not consider any constraint on the number of vehicles.

2.1. Description of the Multigraph

In this section, we describe an alternative model used by our algorithm that does away with the "dummy" vertices. Let $N_0 = N \cup \{0\}$ and let $G_F = (F, A_F)$ be the subgraph of G induced by the subset F of vertices representing the refueling stations where $A_F = \{(i, j) \in A\colon i, j \in F,\ c_{ij} \le Q\}$. For any pair $i, j \in N_0$, we call *refuel path* a simple path in G starting from i, traversing a path in G_F, and ending at j. A refuel path P is associated with a travel time $t(P)$ and a distance $c(P)$, which are equal to the sum of the travel times and distances, respectively, of the arcs it traverses.

We denote by $\mathcal{P}_{ij}$ the index set of all of the refuel paths between two vertices $i \in N_0$ and $j \in N_0$, and define an extended arc set $\mathcal{A}$, which contains one arc (i, j, p) from i to j for each possible path $p \in \mathcal{P}_{ij}$. For notational convenience, we assume that the arc $(i, j) \in A$ is represented by the path index $p = 0$ so that the arc $(i, j, 0) \in \mathcal{A}$ corresponds to the arc $(i, j) \in A$. Each arc $(i, j, p) \in \mathcal{A}$ has a distance $c(i, j, p)$ and a travel time $t(i, j, p)$, which are equal to the total distance and travel time, respectively, of the corresponding path $p \in \mathcal{P}_{ij}$ (where we assume $c(i, j, 0) = c_{ij}$ and $t(i, j, 0) = t_{ij}$). Moreover, we denote by $c_1(i, j, p)$ and $c_2(i, j, p)$ the distances of the first and last arc, respectively, traversed by the path $p \in \mathcal{P}_{ij}$, $p \neq 0$.

In this way, we obtain in this way a multigraph $\mathcal{G} = (N_0, \mathcal{A})$, which contains only one node for each customer plus one depot node. Any circuit in G' representing a vehicle route corresponds to an equivalent circuit in $\mathcal{G}$ having the same total distance and travel time; thus, the G-VRP can be modeled on the multigraph $\mathcal{G}$. Even though its arc set $\mathcal{A}$ can in principle contain a large number of arcs, most of them cannot be part of an optimal solution.

Dominance 1. *Let $P = (i, l, \ldots, k, j)$ be a refuel path from $i \in N_0$ to $j \in N_0$ traversing the subpath $(l, \ldots, k)$ in G_F from $l \in F$ to $k \in F$ (possibly with $l = k$). P is said to be dominated if there exists another refuel path $P' = (i, l', \ldots, k', j)$ traversing a subpath $(l', \ldots, k')$ in G_F from $l' \in F$ to $k' \in F$ (possibly with $l' = k'$) such that*

$$\text{(i) } c_{il'} \le c_{il} \qquad \text{(ii) } c_{k'j} \le c_{kj}$$
$$\text{(iii) } c(P') \le c(P) \qquad \text{(iv) } t(P') \le t(P)$$

and at least one of the inequalities is strict.

It is easy to see that there exists at least one optimal G-VRP solution in which each refuel path traversed by a vehicle is nondominated. Thus, we can assume that the arc set $\mathscr{A}$ contains only arcs (i, j, p) corresponding to nondominated refuel paths plus the arcs $(i, j, 0)$, $\forall\, i, j \in N_0$. The set of all such paths can be computed a priori as detailed in Section EC.1.2 of the online appendix. In the following, we thus assume that the sets $\mathscr{P}_{ij}$ index only nondominated refuel paths. In Section EC.1.1 of the online appendix we also describe some arc elimination rules (see, e.g., Schneider, Stenger, and Goeke 2014) that can be used to identify infeasible arcs, i.e., arcs that cannot belong to any feasible solution. Such arcs are removed from G before constructing the multigraph $\mathscr{G}$. It is worth noting that the size of $\mathscr{G}$ (even without removing all dominated paths or applying any arc elimination rules) is still polynomial in the number of customers and refueling stations. Indeed, an upper bound on the number of arcs of $\mathscr{G}$ is $(n+1)^2 s^2 \kappa$ where $\kappa \le s$ is the maximum cardinality of a shortest path in G_F between any two stations, and s is the number of stations (see Section EC.1.2 of the online appendix). Although an upper bound on the number of arcs of the augmented graph G' is $(n + s(2n) + 1)^2$ (since a refueling station can be visited up to $2n$ times), in practice the use of Dominance 1 and the application of the arc elimination rules reduces the number of arcs in the multigraph $\mathscr{G}$ well below the number of arcs of G'. In Section EC.1.3 of the online appendix we provide a more detailed analysis of the multigraph $\mathscr{G}$ and a numerical comparison of the number of arcs in $\mathscr{G}$ and G'.

Finally, we note that a similar idea underlying our multigraph construction was introduced by Koç and Karaoglan (2016). These authors proposed to model explicitly the refueling paths traversed by a vehicle between two customers, but under the assumption that at most one refueling stop can be made between two customers. This idea was used to develop a mathematical formulation of the G-VRP involving two sets of binary variables indicating for each pair of nodes i and j if a vehicle travels directly from i to j, or from node i to node j via a station k, respectively.

In Section 3, we describe a set partitioning formulation of the G-VRP where each variable corresponds to a simple circuit in $\mathscr{G}$ representing a route.

3. Set Partitioning Formulation

A vehicle route R can be defined as a simple circuit $(a_0, a_1, \ldots, a_{r-1}, a_r)$ in $\mathscr{G}$ starting from the depot 0, traversing arcs $a_k = (i_k, i_{k+1}, p_k)$, $p_k \in \mathscr{P}_{i_k i_{k+1}}$, $k = 0, \ldots, r$, and returning to the depot. The *travel time* $t(R)$ of route R is defined as $t(R) = t(i_0, i_1, p_0) + \cdots + t(i_r, i_{r+1}, p_r)$, and its *cost* $c(R)$ is defined as the sum of the distances of the arcs it traverses. The *driving autonomy* $q_k(R)$ of route R on arrival at a vertex i_k, $k = 0, \ldots, r+1$, is defined as follows:

$$q_0(R) = Q, \quad \text{and}$$
$$q_k(R) = \begin{cases} q_{k-1}(R) - c(i_{k-1}, i_k, p_{k-1}) & \text{if } p_{k-1} = 0, \\ Q - c_2(i_{k-1}, i_k, p_{k-1}) & \text{otherwise.} \end{cases}$$

The driving autonomy $q_k(R)$ indicates how many distance units R can travel without refueling after leaving i_k. A route R is *feasible* if it satisfies the following conditions:

1. $t(R) \le T$.
2. $0 \le q_k(R) \le Q$, $k = 0, \ldots, r+1$.
3. $q_k(R) \ge c_1(i_k, i_{k+1}, p_k)$, $k = 0, \ldots, r$.

Let $\mathscr{R}$ be the index set of all of the feasible routes and let $\theta_{i\ell}$ be a 0–1 coefficient that equals one if route R_ℓ visits customer $i \in N$ and zero otherwise. For each route R_ℓ, $\ell \in \mathscr{R}$, we denote by $c_\ell = c(R_\ell)$ its cost and by $A(R_\ell) \subseteq \mathscr{A}$ the subset of arcs of $\mathscr{G}$ that it traverses. We will use R_ℓ also to denote the set of vertices visited by the route $\ell \in \mathscr{R}$. By defining a 0–1 variable x_ℓ taking value one if and only if the route R_ℓ is in solution, $\forall\, \ell \in \mathscr{R}$, the G-VRP can be modeled as the following Set Partitioning problem (SP):

$$[\text{SP}] \quad z_{\text{SP}} = \min \sum_{\ell \in \mathscr{R}} c_\ell x_\ell \tag{1}$$
$$\text{s.t.} \quad \sum_{\ell \in \mathscr{R}} \theta_{i\ell} x_\ell = 1, \quad \forall\, i \in N, \tag{2}$$
$$x_\ell \in \{0, 1\}, \quad \forall\, \ell \in \mathscr{R}. \tag{3}$$

In the following, we will denote by LSP the Linear Programming relaxation of SP. Note that solving SP directly is impractical because it involves an exponential number of variables. In Section 5, we describe an exact algorithm based on cut-and-column generation for solving SP.

4. Valid Inequalities for SP

As shown in Section 7, the LP relaxation of SP can be rather weak. In this section, we describe three sets of valid inequalities that we have used to improve it, namely, *k-path cuts* (Laporte, Nobert, and Desrochers 1985), *subset row inequalities* (Jepsen et al. 2008), and *weak subset row inequalities* (Baldacci, Mingozzi, and Roberti 2011).

4.1. k-Path Cuts

Let $S \subseteq N$ be any subset of customers, and let $\eta_\ell(S)$ be a coefficient representing the number of arcs (i, j, p) with $i \notin S$ and $j \in S$, which are traversed by a route R_ℓ, $\ell \in \mathcal{R}$. Suppose it is known that, because of maximum route duration constraints, the minimum number of vehicles required to serve a customer subset S in any feasible SP solution is $r(S)$. Then, the k-path cuts are

$$\sum_{\ell \in \mathcal{R}} \eta_\ell(S) x_\ell \geq r(S), \quad \forall\, S \subseteq N\colon |S| \geq 2. \tag{4}$$

The k-path cuts were introduced by Laporte, Nobert, and Desrochers (1985) and embedded within a branch-and-cut algorithm for solving a VRP with both capacity and distance restrictions. However, this algorithm actually used a weaker version of the k-path cuts obtained by replacing the quantity $r(S)$ with a lower bound $l(S)$. Most computational studies we are aware of consider the special case of k-path cuts where $r(S) = 2$ (Kohl et al. 1999, Desaulniers et al. 2016), and studies making use of the k-path cuts with $k > 2$ are rather scarce. Recently, Bektaş and Lysgaard (2015) devised a separation algorithm for a weaker version of the k-path cuts in which $r(S)$ is replaced with a lower bound that is computed by means of cost shares. The resulting inequalities are then embedded within a branch-and-cut algorithm to solve a multiple traveling salesman problem with both minimum and maximum duration constraints. Desaulniers, Lessard, and Hadjar (2008) introduce a generalization of the k-path cuts called generalized k-path inequalities that can dominate the standard k-path cuts and partially take into account the increase in the right-hand side $r(S)$ occurring when certain edges crossing S are traversed by a vehicle. In this paper, we make use of k-path cuts with $k \geq 2$. The remainder of this section describes the method we use to separate them.

4.1.1. Separation of the Lifted k-Path Cuts. It is easy to see that the inequalities (4) can be strengthened by replacing $\eta_\ell(S)$ with $b_\ell(S) = \min\{\eta_\ell(S), 1\}$. The resulting inequalities are called *lifted k-path cuts* and are defined as follows:

$$\sum_{\ell \in \mathcal{R}} b_\ell(S) x_\ell \geq r(S), \quad \forall\, S \subseteq N. \tag{5}$$

In our algorithm, we will use the weaker k-path cuts (4) instead of the lifted ones above because the inequalities (4) do not modify the structure of the pricing problem when solving SP by cut-and-column generation. Nevertheless, the lifted k-path cuts (5) can be easier to separate than exactly than the weaker cuts (4). In the remainder of this section, we thus describe a method that is used by our algorithm to separate a subset of the lifted k-path cuts (5), and in Section 4.1.2 we describe how our algorithm derives the corresponding k-path cuts (4) that are actually added to LSP.

For any customer subset $S \subseteq N$ we define $\tilde{r}(S) \leq r(S)$ to be the minimum (possibly fractional) number of vehicles required to serve the customers subset S in any feasible LSP solution. We then call a set S and the corresponding inequality (5) 1-*fractional* if $r(S) - \tilde{r}(S) < 1$. Note that a lifted 2-path cut defined by a set S with fractional $\tilde{r}(S)$ is 1-fractional. Indeed, if the routes correspond to elementary circuits, the inflow $\sum_{i \in N_0 \setminus S} \sum_{j \in S} \sum_{\ell \in \mathcal{R}} \alpha_{ij\ell} x_\ell$ into S, where $\alpha_{ij\ell}$ is a 0–1 coefficient taking value one if and only if route R_ℓ visits vertex $j \in N_0$ right after $i \in N_0$, is at least 1 (Kohl et al. 1999). In the following, we show that any 1-fractional k-path cut (5) can be expressed as a CG cut of rank one, and we exploit this observation to formulate the separation problem for the family of such inequalities as an MILP.

Proposition 1. *Any* 1*-fractional k-path cut* (5) *is a CG cut of rank one with respect to formulation LSP.*

Proof. Let $\pi_i \in \mathbb{R}_+$, $\forall\, i \in N$, be a nonnegative coefficient associated with a customer i, and let $\pi_i = 0$, $\forall\, i \in N \backslash S$. Each LSP constraint (2) can be equivalently expressed by the two set packing and set covering constraints obtained by replacing "=" with "≤" and "≥," respectively. Given a customer subset S, consider the following inequality obtained by summing all of the set covering constraints associated with the customers in S, each one multiplied by the coefficient π_i of the corresponding customer $i \in S$

$$\sum_{i \in S} \pi_i \sum_{\ell \in \mathcal{R}} \theta_{i\ell} x_\ell \geq \sum_{i \in S} \pi_i. \tag{6}$$

By rearranging the terms and rounding up both sides of (6) we obtain

$$\sum_{\ell \in \mathcal{R}} \left\lceil \sum_{i \in S} \theta_{i\ell} \pi_i \right\rceil x_\ell \geq \left\lceil \sum_{i \in S} \pi_i \right\rceil. \tag{7}$$

Note that (7) is a CG cut of rank one for any $\boldsymbol{\pi} \in \mathbb{R}^n_+$ such that $\pi_i = 0$, $\forall\, i \in N \backslash S$. Take $\boldsymbol{\pi} \neq \mathbf{0}$ satisfying (i) $\sum_{i \in S} \theta_{i\ell} \pi_i \leq 1$, $\forall\, \ell \in \mathcal{R}$, and (ii) $\sum_{i \in S} \pi_i > r(S) - 1$. Then, (7) becomes

$$\sum_{\ell \in \mathcal{R}} b_\ell(S) x_\ell \geq r(S), \tag{8}$$

which corresponds to the lifted k-path cut defined by S. We next show that a vector $\boldsymbol{\pi} \in \mathbb{R}^n_+$ satisfying (i) and (ii) exists if and only if $r(S) - \tilde{r}(S) < 1$. Consider the LP

$$[\text{DS}] \quad z_{\text{DS}} = \max \sum_{i \in S} \pi_i \tag{9}$$

$$\text{s.t.} \quad \sum_{i \in S} \theta_{i\ell} \pi_i \leq 1, \quad \forall\, \ell \in \mathcal{R}, \tag{10}$$

$$\pi_i \geq 0, \quad \forall\, i \in S. \tag{11}$$

Note that a vector $\boldsymbol{\pi} \in \mathbb{R}_+^n$ satisfying (i) and (ii) exists if and only if $z_{DS} > r(S) - 1$. Denoting by x_ℓ the dual variables of the constraints (10), the dual of DS is

$$[\text{PS}] \quad z_{\text{PS}} = \min \sum_{\ell \in \mathcal{R}} x_\ell \tag{12}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathcal{R}} \theta_{i\ell} x_\ell \geq 1, \quad \forall\, i \in S, \tag{13}$$

$$x_\ell \geq 0, \quad \forall\, \ell \in \mathcal{R}. \tag{14}$$

Note that $x_\ell \leq 1$, $\forall\, \ell \in \mathcal{R}$, in any optimal PS solution, and thus $z_{PS} = \tilde{r}(S)$. Assuming that PS is feasible (since otherwise the corresponding G-VRP is infeasible), by strong duality we have $z_{PS} = z_{DS}$ and the required vector $\boldsymbol{\pi}$ exists if and only if $\tilde{r}(S) = z_{PS} = z_{DS} > r(S) - 1$. □

Given a fractional LSP solution $\bar{\mathbf{x}}$, from the observations above it follows that the separation problem for the 1-fractional lifted k-path cuts can be modeled as that of finding a most violated CG cut (7) of rank one defined by a CG multiplier vector $\boldsymbol{\pi} \in \mathbb{R}_+^n$ satisfying the additional constraints (i) and (ii). However, note that constraints (ii) are unnecessary in the sense that any CG cut defined by a vector $\boldsymbol{\pi}$ with support S and satisfying (i) but not (ii) is not maximally violated if S is 1-fractional because it is dominated by another cut. Thus, we look for violated 1-fractional lifted k-path cuts by separating violated CG cuts of rank one defined by a vector $\boldsymbol{\pi}$ satisfying (i). The problem of finding a most violated CG cut of rank one is NP-hard (Eisenbrand 1999), but it can be formulated and solved as an MILP (see Fischetti and Lodi 2007 for a detailed analysis). Given an LSP solution $\bar{\mathbf{x}}$ let $\bar{\mathcal{R}} = \{\ell \in \mathcal{R}: \bar{x}_\ell > 0\}$. In our case, maximally violated CG cuts of rank one satisfying (i) can be separated by solving the following:

$$[\text{SEP}] \quad z_{\text{SEP}} = \left\{ \max z - \sum_{\ell \in \bar{\mathcal{R}}} \bar{x}_\ell \, y_\ell \right\} \tag{15}$$

$$\text{s.t.} \quad \sum_{i \in N} \theta_{i\ell} \pi_i \leq y_\ell, \quad \forall\, \ell \in \bar{\mathcal{R}}, \tag{16}$$

$$\sum_{i \in N} \theta_{i\ell} \pi_i \leq 1, \quad \forall\, \ell \in \mathcal{R}, \tag{17}$$

$$0 \leq z - \sum_{i \in N} \pi_i \leq 1 - \epsilon, \tag{18}$$

$$y_\ell \in \{0,1\}, \quad \forall\, \ell \in \bar{\mathcal{R}},$$

$$\pi_i \geq 0, \quad \forall\, i \in N \text{ and } z \in \mathbb{Z}_+, \tag{19}$$

where ϵ is a small tolerance parameter. An SEP solution $(\mathbf{y}, \boldsymbol{\pi}, z)$ with positive cost provides a violated inequality (5) defined by the customer subset $S = \{i \in N: \pi_i > 0\}$ with $r(S) = z$, if S is 1-fractional. Moreover, solving SEP may also provide other cuts defined by customer subsets S that are not 1-fractional. Such cuts correspond in fact to a weaker version of the lifted k-path cuts (5) with the right-hand side $z < r(S)$, but can still be violated by $\bar{\mathbf{x}}$ and can effectively be treated as regular lifted k-path cuts.

The problem SEP has an exponential number of constraints (17). The problem of finding a violated inequality (17) given a vector $(\mathbf{y}, \boldsymbol{\pi}, z)$ corresponds to the problem of finding a route $\ell \in \mathcal{R}$ having minimum weight with respect to the arc weights $\sigma_{ij} = 0.5\pi_i + 0.5\pi_j$ (with $\pi_0 = 0$). This is equivalent to the pricing problem encountered when solving LSP by column generation (see Section 6) and can be solved by a straightforward adaptation of the same algorithm. Details on how the SEP is solved by our algorithm and related computational issues are discussed in Section 5.2.

We note that removing constraints (17) from the SEP would ease its solution while still yielding violated CG cuts of rank one that do not necessarily correspond to lifted k-path cuts. We do not pursue this approach because the addition of such CG cuts can significantly complicate the solution of the pricing problem (see Petersen, Pisinger, and Spoorendonk 2008, for a computational investigation of more general CG cuts of rank one).

4.1.2. Weakening of the Lifted *k*-Path Cuts. Our algorithm does not actually use the lifted k-path cuts, but rather their weaker version defined by the inequalities (4). For this reason, it may happen that a set S corresponding to an optimal SEP solution defines a lifted k-path cut violated by $\bar{\mathbf{x}}$, while the corresponding weaker cut (4) is actually not violated by $\bar{\mathbf{x}}$. Nevertheless, in some cases it is still possible to derive a violated inequality (4). An example of such a situation is shown in Figure 1, which depicts the support graph $G(\bar{\mathbf{x}})$ induced by a fractional LSP solution $\bar{\mathbf{x}}$ with three positive variables $\bar{x}_1 = \bar{x}_2 = \bar{x}_3 = 0.5$ corresponding to three routes R_1, R_2, and R_3 visiting the vertex sequences $(0,3,1,2,0)$, $(0,1,2,4,0)$, and $(0,3,4,0)$, respectively. Let $\bar{\xi}_{ij} = \sum_{\ell \in \mathcal{R}} \alpha_{ij\ell} \bar{x}_\ell$, where $\alpha_{ij\ell}$ equals 1 if route R_ℓ visits vertex $j \in N_0$ right after $i \in N_0$ in $\mathcal{G}$, and 0 otherwise. The solid lines in Figure 1 represent arcs (i,j) of $G(\bar{\mathbf{x}})$ with $\bar{\xi}_{ij} = 1$, and the dashed lines represent arcs (i,j) with $\bar{\xi}_{ij} = 0.5$. Suppose that the set $S = \{1,4\}$ has $r(S) = 2$. Note that S is a minimal set (with respect to inclusion) defining a violated lifted k-path cut since only 1.5 routes visit this set. However, the weaker k-path cut (4) defined by the same set S is not violated because $\eta_1(S)\bar{x}_1 + \eta_2(S)\bar{x}_2 + \eta_3(S)\bar{x}_3 = \bar{\xi}_{31} + \bar{\xi}_{01} + \bar{\xi}_{24} + \bar{\xi}_{34} = 2$. Nonetheless, note that for any set $S' \supseteq S$ we have $r(S') \geq r(S)$. Thus, in this example we can obtain a violated k-path cut (4) by selecting $S' = S \cup \{2\}$. In general, the problem of determining a set $S' \supseteq S$ that minimizes (with respect to $\bar{\mathbf{x}}$) the left-hand side of a k-path cut (4) defined by S corresponds to a minimum cut problem. These observations are summarized by the following proposition.

Proposition 2. *Let $S \subseteq N$ be a customer subset with $r(S) = k$ defining a lifted k-path cut (5) that is violated by an LSP solution $\bar{\mathbf{x}}$. Let i_S be the supernode obtained by shrinking the node set S in $G(\bar{\mathbf{x}})$, and define the shrunk*

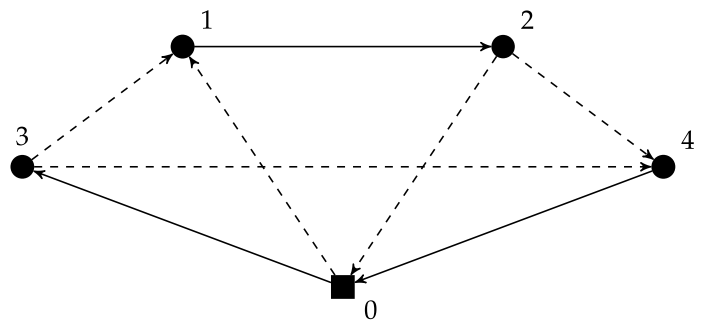

Figure 1. Example of a Violated Lifted k-Path Cut Defined by a Node Set $S=\{1,4\}$ with $r(S)=2$ for Which the Corresponding k-Path Cut (4) Is Not Violated

support graph $G(\bar{\mathbf{x}},S)=(\bar{N},\bar{A})$, *where* $\bar{N}=N_0\backslash S\cup\{i_S\}$ *and* $\bar{A}=\{(i,j)\in A\colon i,j\in N_0\backslash S\}\cup\{(i_S,j)\colon(i,j)\in A, i\in S\}\cup\{(j,i_S)\colon(j,i)\in A, i\in S\}$. *Associate with each arc* $(i,j)\in\bar{A}$ *a capacity* ω_{ij} *defined as* $\omega_{ij}=\bar{\xi}_{ij}$, $\forall\, i,j\in N_0$, *and* $\omega_{i_Sj}=\sum_{i\in S}\bar{\xi}_{ij}$, $\omega_{ji_S}=\sum_{i\in S}\bar{\xi}_{ji}$, $\forall\, j\in N_0$. *Let* $(\bar{S},\bar{N}\backslash\bar{S})$ *be a minimum* $0-i_S$ *cut of* $G(\bar{\mathbf{x}},S)$ *with* $i_S\in\bar{S}$. *A k-path cut* (4) *defined by a set* $S'\supseteq S$ *that has the smallest left-hand side with respect to* $\bar{\mathbf{x}}$ *is obtained by setting* $S'=S\cup\bar{S}\backslash\{i_S\}$.

Proof. See Section EC.2 of the online appendix.

Each time an SEP solution $(\mathbf{y},\boldsymbol{\pi},z)$ provides a violated lifted k-path cut, Proposition 2 is applied by our algorithm to strengthen the corresponding k-path cut (4).

4.2. SR3 Inequalities

SR3 inequalities are a class of valid inequalities for the set packing polytope that correspond to a special case of the subset row inequalities proposed by Jepsen et al. (2008). They are defined by the set $\mathscr{C}=\{\{i,j,k\}\mid i,j,k\in N, i\neq j, k\vee j\neq k\}$ of all customer triplets and are stated as follows:

$$\sum_{\ell\in\mathcal{R}(C)} x_\ell\leq 1,\quad \forall\, C\in\mathscr{C},\tag{20}$$

where for any $C\in\mathscr{C}$ we denote by $\mathcal{R}(C)$ the index subset of all routes visiting at least two of the customers in C. Since there are at most $|N|^3$ inequalities (20), their separation can be done by complete enumeration. In the context of column generation, these cuts are said to be *non robust* meaning that their addition to a model that is solved by column generation may require changing the structure of the pricing subproblems.

4.3. WSR3 Inequalities

WSR3 inequalities are a weakening of the SR3 inequalities proposed by Baldacci, Mingozzi, and Roberti (2011). Unlike SR3 inequalities, WSR3 inequalities are *robust* cuts in the sense that they do not modify the structure of the pricing subproblem when solving LSP by column generation methods. Let $\mathscr{A}_{\{ij\}}\subseteq\mathscr{A}$ be the subset of arcs of $\mathscr{G}$ from i to j and from j to i, $\forall\, i\in N$, $\forall\, j\in N$. WSR3 inequalities are defined by the same set $\mathscr{C}$ of triplets that define the SR3 inequalities and correspond to the inequalities (20) where the set $\mathcal{R}(C)$ is replaced with $\mathcal{R}(E(C))=\{\ell\in\mathcal{R}\colon |A(R_\ell)\cap(\mathscr{A}_{\{ij\}}\cup\mathscr{A}_{\{ik\}}\cup\mathscr{A}_{\{jk\}})|\geq 1\}$.

It is obvious that a WSR3 inequality defined by a triplet C is dominated by the corresponding SR3 inequality. For this reason, we will assume that for any triplet C either the corresponding SR3 inequality or the corresponding WSR3 inequality is added to LSP, but not both. In practice, whenever an SR3 inequality corresponding to a triplet C is added to LSP it may replace the corresponding WSR3 inequality.

5. An Exact Method for Solving SP

In this section, we describe an exact method for solving SP that follows the general schema already adopted by Baldacci, Christofides, and Mingozzi (2008, 2011) for solving the Capacitated VRP. The method executes two phases.

Phase I computes both a lower bound, that we call LB_+, and an upper bound, that we call UB^1, on the optimal SP cost. The upper bound UB^1 is obtained by a metaheuristic algorithm described in Andelmin and Bartolini (2017). The lower bound LB_+ is obtained by solving LSP strengthened by k-path cuts plus WSR3 and SR3 inequalities (called LSP$_+$ for short) via a cut-and-column generation algorithm that we call CCG. However, to speed up the calculation of LB_+, two lower bounding procedures that we call H1 and H2 are executed before CCG. These two procedures compute a weaker lower bound than LB_+, which is obtained by solving heuristically the dual of LSP. Ultimately, the execution of H1 and H2 allows to quickly compute a near-optimal solution $\mathbf{u}^2$ of cost LB_2 to the dual of LSP. This dual solution is used by a dynamic programming algorithm to generate a subset of routes with reduced cost (with respect to $\mathbf{u}^2$) within the gap UB^1-LB_2. These routes form the initial master problem when CCG starts.

Phase II begins by executing a set partitioning heuristic that aims at improving UB^1. This heuristic solves (by an MILP solver) a restricted version of SP defined by a limited subset of routes (which is not guaranteed to contain an optimal solution). This subset contains a set of routes having a reduced cost (with respect to the dual solution of LSP_+ found by CCG) within the gap $UB^1 - LB_+$, which are generated by the dynamic programming algorithm. The set partitioning heuristic terminates with a G-VRP solution of cost UB^2. The best upper bound at this point is $UB^* = \min\{UB^1, UB^2\}$.

After executing the set partitioning heuristic, the algorithm again uses the dynamic programming algorithm to attempt enumerating a subset of routes having reduced cost within the gap $UB^* - LB_+$, which is guaranteed to contain an optimal G-VRP solution. We denote by $\mathcal{R}^*$ the index set of this optimal route subset. If it succeeds, it finally replaces $\mathcal{R}$ with $\mathcal{R}^*$ in SP and solves it to obtain an optimal G-VRP solution.

In the following, we will denote by $\mathbf{u}$ a dual vector associated with the constraints (2) of SP, by $\mathbf{v}$ a dual vector associated with the k-path cuts, by $\mathbf{h}$ a dual vector associated with the WSR3 inequalities, and by $\mathbf{g}$ a dual vector associated with the SR3 inequalities. Both phases of the exact algorithm sketched above rely on a dynamic programming algorithm that takes in input a dual vector $(\mathbf{u}, \mathbf{v}, \mathbf{h}, \mathbf{g})$, a *gap* γ, and a *maximum size* Δ, and outputs a set of at most Δ G-VRP routes having a reduced cost less than γ with respect to $(\mathbf{u}, \mathbf{v}, \mathbf{h}, \mathbf{g})$. This dynamic programming algorithm is detailed in Section 6.3. Overall, the two phases of our exact algorithm execute the following steps.

Phase I of the exact algorithm:

(i) Compute the sets $\mathcal{P}_{ij}$ of nondominated refuel paths for each vertex pair $i, j \in N_0$, $i \neq j$ (see Section EC.1.2 of the online appendix), and construct the multigraph $\mathcal{G}$.

(ii) Use the heuristic of Andelmin and Bartolini (2017) to compute UB^1.

(iii) Execute procedure H1 (Section 5.1) to obtain a dual solution $\mathbf{u}^1$ of cost $LB1$.

(iv) Use the dynamic programming algorithm described in Section 6 with input $(\mathbf{u}^1, \mathbf{0}, \mathbf{0}, \mathbf{0})$, $\gamma = UB^1 - LB1$, and $\Delta = \Delta^1$ to generate a subset $\mathcal{N}^1$ of at most Δ^1 G-VRP routes having a reduced cost with respect to $\mathbf{u}^1$ within the gap $UB^1 - LB1$.

(v) Execute procedure H2 (Section 5.1). The routes in $\mathcal{N}^1$ plus all routes appearing in the Θ best solutions found at step (ii) are added to the master problem at the beginning of H2. H2 terminates with a dual solution $\mathbf{u}^2$ of cost $LB2$.

(vi) Solve LSP_+ by using the procedure CCG (see Section 5.2). CCG returns a solution $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{h}^*, \mathbf{g}^*)$ to the dual of LSP_+ of cost LB_+.

Phase II of the exact algorithm:

(i) *Heuristic route enumeration*. Run the dynamic programming algorithm of Section 6 with input $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{h}^*, \mathbf{g}^*)$, $\gamma = UB^1 - LB_+$, and $\Delta = \Delta^1$, by using the heuristic modifications described in Section EC.4 of the online appendix. We denote by $\mathcal{R}^H$ the index set of the routes obtained in this way.

(ii) Define a reduced problem SPH obtained from SP by replacing $\mathcal{R}$ with $\mathcal{R}^H$ and solve it by an MILP solver. A time limit of τ seconds is imposed in solving SPH. Set $UB^* = \min\{UB^1, UB^2\}$ where UB^2 is the cost of the SPH solution.

(iii) *Exact route enumeration*. The dynamic programming algorithm of Section 6 is run again (now without any heuristic modification) with input $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{h}^*, \mathbf{g}^*)$, $\gamma = UB^* - LB_+$, and $\Delta = \Delta^{\mathrm{MAX}}$. Let $\mathcal{R}^*$ be the index set of all routes it generates.

(iv) If $|\mathcal{R}^*| < \Delta^{\mathrm{MAX}}$, then the reduced problem SP^* obtained from SP by replacing $\mathcal{R}$ with $\mathcal{R}^*$ is guaranteed to contain an optimal SP solution, and it is solved by an MILP solver. Otherwise, the algorithm terminates. Note that in problem SP^* all of the k-path cuts (4) are substituted with their lifted counterpart (5).

The symbols Δ^1, Δ^{MAX}, and Θ in the algorithm above denote a priori defined parameters. In the remainder of this section, we provide a detailed description of the lower bounding procedures H1, H2, and CCG that are used by the algorithm described above.

5.1. Lower Bounding Procedures H1 and H2

Both procedures H1 and H2 are based on the same dual ascent column generation method that computes a near-optimal solution to the dual of LSP, and correspond to an adaptation of a similar procedure proposed by Baldacci, Christofides, and Mingozzi (2008). They only differ in that H1 works with a relaxation of LSP obtained by replacing feasible routes with ng-routes (see Section 6.2), whereas H2 uses elementary routes. The main purpose of H1 and H2 is to provide a near-optimal dual solution of LSP and a corresponding subset of routes that are then used to hot-start a Simplex-based column generation method (called CCG), which finally computes an optimal LSP solution.

In the remainder of this section, we briefly recall the rationale behind the dual ascent method used by H1 and H2, which is based on the following theorem:

Theorem 1. *Let $\lambda_i \in \mathbb{R}$, $\forall\, i \in N$, and $w_i \in \mathbb{R}_+$, $\forall\, i \in N$ be given reals. A feasible solution $\mathbf{u} = (u_1, \ldots, u_n)$ of cost $z_D(\boldsymbol{\lambda}, \mathbf{w})$ to the dual of LSP can be defined as follows:*

$$u_i = w_i \min_{\ell \in \mathcal{R}_i} \left\{ \frac{c_\ell - \lambda(R_\ell)}{w(R_\ell)} \right\} + \lambda_i, \quad \forall\, i \in N, \qquad (21)$$

where $\mathcal{R}_i = \{\ell \in \mathcal{R}\colon \theta_{i\ell} > 0\}$, $w(R_\ell) = \sum_{i \in N} \theta_{i\ell} w_i$, and $\lambda(R_\ell) = \sum_{i \in N} \theta_{i\ell} \lambda_i$.

Proof. See Section EC.2 of the online appendix.

Denoting by $c_\ell(\boldsymbol{\lambda}) = (c_\ell - \lambda(R_\ell))$ and letting ℓ_i be the route giving the minimum in (21) for customer i with respect to $\boldsymbol{\lambda}, \mathbf{w}$, and $\mathcal{R}(\boldsymbol{\lambda}) = \{\ell_i, \forall\, i \in N\}$, Theorem 1 implies that any two real vectors $\boldsymbol{\lambda}$ and $\mathbf{w}$ provide a feasible solution to the dual of LSP through Equations (21) having a cost $z_D(\boldsymbol{\lambda}, \mathbf{w}) = \sum_{i\in N} c_{\ell_i}(\boldsymbol{\lambda}) + \sum_{i\in N} \lambda_i$. Indeed, we have that $\max_{\boldsymbol{\lambda},\mathbf{w}}\{z_D(\boldsymbol{\lambda}, \mathbf{w})\} = \max_{\boldsymbol{\lambda}}\{z_D(\boldsymbol{\lambda}, \mathbf{w}')\} = z_D$, for any fixed $\mathbf{w}' > \mathbf{0}$ (Boschetti, Mingozzi, and Ricciardelli 2008). Hence $\mathbf{w}$ can be (arbitrarily) fixed so that we can write $z_D(\boldsymbol{\lambda})$ in place of $z_D(\boldsymbol{\lambda}, \mathbf{w})$. Moreover, a valid subgradient of $z_D(\boldsymbol{\lambda})$ at point $\boldsymbol{\lambda}$ is given by $\boldsymbol{\varrho} = (\varrho_1, \ldots, \varrho_n)$, where $\varrho_i = \sum_{j\in N} \theta_{i\ell_j}(w_j/w(R_{\ell_j})) - 1$, and z_D equals the optimal cost of the dual of LSP. Clearly, the best dual solution obtained from Theorem 1 is one that maximizes the function $z_D(\boldsymbol{\lambda})$. One can thus heuristically solve $\max_{\boldsymbol{\lambda}\in\mathbb{R}^n}\{z_D(\boldsymbol{\lambda})\}$ by the subgradient method and, given an optimal solution $\boldsymbol{\lambda}^*$, derive an optimal solution of the same cost to the dual of LSP via Equations (21). Since in practice the subgradient method is stopped short of convergence, one obtains a not-necessarily (but almost) optimal dual solution. Although the choice of $\mathbf{w}$ does not affect the value of z_D, the reason for setting $\mathbf{w} \neq \mathbf{1}$ in (21) is that it can speed up the convergence of the subgradient method above. In our algorithm, we set $w_i = c_{0i}, \forall\, i \in N$.

Bounding procedure H1. H1 uses column generation and the dual ascent method above to compute a dual solution $\mathbf{u}^1$ of cost $LB1$ of a relaxation of LSP obtained by replacing the set $\mathcal{R}$ with the set $\mathcal{H}$ of all of the ng-routes (see Section 6.2). It initializes $\boldsymbol{\lambda} = \mathbf{0}$, $\mathbf{u}^1 = \mathbf{0}$, and $LB1 = 0$, and maintains a subset of ng-routes indexed by a subset $\bar{\mathcal{H}} \subseteq \mathcal{H}$, which initially contains all of the single customer routes plus the routes in the best heuristic solution found at step (ii) of phase I. An initial master problem is obtained from LSP by replacing $\mathcal{R}$ with $\bar{\mathcal{H}}$. H1 executes nit_1 macroiterations, each involving two steps:

1. Find a near-optimal dual solution of the master problem: Execute a fixed number nit_2 of subgradient iterations. At each subgradient iteration, use Equations (21) where $\mathcal{R}$ is replaced with $\bar{\mathcal{H}}$ to compute a dual solution of the master problem of cost $\bar{z}_D(\boldsymbol{\lambda})$. Compute a subgradient $\boldsymbol{\varrho}$, and modify the penalties $\boldsymbol{\lambda}$ using the subgradient method. Let $\bar{\boldsymbol{\lambda}}$ be the solution with the largest cost $\bar{z}_D(\bar{\boldsymbol{\lambda}}) = \bar{z}$ found in this way, and let $\bar{\mathbf{u}}$ be the corresponding dual solution given by Theorem 1.

2. Solve the pricing problem: find a subset $\mathcal{N}$ containing at most $2n$ ng-routes having the smallest negative reduced cost with respect to $\bar{\mathbf{u}}$. If $\mathcal{N} = \emptyset$ and $\bar{z} > LB1$, then update $LB1 = \bar{z}$, and $\mathbf{u}^1 = \bar{\mathbf{u}}$. Update $\bar{\mathcal{H}} = \bar{\mathcal{H}} \cup \mathcal{N}$. The pricing problem is solved by dynamic programming as described in Section 6.2.1.

Bounding procedure H2. Procedure H2 corresponds precisely to H1 and executes the same steps 1 and 2 above except that G-VRP routes are used instead of ng-routes (i.e., the set $\mathcal{R}$ is used instead of $\mathcal{H}$). H2 starts with a subset of routes indexed by a set $\bar{\mathcal{R}}$. These routes are generated as described in step (iv) of phase I (see Section 5). In addition, all of the routes appearing in the Θ best solutions found by the heuristic at step (ii) of phase I are included in the initial route set. At step 2, the pricing problem is solved by using the dynamic programming algorithm described in Section 6 with input $(\bar{\mathbf{u}}, \mathbf{0}, \mathbf{0}, \mathbf{0})$, $\gamma = 0$, and $\Delta = 3n$ to generate the set $\mathcal{N}$ of negative reduced cost G-VRP routes that are added to $\bar{\mathcal{R}}$. H2 terminates with a dual solution $\mathbf{u}^2$ of cost $LB2$.

5.2. Cut-and-Column Generation Procedure CCG

The procedure CCG is a standard Simplex-based cut-and-column generation method that solves LSP strengthened by k-path cuts, WSR3 inequalities, and SR3 inequalities. The initial master problem is obtained by replacing $\mathcal{R}$ in LSP with $\bar{\mathcal{R}}$ (i.e., all of the G-VRP routes obtained at the end of H2).

The pricing problem is solved by using the same dynamic programming algorithm of Section 6 with input parameters $\gamma = 0$, and $\Delta = 3n$. LSP is first solved without any additional cuts to obtain an optimal dual solution of LSP. Then, the cutting planes start to be separated. During one iteration of CCG, the master problem is solved, and new cuts are separated and added to LSP. Each time violated cuts are added to LSP, the master problem is reoptimized. Cuts are separated in the following order: WSR3 inequalities, k-path cuts, and SR3 inequalities. However, k-path cuts are only separated if no violated WSR3 inequalities are found, and SR3 inequalities are only separated if either no violated k-path cuts are found, or their addition to LSP increases the objective value less than a threshold ν_{iter}. The pricing problem is solved after each iteration in which no cuts are added to LSP. Additional details on cuts separation are given below.

Separation of the k-path cuts. The 1-fractional lifted k-path cuts (5) are separated by solving (heuristically) the problem SEP described in Section 4.1. SEP is solved by using the MILP solver CPLEX 12.6.2, where the constraints (17) are separated through the CPLEX callback interface by using the dynamic programming algorithm of Section 6 with input parameters $(\mathbf{u}, \mathbf{v}, \mathbf{h}, \mathbf{g}) = (\boldsymbol{\pi}, \mathbf{0}, \mathbf{0}, \mathbf{0})$, $\gamma = -1$, and by setting $c(i,j,p) = 0, \forall\,(i,j,p) \in \mathcal{A}$. For solving SEP, we impose a time limit of $3n$ seconds and store all of the feasible solutions found by CPLEX within this time limit having a cost greater than or equal to ϵ. For each one of these solutions, the corresponding set S is stored and Proposition 2 is used to attempt to find a set $S' \supseteq S$ maximizing the violation of the associated k-path cut (4) which, if violated, is then added to LSP.

Recall that any constraint (17) of SEP is defined by a feasible route. To speed up the solution of SEP, we maintain a pool $\mathcal{R}^0$, which stores the corresponding route whenever a violated constraint (17) is identified. The separation algorithm for the inequalities (17) always checks both $\mathcal{R}^0$ and $\bar{\mathcal{R}}$ for routes defining violated constraints before executing the dynamic programming algorithm of Section 6.

In our preliminary experiments, we have also observed that solving problem SEP may become difficult when the number of customers is large (e.g., when $n > 150$), or when the average "length" (i.e., number of customers) of the routes in the LSP solution is larger than a minimum threshold (e.g., larger than eight). In such cases, our algorithm modifies SEP by imposing that at most ϖ variables π_i can take a positive value, where ϖ is a parameter. This tends to simplify the solution of SEP (because it makes it easier to separate violated constraints (17)), but also implies that only lifted k-path cuts defined by sets S with $|S| \le \varpi$ can be detected. The value of ϖ is determined automatically at the beginning of procedure CCG as described in Section 7.

Separation of the WSR3 and SR3 inequalities. Both WSR3 and SR3 inequalities are separated by complete enumeration just by listing all customer triplets $C \in \mathscr{C}$. However, since the SR3 inequalities can make the pricing subproblem significantly harder, an SR3 inequality is added to LSP only if its violation exceeds a small threshold ε (we use $\varepsilon = 0.01$). Moreover, at most ρ_{sr} SR3 or WSR3 inequalities are added in one iteration.

Computational issues and cutting strategy. Overall, the addition of cutting planes can significantly slow down the convergence of CCG mainly because the addition of the SR3 inequalities makes the pricing problem harder, and the separation of the k-path cuts can be very time consuming. To partially alleviate this drawback, violated inequalities are added in rounds. During one cutting round r, at most $\rho_{\max}$ violated inequalities are added to the master. The separation of all of the inequalities is then stopped, and the round ends as soon as a feasible dual solution is obtained whose cost corresponds to a lower bound lb_r. A round $r > 1$ is called *weak* if $lb_r - lb_{r-1} < \nu_{rnd}$. After the first weak round r, CCG terminates with a dual solution $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{h}^*, \mathbf{g}^*)$ of cost $LB_+ = lb_r$. For instances where $\varpi < n$, the value of ϖ is increased to n after the first round.

The symbols ρ_{sr}, $\rho_{\max}$, and ν_{rnd} in the description above are parameters defined a priori.

6. Solution of the Pricing Problem

Both H2 and CCG in phase I of the exact algorithm, as well as the route enumeration steps in phase II require solving the following *pricing subproblem*: Given a value γ and a dual vector $(\mathbf{u}, \mathbf{v}, \mathbf{h}, \mathbf{g})$, find the largest subset $\mathcal{N}$ of G-VRP routes having a reduced cost with respect to $(\mathbf{u}, \mathbf{v}, \mathbf{h}, \mathbf{g})$ smaller than γ. In this section, we describe a dynamic programming algorithm for solving such problems.

6.1. Reduced Cost of G-VRP Routes and Modified Arc Costs

This section defines the reduced cost of a route R_ℓ (or a path) in $\mathscr{G} = (\mathcal{N}_0, \mathscr{A})$ with respect to a dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{h}}, \bar{\mathbf{g}})$. The reduced cost $\bar{c}_\ell$ of R_ℓ with respect to $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{h}}, \bar{\mathbf{g}})$ is

$$\bar{c}_\ell = c_\ell - \sum_{i \in N} \theta_{i\ell} \bar{u}_i - \sum_{S \in \mathscr{S}} \eta_\ell(S) \bar{v}_S - \sum_{C \in \mathscr{C}:\, \ell \in \mathscr{R}(E(C))} \bar{h}_C - \sum_{C \in \mathscr{C}:\, \ell \in \mathscr{R}(C)} \bar{g}_C. \tag{22}$$

Similarly, a reduced cost can be associated with any path in $\mathscr{G}$ from a vertex $i \in N_0$ to a vertex $j \in N_0$. Let $\mathscr{W}_{ij} \subseteq \mathscr{C}$ be the subset of triplets containing both vertices i and j that define a WSR3 inequality, and let $\mathscr{S}_{ij} \subseteq \mathscr{S}$ be the family of all of the customer subsets containing customer j but not i that define a k-path cut. Define for each arc $(i, j, p) \in \mathscr{A}$ the following *modified arc cost*: $\bar{d}_{ijp} = c_{ijp} - \frac{1}{2}\bar{u}_i - \frac{1}{2}\bar{u}_j - \sum_{S \in \mathscr{S}_{ij}} \bar{v}_S - \sum_{C \in \mathscr{W}_{ij}} \bar{h}_C$ (where $\bar{u}_0 = 0$). Denoting by $\bar{h}_{\{ijl\}}$ the dual variable corresponding to the WSR3 inequality defined by a triplet $C = \{i, j, l\}$, we define the reduced cost of a path $P = (i_0, a_0, i_1, a_1, \ldots, a_{r-1}, i_r)$ visiting the vertex set $V(P) = \{i_0, \ldots, i_r\}$ as

$$\bar{c}(P) = \sum_{k=0}^{r-1} \bar{d}_{i_k i_{k+1} p_k} + \sum_{k=0}^{r-2} \bar{h}_{\{i_k i_{k+1} i_{k+2}\}} - \sum_{C \in \mathscr{C}:\, |V(P) \cap C| \ge 2} \bar{g}_C, \tag{23}$$

where $p_k \in \mathscr{P}_{i_k i_{k+1}}$ is the refueling path corresponding to arc $a_k \in \mathscr{A}$ and we assume $\bar{h}_{\{ijl\}} = 0$ if i, j or l is 0. For a path P corresponding to a route R_ℓ, the reduced cost $\bar{c}(P)$ given by (23) equals $\bar{c}_\ell$.

6.2. G-VRP Paths and Their ng-Path Relaxation

This section introduces some notation needed to define the G-VRP paths used by the dynamic programming algorithm of Section 6.3 to compute G-VRP routes. It also describes an ng-path relaxation (Baldacci, Mingozzi, and Roberti 2011) of the G-VRP paths that is used to speed up their computation. This ng-path relaxation is also used to obtain a relaxation of the G-VRP routes, called ng-routes, which are faster to compute and are used by the bounding procedure H1 in place of the G-VRP routes (see Section 5.1).

6.2.1. G-VRP Forward Paths and G-VRP Routes. A G-VRP forward path, or simply *forward path*, is a simple path $P = (i_0 = 0, a_0, i_1, a_1, \ldots, a_{r-1}, i_r)$ in $\mathscr{G}$ traversing a subset of arcs $A(P) = \{a_0, a_1, \ldots, a_{r-1}\} \subseteq \mathscr{A}$, visiting a subset of customers $V(P)$, and ending at a customer vertex $i_r \in N$. A forward path corresponds to a feasible route that does not necessarily end at the depot, and a forward path ending at the depot is a G-VRP route. Similar to a route, a forward path P is associated with a travel time $t(P)$, a distance $c(P)$, and a driving autonomy $q_k(P)$ at every visited vertex i_k, $k = 1, \ldots, r$, which are defined as for the feasible routes and must satisfy the same feasibility conditions 1–3 (see Section 3). In addition, a forward path P is associated with a reduced cost $\bar{c}(P)$ defined by (23).

6.2.2. Forward ng-Paths and ng-Routes. We define a *forward ng-path* $P = (i_0 = 0, a_0, i_1, a_1, \ldots, a_{r-1}, i_r)$ as a not-necessarily simple path in the multigraph $\mathscr{G}$ starting from the depot 0, traversing a subset $A(P) \subseteq \mathscr{A}$ of arcs, and ending at any vertex $i_r \in N$. For each $i \in N$, let $N_i \subseteq N$ be a (arbitrarily defined) subset of customers such that $i \in N_i$. An ng-path P is associated with a subset $NG(P)$ of the vertices that P has visited, called the ng-set of P, and is defined as $NG(P) = \{i_k \mid i_k \in \bigcap_{l=k+1}^{r} N_{i_l}, k = 1, \ldots, r-1\} \cup \{i_r\}$. An ng-path P ending at i_r must then satisfy $i_r \notin NG(P')$, where P' is the subpath of P from 0 to i_{r-1}. Moreover, P is associated with a travel time $t(P) \leq T$, which is defined in the same way as for a forward path. An *ng-route* is a forward ng-path with $i_r = 0$.

Note that the driving autonomy $q_i(P)$ at every visited vertex i is not defined for ng-paths (and ng-routes); thus, such paths (and routes) do not necessarily satisfy fuel constraints. It would be possible to impose an ng-path that must satisfy the fuel constraints to obtain a stronger ng-path relaxation at the expense of associating additional resources (e.g., a driving autonomy $q_i(P)$) with an ng-path ending in i. However, after some preliminary experiments, we decided not to follow this approach.

Given a dual vector $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{h}}, \bar{\mathbf{g}})$ and the corresponding modified costs $\bar{d}_{ijp}$ (see Section 6.1), we associate a *modified cost* $f(P) \leq \bar{c}(P)$ with each ng-path (or ng-route) P, which is given by expression (23) where $g_C = 0$, $\forall C \in \mathscr{C}$. Let $f(NG, t, j, i)$ be the smallest modified cost of a forward ng-path P ending at i at time $t(P) = t$, visiting j before i, and having an ng-set $NG(P) = NG$. The values $f(NG, t, j, i)$ and the corresponding paths can be computed by means of the following recursion (see Baldacci, Mingozzi, and Roberti 2012 for dominance rules to speed up its computation):

$$f(NG,t,j,i) = \min\{f(NG', t - t(j,i,p), l, j) + \bar{d}_{jip} + \bar{h}_{\{lji\}} \mid NG' \subseteq N_j, j \in NG', NG' \cap N_i = NG \setminus \{i\}, p \in \mathscr{P}_{ji}, l \in N_0 \setminus \{j\}\}. \tag{24}$$

Since the recursion above assumes integer travel times, we actually use it to compute a relaxation of ng-paths and ng-routes, which is obtained by replacing $t(i,j,p)$ with $\lfloor t(i,j,p) \rfloor$, $\forall (i,j,p) \in \mathscr{A}$. The recursion above is used at each iteration by the bounding procedure H1 (see Section 5.1) to compute the set $\mathscr{N}$ containing the ng-routes having the smallest negative reduced cost with respect to the master dual solution $(\bar{\mathbf{u}}, \mathbf{0}, \mathbf{0}, \mathbf{0})$. Note that since in H1 $\bar{\mathbf{h}} = \mathbf{0}$, and $\bar{\mathbf{g}} = \mathbf{0}$ the modified cost of an ng-route equals its reduced cost.

6.2.3. Backward ng-Paths and Completion Bounds. In a similar way as described in Section 6.2.2, one can also define *backward ng-paths*, which correspond to not-necessarily simple paths $P = (i_0, a_0, i_1, a_1, \ldots, a_{r-1}, i_r = 0)$ starting from a customer $i_0 \in N$ and ending at the depot 0. Similar to a forward ng-path, a backward ng-path P is associated with a backward ng-set, which is defined as $NG^b(P) = \{i_k \mid i_k \in \bigcap_{l=0}^{k-1} N_{i_l}, k = 1, \ldots, r-1\} \cup \{i_0\}$, and must satisfy $i_0 \notin NG^b(P')$, where P' is the subpath of P from i_1 to 0. A backward ng-path P is associated with a backward start time $t^b(P) = T - \sum_{k=0}^{r-1} t(i_k, i_{k+1}, a_k)$ and a modified cost $f^b(P)$ computed according to (23) with $g_C = 0$, $\forall C \in \mathscr{C}$.

The smallest modified cost of a backward ng-path P starting from i at time $t^b(P) = t$, visiting j after i, and having backward ng-set $NG^b(P) = NG$ is denoted by $f^b(NG, t, j, i)$. The values $f^b(NG, t, j, i)$ can be computed by dynamic programming in a similar way as the modified costs $f(NG, t, j, i)$ of forward ng-paths. By concatenating a forward path P ending at a customer $i \in N$ at time $t(P)$ with a backward ng-path $\bar{P}$ starting from i, having backward start time $t \geq t(P)$, and backward ng-set NG such that $NG \cap V(P) = \{i\}$, one obtains a relaxation of a G-VRP route. Thus, a lower bound $L(P)$ on the reduced cost of any G-VRP route containing P is

$$L(P) = \bar{c}(P) + \min_{\substack{t \geq t(P),\ j \in N \setminus \{i\} \\ NG:\, NG \cap V(P) = \{i\}}} \{f^b(NG, t, j, i) + \bar{h}_{\{lji\}}\}, \tag{25}$$

where l is the last vertex visited by P before i (we assume $\bar{h}_{\{lij\}} = 0$ if $l = 0$ or $j = 0$, and $L(P) = \bar{c}(P)$ if $i = 0$). The value $L(P)$ is called a *completion bound* of path P.

6.3. A Dynamic Programming Algorithm for Generating G-VRP Routes

The pricing subproblem is solved by a forward dynamic programming algorithm that takes as input the value γ and a dual vector $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{h}}, \bar{\mathbf{g}})$, and outputs a set $\mathscr{N}$ containing the Δ feasible routes having the smallest reduced costs $\bar{c}_\ell < \gamma$ with respect to $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{h}}, \bar{\mathbf{g}})$.

The algorithm is a label setting method analogous to Dijkstra's shortest path algorithm executed on an expanded state-space graph that is dynamically generated, and each vertex, i.e., state, of the state-space graph corresponds to a forward path P. The algorithm uses completion bounds based on the ng-path relaxation described in Section 6.2.3 and standard dynamic programming dominance to fathom suboptimal paths and reduce the size of the state space.

The dynamic programming algorithm maintains two sets $\mathscr{T}$ and $\mathscr{E}$ of *temporary* and *permanent* paths, respectively. It starts with $\mathscr{E} = \varnothing$ and $\mathscr{T} = \{P^0\}$ where P^0 is the path formed by vertex 0 only with $c(P^0) = 0$, $\bar{c}(P^0) = 0$, $t(P^0) = 0$, $q_0(P^0) = Q$, and $L(P^0) = -\infty$. At each iteration, a path P with the smallest bound $L(P)$ is extracted from $\mathscr{T}$. If P ends at the depot 0 (and $P \neq P^0$), it is a G-VRP route having reduced cost $\bar{c}(P) < \gamma$ and is inserted in the set $\mathscr{N}$. Otherwise, it is *expanded* and inserted in $\mathscr{E}$. The expansion of a path P ending at i creates a new path P_{jp}, $\forall j \in N \setminus V(P) \cup \{0\}$, $\forall (i,j,p) \in \mathscr{A}$, ending at j, which is obtained by appending an arc (i,j,p) to the end of P. The travel time of P_{jp} is $t(P_{jp}) = t(P) + t(i,j,p)$ and its cost is $c(P_{jp}) = c(P) + c(i,j,p)$.

Its driving autonomy and reduced cost are instead obtained as follows:

$$q_j(P_{jp}) = \begin{cases} q_i(P) - c(i,j,p) & \text{if } p = 0, \\ Q - c_2(i,j,p) & \text{otherwise,} \end{cases} \tag{26}$$

$$\bar{c}(P_{jp}) = \begin{cases} \bar{c}(P) + \bar{d}_{ijp} + \bar{h}_{\{lij\}} \\ \quad - \sum_{C\in\mathscr{C}:|V(P)\cap C|=2\wedge j\in C} \bar{g}_C & \text{if } i,j \neq 0, \\ \bar{c}(P) + \bar{d}_{ijp} & \text{otherwise,} \end{cases} \tag{27}$$

where l is the customer visited by the path P before i and $\bar{h}_{\{lij\}} = 0$ if $l = 0$. A path P_{jp} is fathomed if one of the following conditions holds:

1. $t(P_{jp}) > T$.
2. $q_j(P_{jp}) < 0$.
3. $q_i(P) < c_1(i,j,p)$ and $p \neq 0$.
4. $L(P_{jp}) \geq \gamma$ (see Section 6.2 for the definition of the completion bound $L(P)$).
5. P_{jp} is *dominated* according to the standard dynamic programming dominance, i.e., there exists another path $P' \in \mathscr{T} \cup \mathscr{E}$ ending at j such that $V(P') = V(P_{jp})$, $c(P') \leq c(P_{jp})$, $q_j(P') \geq q_j(P_{jp})$, and $t(P') \leq t(P_{jp})$.

The algorithm terminates when either $|\mathscr{N}| = \Delta$ or the set $\mathscr{T}$ becomes empty. It terminates prematurely if $|\mathscr{E} \cup \mathscr{T}|$ exceeds a given threshold M.

In Section EC.4 of the online appendix, we describe how to further reduce the number of states generated by the algorithm in the presence of overlapping vertices, i.e., vertices $i, j \in N$ with $c_{ij} = 0$ and $c_{il} = c_{jl}$, $\forall\, l \in N_0 \setminus \{i, j\}$. Moreover, we describe some heuristic modifications to speed up the algorithm by turning it into a heuristic that is used in step (iv) of phase I, and in step (i) of phase II of our exact algorithm (see Section 5).

7. Computational Experiments

In this section, we provide a computational analysis of the exact algorithm described in Section 5. The algorithm was implemented in C and compiled with Visual Studio 2012 64-bit. All computational experiments were run on an Intel Core i7-5600U CPU at 2.66 GHz with 16 GB RAM. CPLEX 12.6.2 was used as the LP solver by the bounding procedure CCG and as the MILP solver for the reduced problems SPH and SP* in phase II of the algorithm, and for the separation problem SEP in procedure CCG.

All of the computing times reported in this paper are in seconds. All of the results have been obtained with the same values of the parameters introduced in Section 5, which were determined by testing different combinations on the instances reported in Table EC.8 of the online appendix. The parameters' values are the following: $M = 60{,}000{,}000$ in the dynamic programming algorithm of Section 6, $\Theta = 0.05L$, where L is the total number of iterations executed by the heuristic at step (ii) of phase I (we used $L = 160$), $\Delta^1 = 50{,}000$ at step (iv) of phase I and step (i) of phase II, $nit_1 = 200$ and $nit_2 = 3n$ in H1 and H2, $\nu_{iter} = 1.0$, $\rho_{sr} = 10$, $\rho_{\max} = 30$, and $\nu_{rnd} = 0.2$ in CCG, $\tau = 600$ at step (ii) of phase II, $\epsilon = 0.1$ in problem SEP, $\Delta^{\text{MAX}} = 1{,}500{,}000$ at step (iii) of phase II. The value of parameter ϖ is determined by CCG by first computing the quantity $nc = n/(\sum_{\ell\in\mathscr{R}} x_\ell)$, where $\mathbf{x}$ is the optimal LSP solution obtained before adding any cut, and then by setting $\varpi = \max\{\lceil n \cdot 0.2 \rceil, 10\}$ if $nc > 8$ or $n > 150$, and $\varpi = n$ otherwise. Finally, we set $|N_i| = 10$, $\forall\, i \in N$ in computing ng-paths and ng-routes, and we define the sets N_i to include the 10 customers j having the smallest value t_{ij}.

7.1. Description of the Test Instances

We have considered two sets of instances. The first set, hereafter called EMH, was proposed by Erdoğan and Miller-Hooks (2012) and comprises 52 instances with 20 to 500 customers and three to 28 refueling stations. We only consider the instances with up to 111 customers. The majority of the instances in this set (40 out of 45 we considered) contain 20 customers and have been randomly generated to simulate different scenarios in terms of customer and refueling station distribution.

The larger EMH instances were created by using a medical textile supply company depot location in Virginia and a customer pool based on hospital locations in Virginia, Maryland, and the District of Columbia. The refueling stations correspond to actual biodiesel stations located in the region. The number of customers in these larger instances ranges from 111 to 500, and we consider the instances with up to 111 customers. Results on the instances with up to 300 customers are provided in Section EC.5 of the online appendix. All these instances assume a maximum driving time T of 11 hours (660 minutes) and the maximum travel distance allowed without refueling is $Q = 300$ miles (the fuel tank capacity is 60 gallons and the fuel consumption rate is $K = 0.2$ gallons per mile). All customers have a service time of 30 minutes, and each refueling stop takes $\delta = 15$ minutes. The vehicles travel at a constant speed of $\text{v} = 40$ miles per hour. In all these instances, the customer and refueling station positions are provided as geographical coordinates (latitude and longitude). All distances are computed by means of the Harvesine formula by using an earth radius of 4,182.45 miles (this is the value used in all previous studies on the G-VRP). In Section EC.3 of the online appendix, we provide the code fragment (in C language) that we used to compute the distances. In our code, we round all of the distances up to the sixth decimal digit, and compute the travel times t_{ij} as described in Section 2 (note that these times are not assumed to be integer).

It should be mentioned that the EMH instances described above may contain some *infeasible* customers. According to Erdoğan and Miller-Hooks, a customer is infeasible if it cannot be visited by a route making at most one refueling stop. In solving these instances, it is assumed that all of the infeasible customers are removed a priori.

Since most of the EMH instances we considered are small, and the five larger instances with 111 customers only differ in the number of refueling stations (ranging from 21 to 28), we have created four additional instances by extracting the first 75 and 100 customers from the two 111-customer instances having 21 and 28 stations.

Moreover, we have created a second set of instances, hereafter called AB, by extracting customers from the larger EMH instances. The set AB contains 40 instances with a number of customers ranging from 50 to 100. We further split this set into two subsets AB1 and AB2. AB1 instances have the same parameter values (T, Q, K, and v) as the original ones of Erdoğan and Miller-Hooks and adopt the same assumptions regarding infeasible customers (i.e., infeasible customers are removed a priori). AB2 instances have the same customers and refueling stations as those in AB1, but the vehicles travel at a higher speed of v = 60 miles per hour. As a rule of thumb, we have considered an increase of ~0.001 liters/kilometer in the fuel consumption for every kilometers/hour increase in the vehicle speed (see Demir, Bektaş, and Laporte 2014), which yields a fuel consumption rate of $\sim K = 0.2137$ gallons per mile. Therefore, we set the maximum distance without refuel equal to $Q = 280$ miles for AB2 instances. Note that the AB2 instances allow longer vehicle routes with respect to EMH and AB1 instances since the higher speed permits to visit more customers within the maximum driving time T.

Finally, unlike in the EMH and AB1 instances, all of the customers are considered feasible in the AB2 instances. Therefore, some AB2 instances contain more feasible customers than the corresponding AB1 ones. AB2 instances also contain customers that cannot be served with a single refueling stop. However, note that this does not necessarily imply that an optimal solution must include routes making more than one refueling stop between two customers (or between the depot and one customer). Indeed, none of the AB2 solutions that we obtained involves more than one refueling stop between two customers.

The EMH instances were kindly provided to us by Erdoğan (2014). The AB instances are available at http://www.vrp-rep.org/variants/item/g-vrp.html.

7.2. Computational Results

This section reports the computational results of our exact algorithm on the two data sets described in the previous section. Additional computational statistics are reported in Section EC.5 of the online appendix to allow for a more detailed analysis of the algorithm.

Tables 1–3 report the results on the EMH and AB instances, respectively. In all these tables, columns "n," "s," and "$|\mathcal{A}|$" report for each instance the number of feasible customers, the number of refueling stations (excluding the one located at the depot), and the number of arcs in $\mathcal{G}$, respectively, while columns "UB^*" and "m" give the best known upper bound and the number of vehicles in the corresponding solution, respectively. Bold entries in column "UB^*" indicate that the instance is solved to optimality by the exact algorithm. Column "LB" reports the initial lower bound corresponding to the optimal cost of LSP before adding any valid inequality, and column "$\%LB$" gives the percentage ratio of LB computed as $100LB/UB^*$. Column "$\%LB_+$" reports the percentage ratio of the final lower bound LB_+ obtained after adding k-path cuts, SR3, and WSR3 inequalities, and column "T_{LB_+}" reports the total CPU time spent by the algorithm for computing LB_+. For the sake of comparison, we also report a column "$\%LB_{SR}$" indicating the percentage ratio of the lower bound LB_{SR} that is obtained by executing procedure CCG (with a time limit of eight hours) when the separation of the k-path cuts is disabled and only the SR3 and WSR3 inequalities are used. Columns "No. of kP" and "No. of SR" report the total number of k-path cuts, and SR3 plus WSR3 inequalities added to LSP, respectively, and column "$|\mathcal{R}^*|$" reports the total number of routes in the final route set $\mathcal{R}^*$ computed at step (iii) of phase II of the exact algorithm. An entry "—" under column "$|\mathcal{R}^*|$" indicates that either the set $\mathcal{R}^*$ was not generated because an integer optimal solution was found by the lower bounding procedure CCG (i.e., $\%LB_+ = 100.00$), or the exact route generation step failed because of insufficient memory. Finally, column "Time" reports the total computing time of the algorithm that also includes the time spent for running the heuristic algorithm at step (ii) of phase I. We impose a time limit for computing the final lower bound LB_+ as follows. After three hours, the separation of all cuts within CCG is stopped and CCG is terminated at the end of the current cutting round. An additional three hours of CPU time are allowed to execute step (iii) of phase II and to solve SP* at step (iv). We also remark that the algorithm is not given any upper bound as input.

The results on the small EMH instances with up to 20 customers are reported in Table EC.3 of the online appendix. The same table also provides a comparison of the results obtained by our algorithm with those obtained by the branch and cut of Koç and Karaoglan (2016). These results show that the lower bound LB can be rather weak on the small EMH instances, and the integrality gap is as high as ~10% (see instance 20c3sC1 in Table EC.3 of the online appendix). Nevertheless, the addition of cutting planes yields an integral LSP solution (i.e., $\%LB_+ = 100$) on all of the small EMH instances.

Table 1 shows that the cutting planes reduce the integrality gap by more than 2% on the larger EMH instances with up to 109 feasible customers. Overall, the algorithm solves all of them in less than 3.5 hours.

Table 1. Results on the Instances of Erdoğan and Miller-Hooks (2012) (Data Set EMH)

Inst.	n	s	$\lvert\mathcal{A}\rvert$	UB^*	m	LB	$\%LB$	$\%LB_{SR}$	$\%LB_+$	T_{LB_+}	No. of kP	No. of SR	$\lvert\mathcal{R}^*\rvert$	Time
75c_21s	75	21	30,522	**3,174.23**	12	3,091.08	97.38	98.04	100.00	5,217	70	147	—	5,219
75c_28s	75	28	36,096	**3,169.29**	12	3,086.14	97.38	98.03	100.00	5,582	84	166	—	5,584
100c_21s	98	21	42,082	**4,431.75**	16	4,378.45	98.80	99.24	100.00	5,206	59	115	—	5,208
100c_28s	98	28	50,282	**4,426.80**	16	4,373.51	98.80	99.24	100.00	6,531	66	143	—	6,534
111c_21s	109	21	57,462	**4,770.47**	17	4,655.37	97.59	98.09	99.96	11,055	116	232	57,359	11,139
111c_22s	109	22	58,480	**4,767.21**	17	4,652.12	97.59	98.09	99.87	10,995	119	247	253,970	11,348
111c_24s	109	24	64,588	**4,767.14**	17	4,652.05	97.59	98.09	99.90	11,176	154	238	55,217	11,274
111c_26s	109	26	66,814	**4,767.14**	17	4,652.05	97.59	98.09	99.85	11,211	118	252	264,414	11,576
111c_28s	109	28	68,878	**4,765.52**	17	4,650.42	97.58	98.09	99.87	11,127	99	225	100,996	11,265
Average							97.81	98.34	99.94	8,678				

Note. Bold entries in column "UB^*" indicate that the corresponding solution found by the exact algorithm is optimal.

For all these instances except 111c_22s the optimal costs match the best upper bounds found by Schneider, Stenger, and Hof (2015).

Tables 2 and 3 report the results on the new instances AB. In particular, Table 2 shows that the algorithm is able to solve all but one AB1 instance with up to 100 customers (only AB110 remains unsolved) and confirms the effectiveness of the cutting planes. On average, the cutting planes reduce the integrality gap by more than 3%, and the k-path cuts contribute significantly.

Table 3 shows that AB2 instances become more difficult for the exact algorithm, and five of them remain unsolved. This is partly because the pricing problem becomes more difficult when longer vehicle routes are allowed (because of the higher vehicle speed). For the same reason, solving the separation problem SEP is also more difficult and on average less violated k-path cuts are identified. Indeed, in some cases the dynamic programming algorithm used to identify violated constraints (17) when solving SEP runs out of memory. Nevertheless, k-path cuts are also very useful on AB2 instances. Indeed, using k-path cuts allows CCG to reduce the integrality gap significantly more than using SR inequalities alone (on average, more than 1.6%).

Overall, Tables 1–3 indicate that the incorporation of the k-path cuts is crucial for our exact algorithm to obtain tight lower bounds and to solve the instances under consideration. As an example, out of the 29 instances reported in Tables 1 and 2 only one (AB101) could be solved by the same algorithm described in this paper with the same parameter and time limit settings if the k-path cuts are ignored.

Table 2. Results on the New Data Set AB1

Inst.	n	s	$\lvert\mathcal{A}\rvert$	UB^*	m	LB	$\%LB$	$\%LB_{SR}$	$\%LB_+$	T_{LB_+}	No. of kP	No. of SR	$\lvert\mathcal{R}^*\rvert$	Time
AB101	50	21	10,590	**2,566.62**	9	2,411.59	93.96	96.97	100.00	1,389	46	74	—	1,391
AB102	50	21	12,768	**2,876.26**	10	2,746.57	95.49	97.46	100.00	2,265	87	135	—	2,266
AB103	50	21	12,604	**2,804.07**	10	2,673.89	95.36	98.88	99.45	1,603	38	84	51,941	1,621
AB104	47	25	7,420	**2,634.17**	9	2,484.15	94.30	97.16	100.00	486	31	47	—	488
AB105	73	21	21,002	**3,939.96**	14	3,761.74	95.48	96.62	99.97	11,028	214	335	33,478	11,082
AB106	74	21	24,956	**3,915.15**	13	3,785.59	96.69	98.44	100.00	3,778	91	97	—	3,781
AB107	75	21	35,694	**3,732.97**	13	3,574.56	95.76	98.44	99.84	10,855	142	296	36,971	10,905
AB108	75	21	31,972	**3,672.40**	13	3,552.68	96.74	98.43	100.00	4,441	101	111	—	4,443
AB109	75	24	29,358	**3,722.17**	13	3,606.29	96.89	98.86	100.00	5,677	82	153	—	5,679
AB110	75	24	29,420	3,612.95	13	3,463.06	95.85	97.96	98.87	8,810	99	303	—	9,806
AB111	71	25	21,462	**3,996.96**	14	3,716.83	92.99	93.44	99.97	8,445	136	240	38,080	8,484
AB112	100	21	52,858	**5,487.87**	18	5,318.29	96.91	98.01	99.93	10,874	228	172	40,151	11,030
AB113	100	21	53,902	**4,804.62**	17	4,675.06	97.30	98.74	99.42	10,779	156	286	706,625	12,276
AB114	100	21	53,686	**5,324.17**	18	5,134.60	96.44	98.31	99.72	11,102	88	303	46,250	11,198
AB115	100	21	50,764	**5,035.35**	17	4,926.93	97.85	99.21	99.86	10,885	121	298	848,786	13,236
AB116	100	21	58,286	**4,511.64**	16	4,430.12	98.19	99.28	99.86	11,021	65	290	133,680	11,371
AB117	99	22	47,174	**5,370.28**	18	5,251.28	97.78	99.02	99.73	10,828	127	248	823,930	12,006
AB118	100	22	48,770	**5,756.88**	19	5,589.62	97.09	98.92	99.48	11,012	163	248	442,413	13,120
AB119	98	25	47,884	**5,599.96**	19	5,424.25	96.86	98.61	99.86	11,146	200	276	41,367	11,226
AB120	96	25	47,658	**5,679.81**	19	5,582.12	98.28	99.10	99.71	11,051	137	250	184,576	11,343
Average							96.31	98.09	99.78	7,874				

Note. Bold entries in column "UB^*" indicate that the corresponding solution found by the exact algorithm is optimal.

Table 3. Results on the New Data Set AB2

Inst.	n	s	$\|\mathcal{A}\|$	UB^*	m	LB	$\%LB$	$\%LB_{SR}$	$\%LB_+$	T_{LB_+}	No. of kP	No. of SR	$\|\mathcal{R}^*\|$	Time
AB201	50	21	19,446	**1,836.25**	6	1,697.91	92.47	97.77	100.00	1,262	38	83	—	1,263
AB202	50	21	19,984	**1,966.82**	6	1,895.35	96.37	100.00	100.00	499	18	54	—	500
AB203	50	21	19,472	**1,921.59**	6	1,881.02	97.89	98.43	100.00	6,147	53	313	—	6,149
AB204	50	25	17,874	**2,001.70**	6	1,875.11	93.68	94.22	100.00	1,772	37	170	—	1,773
AB205	75	21	42,852	**2,793.01**	9	2,678.08	95.89	96.50	99.88	10,745	50	280	36,956	11,056
AB206	75	21	45,558	**2,891.48**	9	2,729.64	94.40	98.49	99.66	11,732	75	260	81,842	12,255
AB207	75	21	54,530	**2,717.34**	8	2,655.32	97.72	100.00	100.00	3,143	30	90	—	3,145
AB208	75	21	49,628	**2,552.18**	8	2,486.74	97.44	99.14	100.00	3,924	42	111	—	3,926
AB209	75	24	51,452	**2,517.69**	8	2,436.94	96.79	98.62	100.00	6,812	68	169	—	6,814
AB210	75	25	53,004	**2,479.97**	8	2,363.20	95.29	96.60	100.00	9,799	62	179	—	9,802
AB211	75	24	47,230	2,977.73[a]	9	2,711.72	91.07	91.80	98.35[b]	7,544	57	153	—	8,358
AB212	100	21	82,270	**3,341.43**	11	3,251.88	97.32	98.80	99.80	11,066	35	208	47,319	11,568
AB213	100	21	90,182	**3,133.24**	10	3,091.73	98.68	99.73	99.97	10,903	32	268	39,914	11,149
AB214	100	21	83,196	3,384.28[a]	11	3,258.59	96.29	98.32	99.41	11,677	33	266	—	12,910
AB215	100	21	83,388	3,480.52[a]	11	3,346.28	96.14	97.34	98.94	11,174	50	239	—	12,029
AB216	100	21	84,630	3,221.78[a]	10	3,150.24	97.78	99.37	99.34	12,327	20	231	—	13,026
AB217	100	22	87,114	**3,714.94**	11	3,660.93	98.55	99.77	99.87	12,280	48	242	45,898	12,759
AB218	100	22	89,486	**3,658.17**	11	3,559.83	97.31	99.03	99.30	11,159	83	246	1,209,047	14,707
AB219	100	25	103,736	3,790.71[a]	11	3,631.40	95.80	97.32	99.12	11,308	48	285	—	12,482
AB220	100	25	88,356	**3,737.88**	11	3,660.60	97.93	99.49	99.87	11,122	44	278	40,473	11,318
Average							96.24	98.04	99.67	8,320				

Note. Bold entries in column "UB^*" indicate that the corresponding solution found by the exact algorithm is optimal.
[a]Best known upper bound found during preliminary experiments.
[b]CCG terminates because of insufficient memory when solving the pricing problem.

8. Conclusions

We have developed an exact algorithm for the green vehicle routing problem (G-VRP) based on a set partitioning formulation strengthened by subset row inequalities and k-path cuts. We have modeled the problem by using a multigraph that does not require to explicitly model the refueling stops, and excludes solutions containing suboptimal refuel paths. We have characterized a subset of the k-path cuts as Chvátal–Gomory cuts of rank one allowing to cast their separation problem as an MILP and embedded the resulting separation algorithm within a lower bounding procedure that constitutes the core of our exact algorithm. Our computational results on benchmark instances introduced by Erdoğan and Miller-Hooks (2012) and new ones show that the exact algorithm provides tight lower bounds and can optimally solve instances with up to ~110 customers.

Acknowledgments

The authors thank Professor Aristide Mingozzi for providing useful comments to an earlier version of the manuscript, and three anonymous referees and the associate editor for several comments and suggestions that improved the paper. This work has been partly funded by the Energy Science Initiative of Aalto University School of Science. This support is gratefully acknowledged.

References

Andelmin J, Bartolini E (2017) A multi-start local search heuristic for the green vehicle routing problem based on a multigraph reformulation. Technical report, Aalto University School of Science, Aalto, Finland.

Avci B, Girotra K, Netessine S (2014) Electric vehicles with a battery switching station: Adoption and environmental impact. *Management Sci.* 61(4):772–794.

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

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

Baldacci R, Mingozzi A, Roberti R (2012) New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS J. Comput.* 24(3):356–371.

Bard JF, Huang L, Dror M, Jaillet P (1998) A branch and cut algorithm for the VRP with satellite facilities. *IIE Trans.* 30(9):821–834.

Bektaş T, Lysgaard J (2015) Optimal vehicle routing with lower and upper bounds on route durations. *Networks* 65(2):166–179.

Boschetti MA, Mingozzi A, Ricciardelli S (2008) A dual ascent procedure for the set partitioning problem. *Discrete Optim.* 5(4): 735–747.

Conrad RG, Figliozzi MA (2011) The recharging vehicle routing problem. Doolen T, Van Aken E, eds. *Proc.* 2011 *Indust. Engrg. Res. Conf.* (IIE, Norcross, GA).

Crevier B, Cordeau J-F, Laporte G (2007) The multi-depot vehicle routing problem with inter-depot routes. *Eur. J. Oper. Res.* 176(2):756–773.

Demir E, Bektaş T, Laporte G (2014) A review of recent research on green road freight transportation. *Eur. J. Oper. Res.* 237(3): 775–793.

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Sci.* 42(3):387–404.

Desaulniers G, Errico F, Irnich S, Schneider M (2016) Exact algorithms for electric vehicle-routing problems with time windows. *Oper. Res.* 64(6):1388–1405

Eisenbrand F (1999) On the membership problem for the elementary closure of a polyhedron. *Combinatorica* 19(2):297–300.

Erdoğan S (2014) Personal communication, April 24.

Erdoğan S, Miller-Hooks E (2012) A green vehicle routing problem. *Transportation Res. Part E* 48(1):100–114.

Felipe Á, Ortuño MT, Righini G, Tirado G (2014) A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Res. Part E* 71:111–128.

Fischetti M, Lodi A (2007) Optimizing over the first Chvátal closure. *Math. Programming* 110(1):3–20.

Goeke D, Schneider M (2015) Routing a mixed fleet of electric and conventional vehicles. *Eur. J. Oper. Res.* 245(1):81–99.

Hiermann G, Puchinger J, Røpke S, Hartl RF (2016) The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *Eur. J. Oper. Res.* 252(3):995–1018.

Ichimori T, Ishii H, Nishida T (1981) Routing a vehicle with the limitation of fuel. *J. Oper. Res. Soc. Japan* 24(3):277–281.

Ichimori T, Ishii H, Nishida T (1983) Two routing problems with the limitation of fuel. *Discrete Appl. Math.* 6(1):85–89.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) Subset-row inequalities applied to the vehicle-routing problem with time windows. *Oper. Res.* 56(2):497–511.

Koç Ç, Karaoglan I (2016) The green vehicle routing problem: A heuristic based exact solution approach. *Appl. Soft Comput.* 39:154–164.

Kohl N, Desrosiers J, Madsen OBG, Solomon M, Soumis F (1999) 2-path cuts for the vehicle routing problem with time windows. *Transportation Sci.* 33(1):101–116.

Laporte G, Nobert Y, Desrochers M (1985) Optimal routing under capacity and distance restrictions. *Oper. Res.* 33(5): 1050–1073.

Lin C, Choy KL, Ho GT, Chung S, Lam H (2014) Survey of green vehicle routing problem: Past and future trends. *Expert Syst. Appl.* 41(4):1118–1138.

Montoya A, Guéret C, Mendoza JE, Villegas JG (2016) A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Res. Part C* 70:113–128.

Pelletier S, Jabali O, Laporte G (2016) 50th anniversary invited article—Goods distribution with electric vehicles: Review and research perspectives. *Transportation Sci.* 50(1):3–22.

Petersen B, Pisinger D, Spoorendonk S (2008) Chvátal–Gomory rank-1 cuts used in a Dantzig–Wolfe decomposition of the vehicle routing problem with time windows. Golden B, Raghavan S, Wasil E, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, New York), 397–419.

Schneider M, Stenger A, Goeke D (2014) The electric vehicle-routing problem with time windows and recharging stations. *Transportation Sci.* 48(4):500–520.

Schneider M, Stenger A, Hof J (2015) An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *OR Spectrum* 37(2):353–387.