OPERATIONS RESEARCH
Vol. 59, No. 5, September–October 2011, pp. 1269–1283
ISSN 0030-364X | EISSN 1526-5463 | 11 | 5905 | 1269

http://dx.doi.org/10.1287/opre.1110.0975

New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem

Roberto Baldacci
Department of Electronics, Computer Science and Systems, University of Bologna, 47521 Cesena, Italy, r.baldacci@unibo.it

Aristide Mingozzi
Department of Mathematics, University of Bologna, 47521 Cesena, Italy, mingozzi@csr.unibo.it

Roberto Roberti
DEIS, University of Bologna, 40136 Bologna, Italy, roberto.roberti6@unibo.it

In this paper, we describe an effective exact method for solving both the capacitated vehicle routing problem (CVRP) and the vehicle routing problem with time windows (VRPTW) that improves the method proposed by Baldacci et al. [Baldacci, R., N. Christofides, A. Mingozzi. 2008. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Math. Programming* **115**(2) 351–385] for the CVRP. The proposed algorithm is based on the set partitioning (SP) formulation of the problem. We introduce a new route relaxation called *ng*-route, used by different dual ascent heuristics to find near-optimal dual solutions of the LP-relaxation of the SP model. We describe a column-and-cut generation algorithm strengthened by valid inequalities that uses a new strategy for solving the pricing problem. The new *ng*-route relaxation and the different dual solutions achieved allow us to generate a reduced SP problem containing all routes of any optimal solution that is finally solved by an integer programming solver. The proposed method solves four of the five open Solomon's VRPTW instances and significantly improves the running times of state-of-the-art algorithms for both VRPTW and CVRP.

Subject classifications: vehicle routing; time windows; dual ascent heuristic; column-and-cut generation.
Area of review: Transportation.
History: Received March 2010; revision received September 2010; accepted November 2010.

1. Introduction

The problem of supplying customers using vehicles based at a central depot is generally known as a vehicle routing problem (VRP). The solution of a VRP calls for the design of a set of routes, each performed by a vehicle starting and ending at the depot, such that all customers are serviced, a set of operational constraints are satisfied, and the total route cost is minimized.

The two most studied members of the VRP family are the capacitated VRP (CVRP) and the VRP with time windows (VRPTW). In the CVRP, a fleet of identical vehicles located at a central depot has to be optimally routed to supply a set of customers with known demands. Each vehicle can perform at most one route, and the total demand delivered by a route cannot exceed the vehicle capacity. The VRPTW generalizes the CVRP imposing that each customer is to be visited within a specified time interval, called time window. As the CVRP is $\mathcal{NP}$-hard, so is the VRPTW.

The most effective exact algorithms for the CVRP are due to Baldacci et al. (2004), Lysgaard et al. (2004), Fukasawa et al. (2006), and Baldacci et al. (2008). Baldacci et al. (2004) described a branch-and-cut (BC) algorithm based on a two-commodity network flow formulation of the CVRP. Lysgaard et al. (2004) proposed a BC algorithm that enhances the method of Augerat et al. (1995). The method of Fukasawa et al. (2006) combines a BC based on the algorithm of Lysgaard et al. (2004) with a new branch-and-cut-and-price (BCP) algorithm based on the two-index and set partitioning (SP) formulations. The lower bound is computed by a column-and-cut generation method that uses s-cycle-free q-routes (with s up to 4) instead of feasible CVRP routes and the valid inequalities used by Lysgaard et al. (2004). The algorithm decides at the root node to use either a pure BC or the BCP algorithm. Baldacci et al. (2008) presented an algorithm based on the SP model strengthened with capacity and clique inequalities and were the first to compute lower bounds based on elementary routes. Their algorithm could not solve three instances solved by Fukasawa et al. (2006). However, it is faster with regard to the problems solved by both methods.

The first exact algorithm for the VRPTW based on the SP formulation was the branch-and-price algorithm of Desrochers et al. (1992). This method was improved by Kohl et al. (1999) by adding 2-path inequalities to LP-relaxation of the SP formulation. Kohl and Madsen (1997) proposed a branch-and-bound algorithm where the lower bounds were computed using subgradient and bundle

methods. These methods were based on 2-cycle elimination algorithms. Irnich and Villeneuve (2006) described a branch-and-price algorithm where the pricing subproblem is solved using a k-cycle elimination procedure. Algorithms based on elementary routes were proposed by Feillet et al. (2004) and Chabrier (2006).

Jepsen et al. (2008) described a BCP based on the SP model and subset-row (SR) inequalities. This method was improved by Desaulniers et al. (2008) by adding both SR and generalized k-path inequalities and using a tabu search heuristic, instead of dynamic programming (DP), to rapidly generate negative reduced cost routes. Their method outperforms all other algorithms, remarkably decreasing the computational time, and solving 5 of the 10 open Solomon instances.

In general, any exact algorithm for the VRPTW based on the SP model can be easily adapted to solve the CVRP by simply relaxing the time window constraints in the route generation phase. Nonetheless, such simple adaptation might not be effective, and none of the methods published so far in the literature for the VRPTW have been proven to effectively solve the CVRP.

The method of Baldacci et al. (2008) for the CVRP is based on an exact solution framework that can be tailored to solve the VRPTW as well as several other VRPs (see Baldacci et al. 2010). The method has the following steps: (i) use different bounding procedures to find near-optimal dual solutions of the LP-relaxation of the SP model strengthened by valid inequalities; (ii) use a column-and-cut generation procedure to reduce the integrality gap by adding in a cutting plane fashion both strengthened capacity and clique constraints; (iii) use the final dual solution achieved at Step (ii) to generate a reduced SP problem containing only the routes of reduced cost less than or equal to the gap between a known upper bound and the lower bound obtained; (iv) solve the resulting reduced problem by an integer programming (IP) solver.

The key components of this method are the bounding procedures of Step (i) that are based on state-space relaxation (see Christofides et al. 1981) to extend the route set with a relaxation of feasible routes easier to compute, and the use of bounding functions to reduce the state space graph computed by DP when solving the pricing problem and generating the final SP model.

Contributions of This Paper. In this paper, we describe an exact method to solve both the VRPTW and the CVRP that improves the method of Baldacci et al. (2008).

We introduce a new route relaxation, called *ng*-route, that improves other nonelementary route relaxations proposed for the CVRP and VRPTW. This relaxation is particularly effective for difficult VRPTW instances with wide time windows and loose capacity constraints. The *ng*-route relaxation is used at Step (i) to derive a new dual ascent heuristic and at Steps (ii) and (iii) to reduce the state-space graph of the DP algorithm in generating elementary routes.

We describe a new family of valid inequalities, called *weak* subset-row inequalities, that are a relaxation of SR inequalities. The main advantage of these new inequalities is that their duals can be implicity considered in solving the pricing problem.

We propose new ideas to improve the pricing algorithm in the column-and-cut procedure based on the use of the dual solution achieved at Step (i) to eliminate routes of negative reduced costs with respect to the current dual solution that cannot be in any optimal solution.

We report computational results for both VRPTW and CVRP showing that the proposed method solves 4 of the 5 open Solomon's VRPTW instances and significantly improves the running times of state-of-the-art algorithms for both VRPTW and CVRP.

In the following, we describe the bounding procedures and the exact method for the VRPTW and their adaptations for the CVRP. This paper is organized as follows. In §2, we describe the SP model for the VRPTW and its relaxations. This section also describes the *ng*-route relaxation. Section 3 describes the exact method. Section 4 presents the bounding procedures and the algorithms for solving the associated pricing subproblems. Section 5 presents the column-and-cut algorithm. Section 6 describes the procedure used to generate the final SP model. Section 7 reports computational results for both VRPTW and CVRP. The concluding remarks are given in §8.

2. Mathematical Formulation and Its Relaxations

The VRPTW is defined on a complete digraph $G = (V', A)$, where $V' = \{0, 1, \ldots, n\}$ is a set of $n+1$ vertices and A is the arc set. Vertex 0 represents the depot, and the vertex subset $V = V' \backslash \{0\}$ corresponds to n customers. With each vertex $i \in V'$ is associated a demand q_i (we assume $q_0 = 0$) and a time window $[e_i, l_i]$, where e_i and l_i represent the earliest and latest time to visit vertex i. With each arc $(i, j) \in A$ is associated a travel cost d_{ij} and a travel time $t_{ij} > 0$, the latter including the service time at vertex i, so the departure time at any customer $i \in V$ coincides with the end of its service. We assume that matrices d_{ij} and t_{ij} satisfy the triangle inequality; therefore, time windows, travel times, and customer demands can be used to properly reduce the set of arcs A (see Desrochers et al. 1992). For each vertex $i \in V'$, we indicate with $\Gamma_i \subseteq V'$ the set of *successors* of i in G and with $\Gamma_i^{-1} \subseteq V'$ the set of *predecessors* of i in G.

A fleet of m identical vehicles of capacity Q stationed at the depot has to fulfill customer demands. A vehicle route $R = (0, i_1, \ldots, i_r, 0)$, with $r \geqslant 1$, is a simple circuit in G passing through the depot, visiting vertices $V(R) = \{0, i_1, \ldots, i_r\}$, $V(R) \subseteq V'$, and such that (i) the total demand of visited customers does not exceed the vehicle capacity Q; (ii) the vehicle leaves the depot 0 at time e_0, visits each customer in $V(R)$ within its time window, and returns to the depot before l_0; (iii) if the vehicle arrives at

$i \in V(R)$ before e_i, the service is delayed to time e_i. The cost of route R is equal to the sum of the travel costs of the arc set, $A(R)$, traversed by route R.

The VRPTW consists of designing at most m routes of minimum total cost such that each customer is visited exactly once by exactly one route.

We use the following additional notation. Given a set $S \subseteq V$, $q(S) = \sum_{i \in S} q_i$ denotes the total demand of customers in S and $k(S)$ the minimum number of vehicles needed to serve all customers in S. $q_{\min} = \min\{\min_{i \in V}\{q_i\}, q(V) - (m-1)Q\}$ is the minimum customer demand of any feasible route. Also, $z(ub)$ denotes an upper bound on the optimal solution cost.

2.1. Set Partitioning Formulation

Let $\mathcal{R}$ be the index set of all feasible routes, and let $a_{i\ell}$, $i \in V'$, $\ell \in \mathcal{R}$, be a (0-1) binary coefficient equal to 1 if $i \in V(R_\ell)$, 0 otherwise (we assume that $a_{0\ell} = 1$, $\forall\, \ell \in \mathcal{R}$). Each route $\ell \in \mathcal{R}$ has an associated cost c_ℓ. Let x_ℓ, $\ell \in \mathcal{R}$, be a (0-1) binary variable equal to 1 if and only if route ℓ is in the optimal solution. The VRPTW formulation based on the SP model, hereafter called F, is

$$(F) \quad z(F) = \min \sum_{\ell \in \mathcal{R}} c_\ell x_\ell, \tag{1}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathcal{R}} a_{i\ell} x_\ell = 1, \quad \forall\, i \in V \tag{2}$$

$$\sum_{\ell \in \mathcal{R}} x_\ell \leqslant m \tag{3}$$

$$x_\ell \in \{0, 1\}, \quad \forall\, \ell \in \mathcal{R}. \tag{4}$$

Constraints (2) specify that each customer $i \in V$ has to be visited by exactly one route. Constraint (3) requires that at most m routes are selected.

2.2. Relaxation *LF*

The LP-relaxation of formulation F can be strengthened with the following valid inequalities.

Capacity Constraints (CCs). Let $\mathcal{S} \subseteq \{S\colon S \subseteq V, |S| \geqslant 2\}$, and let $\rho_\ell(S) = |\{(i,j) \in A(R_\ell)\colon i \in V' \backslash S, j \in S\}|$. The capacity constraints (CCs) are

$$\sum_{\ell \in \mathcal{R}} \rho_\ell(S) x_\ell \geqslant k(S), \quad \forall\, S \in \mathcal{S}. \tag{5}$$

In solving the VRPTW we ignore CCs, while in solving the CVRP we consider only a subset $\mathcal{S}$ of CCs a priori generated, as described in Baldacci et al. (2008).

Strengthened Capacity (SC) **Inequalities**. These inequalities, introduced by Baldacci et al. (2004), lift CCs and are given by inequalities (5), where the route coefficient $\rho_\ell(S)$ is equal to 1 if $V(R_\ell) \cap S \neq \varnothing$, and 0 otherwise. The SCs are used as alternatives to the CCs (see §§4 and 5).

Subset-Row (SR3) **Inequalities**. Let $\mathcal{C} \subseteq \{C \subseteq V\colon |C| = 3\}$ be a subset of all customer triplets, and let $\mathcal{R}(C) \subseteq \mathcal{R}$ be the subset of routes serving at least two customers in C (i.e., $\mathcal{R}(C) = \{\ell \in \mathcal{R}\colon |V(R_\ell) \cap C| \geqslant 2\}$). Subset-row (SR3) inequalities are

$$\sum_{\ell \in \mathcal{R}(C)} x_\ell \leqslant 1, \quad \forall\, C \in \mathcal{C}. \tag{6}$$

SR3s correspond to a subset of SR and clique inequalities used by Jepsen et al. (2008) for the VRPTW and by Baldacci et al. (2008) for the CVRP, respectively. Hereafter, with C (where $C \in \mathcal{C}$) we refer to both the index and the customer triplet of an SR3 inequality.

Weak Subset-Row (WSR3) **Inequalities**. These inequalities are a relaxed form of the SR3s (6) where, given $C \in \mathcal{C}$, the route set $\mathcal{R}(C)$ contains only those routes traversing at least one arc (i,j) with $i, j \in C$. The reason for using WSR3 instead of SR3s is that WSR3 duals can be implicity considered in solving the pricing problem (see §4.5). The SR3s (6), and thus the WSR3s, are separated by complete enumeration.

We denote by LF the LP-relaxation of formulation F strengthened with inequalities (5) and (6) and by $z(LF)$ its optimal solution cost. Moreover, we denote by DF the dual of problem LF. The dual variables are given by the vectors $\mathbf{u} = (u_0, u_1, \ldots, u_n)$, $\mathbf{v} = (v_1, v_2, \ldots, v_{|\mathcal{S}|})$, and $\mathbf{g} = (g_1, g_2, \ldots, g_{|\mathcal{C}|})$, where $u_1, \ldots, u_n$ are associated with constraints (2), $u_0 \leqslant 0$ with constraint (3), $\mathbf{v} \geqslant \mathbf{0}$ with inequalities (5), and $\mathbf{g} \leqslant \mathbf{0}$ with inequalities (6).

By enlarging the route set $\mathcal{R}$ to contain also nonnecessarily elementary routes, it is possible to design efficient dual ascent heuristic procedures to find near optimal solutions of DF. In §4, we describe three bounding procedures, called H^1, H^2 and H^3, where H^k provides lower bound LB_k corresponding to the cost of both a feasible DF solution $(\mathbf{u}^k, \mathbf{v}^k, \mathbf{g}^k)$ and a nonnecessarily feasible LF solution $\mathbf{x}^k$. In §5, we describe a column-and-cut generation procedure, called H^4, for solving LF, that computes lower bound LB_4 corresponding to DF solution $(\mathbf{u}^4, \mathbf{v}^4, \mathbf{g}^4)$. In the following, we denote with c_ℓ^k the reduced cost with respect to $(\mathbf{u}^k, \mathbf{v}^k, \mathbf{g}^k)$ of route $\ell \in \mathcal{R}$.

Procedure H^4 differs from classical column-and-cut generation methods for the new strategy used to solve the pricing problem and the use of the dual solution $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ to reduce the size of the route set $\mathcal{R}$ by removing any route such that $c_\ell^3 > z(ub) - LB_3$. The use of the dual solution $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ has the main benefits of (i) eliminating routes of negative reduced cost when solving the pricing problem in H^4 and (ii) improving the final lower bound $z(LF)$. Procedures H^1, H^2 and H^3 and H^4 are executed in sequence, and the dual solution, $(\mathbf{u}^k, \mathbf{v}^k, \mathbf{g}^k)$, of H^k is used to hot-start procedure H^{k+1}, $k = 1, 2, 3$.

2.3. *ng*-Route Relaxation

In this section, we describe a new relaxation, called *ng*-route, that is used in solving the pricing subproblems in the bounding procedures H^2, H^3, and H^4.

2.3.1. *ng*-Route Relaxations for the VRPTW. A *forward path* $P = (0, i_1, \ldots, i_{k-1}, i_k)$ is an elementary path starting from depot 0 at time e_0, visiting vertices $V(P) = \{0, i_1, \ldots, i_{k-1}, i_k\}$ within their time windows, and ending at customer $i_k = \sigma(P)$ at time $t(P)$ with $e_{\sigma(P)} \leqslant t(P) \leqslant l_{\sigma(P)}$. We denote by $A(P)$ the set of arcs traversed by path P and by $c(P) = \sum_{(i,j)\in A(P)} d_{ij}$ the cost of path P.

A well-known relaxation of forward paths is the (t, i)-relaxation. A (t, i)-path is a nonnecessarily elementary path starting from the depot at time e_0, visiting a set of customers (even more than once) within their time windows, and ending at vertex i at time $e_i \leqslant t \leqslant l_i$. In the (t, i)-relaxation the vehicle capacity constraint is ignored. The cost $f(t, i)$ of the least cost (t, i)-path with 2-cycle elimination can be computed using DP (see Christofides et al. 1981). It can be shown that $f(t, i)$ is a valid lower bound on the cost $c(P)$ of any forward path P, such that $t(P) = t$ and $\sigma(P) = i$.

A (t, i)-route is $(t, 0)$-path, visiting at time t the last customer i before arriving at depot 0. The cost of the (t, i)-route of minimum cost is given by $f(t, i) + d_{i0}$.

The new *ng*-route relaxation can be described as follows. Let $N_i \subseteq V$ be a set of selected customers for vertex i (according to some criterion), such that $N_i \ni i$ and $|N_i| \leqslant \Delta(N_i)$, where $\Delta(N_i)$ is a parameter (e.g., $\Delta(N_i) = 5$, $\forall\, i \in V$, and N_i contains i and the four nearest customers to i). The sets N_i allow us to associate with each forward path $P = (0, i_1, \ldots, i_k)$ the subset $\Pi(P) \subseteq V(P)$ containing customer i_k and every customer i_r, $r = 1, \ldots, k-1$ of P that belongs to all sets $N_{i_{r+1}}, \ldots, N_{i_k}$ associated with the customers $i_{r+1}, \ldots, i_k$ visited after i_r. The set $\Pi(P)$ is defined as

$$\Pi(P) = \left\{ i_r \colon i_r \in \bigcap_{s=r+1}^{k} N_{i_s},\ r = 1, \ldots, k-1 \right\} \bigcup \{i_k\}. \tag{7}$$

For example, let $P = (0, 1, 2, 3, 4, 5)$ be a path ending at vertex 5, and let $N_1 = \{1, 10, 11\}$, $N_2 = \{2, 10, 11\}$, $N_3 = \{1, 2, 3\}$, $N_4 = \{2, 3, 4\}$, and $N_5 = \{2, 3, 5\}$. Then $1 \notin N_2 \cap N_3 \cap N_4 \cap N_5$, $2 \in N_3 \cap N_4 \cap N_5$, $3 \in N_4 \cap N_5$, and $4 \notin N_5$. Thus, from expression (7) we have $\Pi(P) = \{2, 3, 5\}$.

A *forward ng-path* (NG, t, i) is a nonnecessarily elementary path $P = (0, i_1, \ldots, i_{k-1}, i_k = i)$ starting from the depot at time e_0, visiting a subset of customers (even more than once) within their time windows such that $NG = \Pi(P)$, ending at customer i at time $e_i \leqslant t \leqslant l_i$, and such that $i \notin \Pi(P')$, where $P' = (0, i_1, \ldots, i_{k-1})$. We denote by $f(NG, t, i)$ the cost of the least cost forward *ng*-path (NG, t, i).

An (NG, t, i)-route is an $(NG, t, 0)$-path visiting at time t the last customer i before arriving at the depot. The cost of the least cost (NG, t, i)-route is given by $f(NG, t, i) + d_{i0}$.

Functions $f(NG, t, i)$ can be computed using DP as follows. Let $\Omega(t, j, i)$ be the subset of departure times from customer j to arrive at customer i at time t when j is visited immediately before i, that is, (i) $\Omega(t, j, i) = \{t' \colon e_j \leqslant t' \leqslant \min\{l_j, t - t_{ji}\}\}$ if $t = e_i$, and (ii) $\Omega(t, j, i) = \{t - t_{ji} \colon e_j \leqslant t - t_{ji} \leqslant l_j\}$ if $e_i < t \leqslant l_i$. The state-space graph $\mathcal{H} = (\mathcal{C}, \Psi)$ is defined as follows:

$$\mathcal{C} = \{(NG, t, i) \colon \forall\, NG \subseteq N_i \text{ s.t. } NG \ni i,\ \forall\, t, e_i \leqslant t \leqslant l_i,\ \forall\, i \in V'\}, \tag{8}$$

$$\Psi = \{((NG', t', j), (NG, t, i)) \colon \forall\, (NG', t', j) \in \Psi^{-1}(NG, t, i),\ \forall\, (NG, t, i) \in \mathcal{C}\}, \tag{9}$$

where $\Psi^{-1}(NG, t, i) = \{(NG', t', j) \colon \forall\, NG' \subseteq N_j$, s.t. $NG' \ni j$ and $NG' \cap N_i = NG \backslash \{i\}$, $\forall\, t' \in \Omega(t, j, i)$, $\forall\, j \in \Gamma_i^{-1}\}$. Notice that the condition $NG' \cap N_i = NG \backslash \{i\}$ imposes that a state (NG, t, i) can be reached only from a state (NG', t', j) such that $i \notin NG'$.

The dynamic programming recursion for computing $f(NG, t, i)$ is as follows:

$$f(NG, t, i) = \min_{(NG', t', j) \in \Psi^{-1}(NG, t, i)} \{f(NG', t', j) + d_{ji}\}, \quad \forall\, (NG, t, i) \in \mathcal{C}. \tag{10}$$

The following initialization is required: $f(\{0\}, e_0, 0) = 0$ and $f(\{0\}, t, 0) = \infty$, $\forall\, t$ such that $e_0 < t \leqslant l_0$.

The cost $c(P)$ of any elementary forward path P satisfies the following inequality:

$$c(P) \geqslant \min_{NG \subseteq V(P) \cap N_{\sigma(P)}} \{f(NG, t(P), \sigma(P))\}. \tag{11}$$

Notice that recursion (10) allows 2-vertex loops, which can be eliminated by using the method of Christofides et al. (1981). Nonetheless, we do not implement this method because of the additional memory required and because, in practice, whenever sets N_i contain the 8–10 nearest customers to i, rarely the resulting (NG, t, i)-paths contain 2-vertex loops.

The quality of functions $f(NG, t, i)$ strongly depend on the sets N_i, $i \in V$. When N_i contains all customers (i.e., $N_i = V$, $\forall\, i \in V$), then it is quite obvious that $f(NG, t, i)$ provide elementary least cost paths. In the computational tests (see §7), we show that a good trade-off between the quality of functions $f(NG, t, i)$ achieved and the computing time spent to compute (10) is to define each set N_i, $\forall\, i \in V$, to contain the k-nearest customers to i, for a limited value of k ($k = 8, 10$).

Notice that the decremental state-space (see Righini and Salani 2008) and the partial elementarity (see Desaulniers et al. 2008) relaxations are special cases of the relaxed state-space $\mathcal{H}$. These relaxations correspond to the relaxed state-space $\mathcal{H}$ obtained by setting $N_i = \hat{V}$, $i \in V$, where $\hat{V} \subseteq V$ is a selected subset of customers. The resulting DP algorithm (10) prevents multiple visits to the customers in $\hat{V}$ in the paths associated with $f(NG, t, i)$, allowing multiple visits to the others.

2.3.2. Reverse Functions $f^{-1}(t,i)$ and $f^{-1}(NG,t,i)$. We define a *backward path* $\bar{P}=(\sigma(\bar{P})=i_k,i_{k+1},\dots,i_h,0)$ as a path starting from vertex $\sigma(\bar{P})$ at time $t(\bar{P})$, visiting customers in $V(\bar{P})=\{i_k,i_{k+1},\dots,i_h,0\}$ within their time windows, and ending at the depot before time l_0. Lower bounds on the cost $c(\bar{P})$ of $\bar{P}$ can be derived using the (t,i)-path and the *ng*-path as follows.

A *backward* (t,i)-path is a nonnecessarily elementary path starting from i at time t, visiting a set of customers (even more than once) within its time windows, and ending at the depot before l_0. Denote by $f^{-1}(t,i)$ the cost of the least cost backward (t,i)-path without 2-vertex loops.

With a *backward* $\bar{P}=(i_k,i_{k+1},\dots,i_h,0)$, we associate the subset $\Pi^{-1}(\bar{P})\subseteq V(\bar{P})$ containing customer i_k and every customer i_r, $r=k+1,\dots,h$, of $\bar{P}$ that belongs to all sets $N_{i_k},N_{i_{k+1}},\dots,N_{i_{r-1}}$. The set $\Pi^{-1}(\bar{P})$ is defined as:

$$\Pi^{-1}(\bar{P})=\{i_k\}\cup\left\{i_r\colon i_r\in\bigcap_{s=k}^{r-1}N_{i_s},\ r=k+1,\dots,h\right\}. \quad (12)$$

For example, let $\bar{P}=(5,6,7,8,9,0)$ be a path starting from vertex 5, and let $N_5=\{5,7,8\}$, $N_6=\{6,7,8\}$, $N_7=\{7,8,10\}$, $N_8=\{8,9,10\}$ and $N_9=\{8,9,10\}$. Then, $6\notin N_5$, $7\in N_5\cap N_6$, $8\in N_5\cap N_6\cap N_7$, and $9\notin N_5\cap N_6\cap N_7\cap N_8$. Thus, from expression (12) we have $\Pi(\bar{P})=\{5,7,8\}$.

A *backward ng*-path (NG,t,i) is a nonnecessarily elementary path $\bar{P}=\{i_k,i_{k+1},\dots,i_h,0\}$ starting from i at time $e_i\leqslant t\leqslant l_i$, visiting a subset of customers (even more than once) within their time windows such that $NG=\Pi^{-1}(\bar{P})$, ending at the depot before l_0, and such that $i\notin\Pi^{-1}(P')$, where $P'=(i_{k+1},\dots,i_h,0)$. Let $f^{-1}(NG,t,i)$ be the cost of the least-cost backward *ng*-path (NG,t,i).

Functions $f^{-1}(t,i)$ and $f^{-1}(NG,t,i)$ can be computed with the same DP recursions used to compute $f(t,i)$ and $f(NG,t,i)$ on the VRPTW instance resulting from the following operations: (i) for each vertex $i\in V'$, replace time window $[e_i,l_i]$ with $[e_i',l_i']$, where $e_i'=l_0-l_i$ and $l_i'=l_0-e_i$; (ii) replace the cost and time matrices $[d_{ij}]$ and $[t_{ij}]$ with their transposed matrices $[d_{ij}]^T$ and $[t_{ij}]^T$. It is easy to see that $c(\bar{P})$ and $f^{-1}(NG,t,i)$ satisfy an inequality similar to (11).

2.3.3. *ng*-Route Relaxations for the CVRP. To solve the CVRP, we use the (q,i)-path and *ng*-path relaxations defined as follows.

A (q,i)-path is a nonnecessarily elementary path starting from the depot, visiting a set of customers of total demand equal to q, and ending at vertex i. The cost $f(q,i)$ of the least cost (q,i)-path can be computed as described by Christofides et al. (1981). A (q,i)-route is a $(q,0)$-path where i is the last customer visited before arriving at the depot.

A *forward ng*-path (NG,q,i) is a nonnecessarily elementary path $P=(0,i_1,\dots,i_{k-1},i_k=i)$ starting from the depot, visiting a subset of customers of total demand equal to q such that $NG=\Pi(P)$, ending at customer i, and such that $i\notin\Pi(P')$, where $P'=(0,i_1,\dots,i_{k-1})$. We denote by $f(NG,q,i)$ the cost of the least cost forward *ng*-path (NG,q,i). An (NG,q,i)-route is an $(NG,q,0)$-path where i is the last customer visited before arriving at the depot. Functions $f(NG,q,i)$ can be computed using DP recursions similar to (10) on graph $\mathcal{H}=(\mathcal{E},\Psi)$ defined as

$$\mathcal{E}=\left\{(NG,q,i)\colon q_i\leqslant q\leqslant Q,\ \forall NG\subseteq N_i \text{ s.t. } NG\ni i \text{ and } \sum_{j\in NG}q_j\leqslant q,\ \forall i\in V'\right\}, \quad (13)$$

$$\Psi=\{((NG',q',j),(NG,q,i))\colon \forall(NG',q',j)\in\Psi^{-1}(NG,q,i),\ \forall(NG,q,i)\in\mathcal{E}\}, \quad (14)$$

where $\Psi^{-1}(NG,q,i)=\{(NG',q-q_i,j)\colon \forall NG'\subseteq N_j \text{ s.t. } NG'\ni j \text{ and } NG'\cap N_i=NG\backslash\{i\},\ \forall j\in\Gamma_i^{-1}\}$.

2.3.4. Reverse Functions $f^{-1}(q,i)$ and $f^{-1}(NG,q,i)$. By using the transpose of the cost matrix $[d_{ij}]$, we can compute the reverse functions $f^{-1}(q,i)$ and $f^{-1}(NG,q,i)$ with recursions similar to those used for $f(q,i)$ and $f(NG,q,i)$. Obviously, for symmetric CVRPs we have $f^{-1}(q,i)=f(q,i)$ and $f^{-1}(NG,q,i)=f(NG,q,i)$.

3. An Exact Algorithm

In this section, we describe an exact algorithm for solving both the VRPTW and the CVRP. The algorithm generates a reduced problem $\hat{F}$ obtained from F by replacing the route set $\mathcal{R}$ with $\hat{\mathcal{R}}\subseteq\mathcal{R}$ containing any optimal solution and solves $\hat{F}$ with an IP solver. The core of the algorithm is the bounding procedures H^1, H^2, H^3, and H^4 introduced in §2.2 and described in §§4.2, 4.3, 4.4, and 5. The algorithm can be described as follows.

1. Execute in sequence H^1, H^2, and H^3. If the solution $\mathbf{x}^k$ corresponding to LB_k is a feasible F solution, for some $k\in\{1,2,3\}$, $\mathbf{x}^k$ is an optimal solution, stop.

2. Call procedure GENR (see §4.5) to generate the largest route set $\mathcal{R}^3\subseteq\mathcal{R}$ such that: (*a*) $c_\ell^3\leqslant z(ub)-LB_3$, $\forall\,\ell\in\mathcal{R}^3$, and (*b*) $|\mathcal{R}^3|\leqslant\Delta(\mathcal{R})$, where $\Delta(\mathcal{R})$ is a parameter. If $|\mathcal{R}^3|<\Delta(\mathcal{R})$, $\mathcal{R}^3$ contains the routes of any optimal solution of cost less than or equal to $z(ub)$ and is defined *optimal*. Otherwise, $\mathcal{R}^3$ is defined *not-optimal*.

3. Call procedure H^4 to compute LB_4 corresponding to the cost of the DFsolution$(\mathbf{u}^4,\mathbf{v}^4,\mathbf{g}^4)$. If the optimal primal solution $\mathbf{x}^4$ of LF is integer, stop.

4. We have two cases:

(i) $\mathcal{R}^3$ is optimal. Select the subset $\hat{\mathcal{R}}\subseteq\mathcal{R}^3$ of routes such that $c_\ell^4\leqslant z(ub)-LB_4$.

(ii) $\mathcal{R}^3$ is not-optimal. Call GENRF (see §6) to compute the set of routes $\hat{\mathcal{R}}$ such that:

$$\text{(a) } c_\ell^3\leqslant z(ub)-LB_3,\ \forall\,\ell\in\hat{\mathcal{R}},$$

$$\text{(b) } c_\ell^4\leqslant z(ub)-LB_4,\ \forall\,\ell\in\hat{\mathcal{R}},\quad \text{(c) } |\hat{\mathcal{R}}|\leqslant\Delta(\mathcal{R}). \quad (15)$$

If $|\hat{\mathcal{R}}|<\Delta(\mathcal{R})$, then $\hat{\mathcal{R}}$ contains the routes of any optimal solution and is defined *optimal*.

5. Solve problem $\hat{F}$ derived from F by replacing $\mathcal{R}$ with $\hat{\mathcal{R}}$ and adding the subsets of SCs and SR3s saturated by the optimal LF solution produced by H^4. Let $z(\hat{F})$ be the cost of the optimal solution $\hat{\mathbf{x}}$ of $\hat{F}$ (we assume $z(\hat{F}) = \infty$ if no feasible solution exists). If $\hat{\mathcal{R}}$ is optimal, $\hat{\mathbf{x}}$ is an optimal solution; otherwise, $\min\{z(\hat{F}), z(ub)\}$ is a valid upper bound to $z(F)$.

4. Bounding Procedures H^1, H^2, and H^3

Bounding procedures H^1, H^2, and H^3 use the same column-and-cut generation method, called CCG, to find lower bounds LB_1, LB_2 and LB_3 corresponding to the cost of three different near-optimal DF solutions. Procedures H^1 and H^2 are based on two different relaxations and add CCs. Procedure H^3 is based on elementary routes and adds SCs and WSR3s.

4.1. Algorithm *CCG*

CCG differs from standard column-and-cut generation methods based on the simplex algorithm as it uses a dual ascent heuristic to solve the master problem. CCG was proposed by Baldacci et al. (2008) for the CVRP and used by Boschetti et al. (2008) for the SP problem, who showed that CCG is faster than the simplex as it is not affected by the typical degeneracy of the simplex. A step-by-step description of CCG is provided in the e-companion paper.

Define $A(C) = \{(i, j) \in A:\ i, j \in C\}$, and let $\mathcal{C}_\ell = \{C \in \mathcal{C}:\ |A(C) \cap A(R_\ell)| \geqslant 1\}$ be the subset of WSR3s involving the route ℓ. Algorithm CCG is based on the following theorem.

THEOREM 1. *Let us associate penalties* $\lambda_i \in \mathbb{R}$, $\forall\, i \in V$, *with constraints* (2), $\lambda_0 \leqslant 0$ *with constraint* (3), $\mu_S \geqslant 0$, $\forall\, S \in \mathcal{S}$, *with constraints* (5) *in the form of either* SC *or* CC, *and* $\omega_C \leqslant 0$, $\forall\, C \in \mathcal{C}$, *with constraints* (6) *in the relaxed form of* WSR3. *Let* $\mathcal{R}$ *be the index set of non-necessarily elementary routes, and let* $\mathcal{R}_i \subseteq \mathcal{R}$, $i \in V$, *be the index subset of the routes visiting customer* i. *For each* $i \in V$, *compute*

$$b_i = q_i \min_{\ell \in \mathcal{R}_i} \left\{ \frac{c_\ell - \sum_{i \in V'} a_{i\ell}\lambda_i - \sum_{S \in \mathcal{S}} \rho_\ell(S)\mu_S - \sum_{C \in \mathcal{C}_\ell} \omega_C}{\sum_{i \in V} a_{i\ell} q_i} \right\}. \tag{16}$$

A feasible DF solution $(\mathbf{u}, \mathbf{v}, \mathbf{g})$ *of cost* $z(DF(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}))$ *is given by the following expressions*:

$$u_i = b_i + \lambda_i,\ \forall\, i \in V, \quad u_0 = \lambda_0, \quad v_s = \mu_S,\ \forall\, S \in \mathcal{S},$$
$$\text{and} g_C = \omega_C,\ \forall\, C \in \mathcal{C}. \tag{17}$$

PROOF. The proof is provided in the e-companion to this paper. □

Algorithm CCG is a column-and-cut generation-like method for solving the problem

$$LCG = \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}} \{z(DF(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}))\}. \tag{18}$$

CCG executes a number of macro-iterations to compute a dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ of the master problem, defined by the route subset $\bar{\mathcal{R}} \subseteq \mathcal{R}$, solving problem (18) with a predefined number $Maxit2$ of subgradient iterations to modify the penalty vectors $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$.

Baldacci et al. (2008) have shown that a valid subgradient of function $z(DF(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}))$ at point $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega})$ can be computed by associating with the current DF solution a nonnecessarily feasible LF solution $\mathbf{x}$ of cost $z(LF) = z(DF(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}))$ defined as follows. Let $\tilde{\mathcal{R}}$ be the index set of the distinct routes producing b_i, $i \in V$, in expressions (16), and let $\ell(i)$ be the index of the route in $\tilde{\mathcal{R}}$ associated with b_i, $i \in V$. Solution $\mathbf{x}$ is computed as $x_\ell = \sum_{i \in V} a_{i\ell}(q_i / (\sum_{i \in V} a_{i\ell} q_i))\xi^i_\ell$, $\ell \in \tilde{\mathcal{R}}$, by setting $\xi^i_{\ell(i)} = 1$ and $\xi^i_\ell = 0$, $\forall\, \ell \in \tilde{\mathcal{R}} \backslash \{\ell(i)\}$, $\forall\, i \in V$. The values of the left-hand side of constraints (2), (3), (5), and (6) with respect to $\mathbf{x}$ are used to update penalty vectors $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{\omega}$ by means of the usual sugradient expressions, where in computing the step size we do not assume to know an upper bound $z(ub)$ but use $z(ub) = 1.2z(LF)$.

The set $\mathcal{S}$ of CC (or SC) inequalities is generated a priori (see §2.2). After computing the master dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$, CCG adds to the master as subset of WSR3 inequalities violated by the LF solution $\mathbf{x}$. Moreover, CCG generates a subset $\mathcal{N}$ of routes having negative reduced cost with respect to $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$. If $\mathcal{N} \neq \varnothing$, then CCG sets $\bar{\mathcal{R}} = \bar{\mathcal{R}} \cup \mathcal{N}$; otherwise, $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ is a feasible DF solution. CCG terminates after $Maxit1$ macro iterations ($Maxit1$ defined a priori).

We denote by $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{g}^*)$ and $\mathbf{x}^*$ the final DF and LF solutions of cost LCG achieved by CCG using penalty vectors $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*)$, respectively. A step-by-step description of CCG is provided in the e-companion to this paper.

4.2. Bounding Procedure H^1

Procedure H^1 enlarges the route set $\mathcal{R}$ with the t-routes (to solve the VRPTW) or the q-routes (in the CVRP, see Christofides et al. 1981). The initial route set $\bar{\mathcal{R}}$ of the master problem contains all single customer routes $(0, i, 0)$, $i \in V$. WSR3s are ignored (i.e., we set $\mathcal{C} = \varnothing$); solving the VRPTW, we initialize $\mathcal{S} = \varnothing$, whereas in the CVRP the set $\mathcal{S}$ contains CCs and is generated a priori as described in Baldacci et al. (2008). We initialize $\boldsymbol{\lambda} = \mathbf{0}$, $\boldsymbol{\mu} = \mathbf{0}$, and $\boldsymbol{\omega} = \mathbf{0}$.

Define the modified arc cost $\bar{d}_{ij} = d_{ij} - (1/2)(\bar{u}_i + \bar{u}_j) - \sum_{S \in \mathcal{S}_{ij}} \bar{v}_S$, $\forall\, (i, j) \in A$, with respect to the current dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$, where $\mathcal{S}_{ij} = \{S \in \mathcal{S}:\ (i, j) \in A,\ i \in V' \backslash S,\ j \in S\}$. The set $\mathcal{N}$ is computed as follows. If we use t-routes, we compute functions $f(t, i)$ using the modified arc costs $\bar{d}_{ij}$ instead of d_{ij}. Let $h(i) = \min_{e_i \leqslant t \leqslant l_i} \{f(t, i) + \bar{d}_{i0}\}$. The set $\mathcal{N}$ contains any t-route corresponding to $h(i) < 0$, $i \in V$. Similarly, we compute $\mathcal{N}$ when $\bar{\mathcal{R}}$ contains q-routes.

At the end, H^1 sets $(\mathbf{u}^1, \mathbf{v}^1, \mathbf{g}^1) = (\mathbf{u}^*, \mathbf{v}^*, \mathbf{g}^*)$, $\mathbf{x}^1 = \mathbf{x}^*$, $(\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1, \boldsymbol{\omega}^1) = (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*)$, $LB_1 = LCG$.

4.3. Bounding Procedure H^2

Procedure H^2 enlarges the route set $\mathcal{R}$ with the ng-routes and adds to LF the same set $\mathcal{S}$ of CCs used by H^1. WSR3s (6) are ignored. We initialize $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\boldsymbol{\lambda}^1, \boldsymbol{\mu}^1, \boldsymbol{\omega}^1)$, define $d^1_{ij} = d_{ij} - (1/2)(u^1_i + u^1_j) - \sum_{S \in \mathcal{S}_{ij}} v^1_S$, $\forall (i,j) \in A$, and compute N_i to be the $\Delta(N_i)$ nearest customers to i according to d^1_{ij}. We compute functions $f(NG, t, i)$ using d^1_{ij} instead of d_{ij} in recursions (10) and the costs $h(i) = \min_{(NG,t,i) \in \mathcal{E}}\{f(NG,t,i) + d^1_{i0}\}$, $i \in V$, of the least-cost ng-route visiting i immediately before arriving at the depot. The route set $\bar{\mathcal{R}}$ contains the ng-routes corresponding to $h(i) < 0$, $i \in V$.

At each iteration, to generate the set $\mathcal{N}$, we compute functions $f(NG, t, i)$ with the modified arc cost $\bar{d}_{ij}$, $\forall (i,j) \in A$, and N_i contains the $\Delta(N_i)$ nearest customers to i according to $\bar{d}_{ij}$. $\mathcal{N}$ contains every ng-route corresponding to $h(i) < 0$, $i \in V$.

At the end, H^2 sets $(\mathbf{u}^2, \mathbf{v}^2, \mathbf{g}^2) = (\mathbf{u}^*, \mathbf{v}^*, \mathbf{g}^*)$, $\mathbf{x}^2 = \mathbf{x}^*$, $(\boldsymbol{\lambda}^2, \boldsymbol{\mu}^2, \boldsymbol{\omega}^2) = (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*)$, $LB_2 = LCG$.

4.4. Bounding Procedure H^3

Procedure H^3 uses the set of elementary routes $\mathcal{R}$. The sets $\bar{\mathcal{R}}$ and $\mathcal{N}$ are generated using the procedure GENR described below. Given a DF solution $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$ and two parameters Δ and γ, GENR generates at most Δ routes of reduced cost less than or equal to γ with respect to $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$.

The initial route set $\bar{\mathcal{R}}$ is obtained by setting $\Delta = \Delta^a$, $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}}) = (\mathbf{u}^2, \mathbf{v}^2, \mathbf{g}^2)$, and $\gamma = z(ub) - LB_2$, and adding to $\bar{\mathcal{R}}$ all single customer routes. Moreover, we initialize $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\omega}) = (\boldsymbol{\lambda}^2, \boldsymbol{\mu}^2, \boldsymbol{\omega}^2)$. In generating $\mathcal{N}$, we set $\Delta = \Delta^b$, $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}}) = (\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$, and $\gamma = -\varepsilon$ (say $\varepsilon = 10^{-6}$).

H^3 adds to LF the set $\mathcal{S}$ of CCs used by H^1 and H^2 in the form of SCs and separates WSR3s.

At the end, H^3 sets $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3) = (\mathbf{u}^*, \mathbf{v}^*, \mathbf{g}^*)$, $\mathbf{x}^3 = \mathbf{x}^*$, $(\boldsymbol{\lambda}^3, \boldsymbol{\mu}^3, \boldsymbol{\omega}^3) = (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\omega}^*)$, $LB_3 = LCG$.

4.5. Procedure GENR

GENR is a DP algorithm that generates elementary routes using bounding functions based on the ng-path relaxation. GENR is an extension of the algorithm described by Baldacci et al. (2008) for the symmetric CVRP that, in turn, is based on the method proposed by Mingozzi et al. (1994) and adapted by Baldacci et al. (2006) for the asymmetric CVRP on a multigraph. Similar methods have been used by Righini and Salani (2008), Jepsen et al. (2008), and Desaulniers et al. (2008).

In §2.3, we gave the definition of forward and backward paths. In addition, for a forward path $P = (0, i_1, \ldots, i_{k-1}, i_k)$ we refer to i_{k-1} with $\pi(P)$ and set $q(P) = \sum_{i \in V(P)} q_i$, and for a backward path $\bar{P} = \{i_k, i_{k+1}, \ldots, i_h, 0\}$ we refer to i_{k+1} with $\pi(\bar{P})$ and set $q(\bar{P}) = \sum_{i \in V(\bar{P})} q_i$.

Let τ be a time such that $e_0 < \tau < l_0$ (say, $\tau = \lfloor (l_0 - e_0)/2 \rfloor$). We define $\mathcal{F}$ as the set of all forward paths such that $\pi(P)$ is visited at time less than τ, $\forall P \in \mathcal{F}$, and $\mathcal{B}$ as the set of all backward paths such that $\pi(\bar{P})$ is visited at time greater than τ, $\forall \bar{P} \in \mathcal{B}$, plus all backward paths $\bar{P} = (k, 0)$, $\forall k \in V$. GENR is based on the observation that all feasible VRPTW routes can be obtained combining any pair of paths $(P, \bar{P})$, $P \in \mathcal{F}$, $\bar{P} \in \mathcal{B}$, satisfying the following feasibility conditions:

$$\sigma(P) = \sigma(\bar{P}), \quad V(P) \cap V(\bar{P}) = \{0, \sigma(P)\},$$
$$t(P) \leqslant t(\bar{P}), \quad \text{and} \quad q(P) + q(\bar{P}) - q_{\sigma(P)} \leqslant Q. \tag{19}$$

GENR is a two-phase algorithm. Given a DF solution $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$, it generates at most Δ routes of reduced cost less than or equal to γ. In Phase 1, it generates $\mathcal{F}$ and $\mathcal{B}$ using a procedure called GENP; in Phase 2, it derives feasible routes by combining $\mathcal{F}$ and $\mathcal{B}$ using a procedure called COMBINE. GENR is based on the following lemma.

LEMMA 1. *Let $\tilde{d}_{ij} = d_{ij} - (1/2)(\tilde{u}_i + \tilde{u}_j) - \sum_{S \in \mathcal{S}_{ij}} \tilde{v}_s$ be the arc reduced costs with respect to $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$, and let $\tilde{c}(P) = \sum_{\{i,j\} \in A(P)} \tilde{d}_{ij}$, $P \in \mathcal{F}$, and $\tilde{c}(\bar{P}) = \sum_{\{i,j\} \in A(\bar{P})} \tilde{d}_{ij}$, $\bar{P} \in \mathcal{B}$. Let $\ell \in \mathcal{R}$ be the route of reduced cost $\tilde{c}_\ell = c_\ell - \sum_{i \in V(R_\ell)} \tilde{u}_i - \sum_{S \in \mathcal{S}} \rho_\ell(S) \tilde{v}_S - \sum_{C \in \mathcal{C}_\ell} \tilde{g}_C$, where $\mathcal{S}$ corresponds to SCs, resulting from a given pair of paths P and $\bar{P}$ satisfying conditions* (19)*. The following inequalitity holds*:

$$\tilde{c}(P) + \tilde{c}(\bar{P}) + \leqslant \tilde{c}_\ell. \tag{20}$$

PROOF. The proof is provided in the e-companion to this paper. □

GENR is called (i) at the beginning of H^3 to generate the route set $\bar{\mathcal{R}}$ of the initial master problem setting $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}}) = (\mathbf{u}^2, \mathbf{v}^2, \mathbf{g}^2)$ and $\gamma = z(ub) - LB_2$; (ii) in H^3 to generate the set $\mathcal{N}$ of negative reduced cost routes setting $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}}) = (\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ and $\gamma = -\varepsilon$; (iii) at Step 2 of the exact method (see §3) to generate the route set $\mathcal{R}^3$ setting $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}}) = (\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ and $\gamma = z(ub) - LB_3$.

4.5.1. Procedure GENP. Procedure GENP is Phase 1 of GENR and computes $\mathcal{F}$ and $\mathcal{B}$ using bounding functions based on the ng-path relaxation described in §2.3.

In generating the set $\mathcal{B}$, GENP imposes that any path $\bar{P} \in \mathcal{B}$ is *not dominated* by any other path $\bar{P}' \in \mathcal{B}$ such that $V(\bar{P}) = V(\bar{P}')$, $\sigma(\bar{P}) = \sigma(\bar{P}')$, $c(\bar{P}) \geqslant c(\bar{P}')$, and $t(\bar{P}) \leqslant t(\bar{P}')$.

Let $\tilde{lb}(\bar{P})$ be a lower bound on the reduced cost $\tilde{c}_\ell$, with respect to $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$, of any route R_ℓ containing path $\bar{P}$. GENP is a Dijkstra-like algorithm (see Baldacci et al. 2008) generating $\mathcal{B}$ as a sequence of *undominated* paths $(\bar{P}^1, \ldots, \bar{P}^h)$, with $h \leqslant \Delta(\mathcal{B})$, such that $\tilde{lb}(\bar{P}^1) \leqslant \cdots \leqslant \tilde{lb}(\bar{P}^h) \leqslant \gamma$, where $\Delta(\mathcal{B})$ is a parameter. To compute $\tilde{lb}(\bar{P})$ we have to consider two cases.

Case 1: $\tilde{\mathbf{g}} = \mathbf{0}$ (i.e., $\mathcal{C} = \varnothing$), functions $f(NG, t, i)$ are computed with the modified costs $\tilde{d}_{ij}$, and the subsets N_i, $i \in V$ contain the $\Delta(N_i)$ nearest customers to i according to $\tilde{d}_{ij}$. $\tilde{lb}(\bar{P})$ is given by

$$\tilde{lb}(\bar{P}) = \tilde{c}(\bar{P}) + \min_{NG \subseteq N_i \text{ s.t. } NG \cap V(\bar{P}) = \{\sigma(\bar{P})\}} \{f(NG, t', \sigma(\bar{P}))\}. \tag{21}$$

Case 2: $\tilde{\mathbf{g}} \neq \mathbf{0}$, $\tilde{lb}(\bar{P})$ is computed extending the *ng*-relaxation to consider the dual variables $\tilde{\mathbf{g}}$. We define $N_i = \bigcup_{C \in \mathscr{C}:\ i \in C} C$, $i \in V$ and, for all $i \in V$ such that $|N_i| < \Delta(N_i)$, we add to N_i the $\Delta(N_i) - |N_i|$ nearest customers to i according to $\tilde{d}_{ij}$ not in N_i. We derive an expanded state-space graph $\tilde{\mathscr{H}} = (\tilde{\mathscr{E}}, \tilde{\Psi})$ from $\mathscr{H}$ by partitioning all paths represented by state $(NG, t, i) \in \mathscr{E}$ in $|\Gamma_i^{-1}|$ partitions, where each partition is identified by the last vertex j visited before i and is represented by a state $(NG, t, j, i) \in \tilde{\mathscr{E}}$. The set $\tilde{\mathscr{E}}$ and function $\tilde{\Psi}^{-1}(NG, t, j, i)$ are as follows:

$$\tilde{\mathscr{E}} = \{(NG, t, j, i):\ \forall\, NG \subseteq N_i \text{ s.t. } NG \ni i,\ \forall\, j \in \Gamma_i^{-1},\ \forall\, i \in V',\ \forall\, t, e_i \leqslant t \leqslant l_i]\}, \tag{22}$$

$$\tilde{\Psi}^{-1}(NG, t, j, i) = \{(NG', t', k, j):\ \forall\, NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \backslash \{i\},\ \forall\, k \in \Gamma_j^{-1},\ \forall\, j \in \Gamma_i^{-1},\ \forall\, t' \in \Omega(t, j, i)\}, \quad \forall\, (NG, t, j, i) \in \tilde{\mathscr{E}}. \tag{23}$$

Then, $\tilde{\Psi} = \{((NG', t', k, j), (NG, t, j, i)):\ \forall\, (NG', t', k, j) \in \tilde{\Psi}^{-1}(NG, t, j, i),\ \forall\, (NG, t, j, i) \in \tilde{\mathscr{E}}\}$. Let $\mathscr{C}_{ij} = \{C \in \mathscr{C}:\ (i, j) \in A(C)\}$. The DP recursion (10) can be easily modified to compute the cost $f(NG, t, j, i)$ of the least-cost *ng*-path (NG, t, j, i) as follows:

$$f(NG, t, j, i) = \min_{(NG', t', k, j) \in \tilde{\Psi}^{-1}(NG, t, j, i)} \left\{ f(NG', t', k, j) + \tilde{d}_{ji} - \sum_{C \in \mathscr{C}_{ji} \backslash \mathscr{C}_{kj}} \tilde{g}_C \right\}, \quad \forall\, (NG, t, j, i) \in \tilde{\mathscr{E}}. \tag{24}$$

The following lemma gives a method for computing $\tilde{lb}(\bar{P})$.

Lemma 2. *Let $i = \sigma(\bar{P})$ and $k = \pi(\bar{P})$. Lower bound $\tilde{lb}(\bar{P})$ can be computed as follows*:

$$\tilde{lb}(\bar{P}) = \tilde{c}(\bar{P}) - \sum_{\substack{C \in \mathscr{C} \text{ s.t.} \\ |A(C) \cap A(\bar{P})| \geqslant 1}} \tilde{g}_C + \min_{\substack{NG \subseteq N_i \text{ s.t. } NG \cap V(\bar{P}) = \{i\} \\ t' \leqslant t(\bar{P}),\ j \in \Gamma_i^{-1}}} \{ f(NG, t', j, i) + \tilde{g}_{\{jik\}} \}, \tag{25}$$

where $\tilde{g}_{\{jik\}}$ is the dual of inequality (6) *corresponding to $C = \{j, i, k\}$ ($\tilde{g}_{\{jik\}} = 0$ if $\{j, i, k\} \notin \mathscr{C}$).*

Proof. The proof is provided in the e-companion to this paper. □

Similarly, GENP generates the path set $\mathscr{F}$ as a sequence of *undominated* paths $(P^1, \ldots, P^h)$, with $h \leqslant \Delta(\mathscr{F})$, such that $\tilde{lb}(P^1) \leqslant \cdots \leqslant \tilde{lb}(P^h) \leqslant \gamma$, where $\Delta(\mathscr{F})$ is a parameter. $\tilde{lb}(P)$ is computed similarly as described above for $\mathscr{B}$ using the reverse functions $f^{-1}(NG, t, i)$ and $f^{-1}(NG, t, j, i)$.

4.5.2. Procedure COMBINE**.** This procedure combines the path sets $\mathscr{F}$ and $\mathscr{B}$ to derive at most Δ routes of reduced cost with respect to $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$ less than or equal to γ using the iterative method described by Baldacci et al. (2006).

Because of Lemma 1, routes of reduced cost less than or equal to γ can be generated combining only path pairs $(P, \bar{P})$ such that $\tilde{c}(P) + \tilde{c}(\bar{P}) \leqslant \gamma$. COMBINE dynamically generates a sequence of pairs $(P^{r_1}, \bar{P}^{s_1}), \ldots, (P^{r_k}, \bar{P}^{s_k}), \ldots, (P^{r_h}, \bar{P}^{s_h})$ such that each pair satisfies conditions (19) and $\tilde{c}(P^{r_1}) + \tilde{c}(\bar{P}^{s_1}) \leqslant \cdots \leqslant \tilde{c}(P^{r_k}) + \tilde{c}(\bar{P}^{s_k}) \leqslant \cdots \leqslant \tilde{c}(P^{r_h}) + \tilde{c}(\bar{P}^{s_h}) \leqslant \gamma$. The pool of routes generated by COMBINE contains any route R resulting from the pairs of paths in the sequence, such that $\tilde{c}(R) \leqslant \gamma$ and R is *not dominated* by any other route R' previously generated (i.e., R' *dominates* R if $V(R') = V(R)$ and $\tilde{c}(R') \leqslant \tilde{c}(R)$).

The procedure terminates as soon as Δ routes have been found or all pairs have been considered.

4.5.3. GENR **for the** CVRP**.** Unlike the VRPTW, the path sets $\mathscr{F}$ and $\mathscr{B}$ are defined as in Baldacci et al. (2008): $\mathscr{F}$ contains any forward path P such that $q(P) \leqslant Q/2 + q_{\sigma(P)}$, and $\mathscr{B}$ contains any backward path $\bar{P}$ such that $q(\bar{P}) \leqslant Q/2 + q_{\sigma(\bar{P})}$ (for symmetric CVRPS $\mathscr{F}$ and $\mathscr{B}$ coincide).

Moreover, the lower bound $\tilde{lb}(\bar{P})$ is computed with bounding functions $f(NG, q, i)$, if $\tilde{\mathbf{g}} = \mathbf{0}$, or $f(NG, q, j, i)$, if $\tilde{\mathbf{g}} \neq \mathbf{0}$. Functions $f(NG, q, j, i)$ are derived by expanding $f(NG, q, i)$ so that j is the vertex preceding i as described in §4.5.1 for functions $f(NG, t, j, i)$.

5. Column-and-Cut Generation Procedure H^4

H^4 is a column-and-cut generation procedure based on the simplex algorithm to solve relaxation LF. H^4 differs from the methods of Jepsen et al. (2008), Desaulniers et al. (2008), and Baldacci et al. (2008) for the strategy used for solving the pricing problem.

Before starting H^4 (see Step 2 of the exact algorithm in §3) an attempt is made to generate the set $\mathscr{R}^3$ of all routes such that $c^3_\ell \leqslant z(ub) - LB_3$. The set $\mathscr{R}^3$ is generated by GENR imposing that $|\mathscr{F}| \leqslant \Delta(\mathscr{F})$, $|\mathscr{B}| \leqslant \Delta(\mathscr{B})$, and that at most $\Delta(\mathscr{R})$ are generated. We call $\mathscr{F}^3$ the set $\mathscr{F}$, $\mathscr{B}^3$ the set $\mathscr{B}$, and $\mathscr{R}^3$ the set $\mathscr{R}$ generated by GENR. Define each set $\mathscr{F}^3$, $\mathscr{B}^3$, and $\mathscr{R}^3$ as *optimal* if $|\mathscr{F}^3| < \Delta(\mathscr{F})$, $|\mathscr{B}^3| < \Delta(\mathscr{B})$, and $|\mathscr{R}^3| < \Delta(\mathscr{R})$. The set $\mathscr{R}^3$ is optimal if both $\mathscr{F}^3$ and $\mathscr{B}^3$ are optimal.

The route set $\bar{\mathscr{R}}$ of the initial master problem of H^4 is obtained by extracting the Δ^a routes of minimum reduced cost with respect to $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ from $\mathscr{R}^3$ and by adding all single customer routes. We initialize the set $\mathscr{C}$ of SR3s as $\mathscr{C} = \varnothing$ and use the same set $\mathscr{S}$ of SCs of H^3.

At each iteration, a set $\mathscr{N}$ of at most Δ^b negative reduced cost routes with respect to the current dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{g}}, \bar{\mathbf{v}})$ is produced by procedure GENR4 (see §5.1). The set $\mathscr{N}$ is either extracted from $\mathscr{R}^3$ or generated by combining sets

$\mathcal{F}^3$ and $\mathcal{B}^3$. The sets $\mathcal{F}^3$ and $\mathcal{B}^3$ are newly generated when necessary if they are not-optimal.

At each iteration, H^4 adds the set $\mathcal{C}'$ of at most $\Delta(\mathcal{C})$ SR3s most violated by the current LF solution. H^4 ends if $\mathcal{N}=\varnothing$ and $\mathcal{C}'=\varnothing$ and achieves a DF solution $(\mathbf{u}^4, \mathbf{v}^4, \mathbf{g}^4)$ of cost LB_4.

5.1. Procedure GENR4

Procedure GENR4 is called at each iteration of H^4 to generate the set $\mathcal{N}$ of at most Δ^b routes of negative reduced cost with respect to the current dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ of the master problem.

GENR4 performs the following steps:

1. Extract from $\mathcal{R}^3$ a set of at most Δ^b routes of negative reduced cost and add them to $\mathcal{N}$. If $\mathcal{N}\neq\varnothing$ or $\mathcal{R}^3$ is optimal, stop.
2. Call procedure COMBINE4 that combines the sets $\mathcal{F}^3$ and $\mathcal{B}^3$ to derive the set $\mathcal{N}$. If $\mathcal{N}\neq\varnothing$ or both sets $\mathcal{F}^3$ and $\mathcal{B}^3$ are optimal, stop.
3. If $\mathcal{F}^3$ is not-optimal, call procedure GENP4 to generate a new path set $\mathcal{F}$ and set $\mathcal{F}^3=\mathcal{F}$ ($\mathcal{F}^3$ is still not-optimal).
4. Similarly to the previous step, if $\mathcal{B}^3$ is not-optimal, call procedure GENP4 to generate a new path set $\mathcal{B}$ and set $\mathcal{B}^3=\mathcal{B}$ ($\mathcal{B}^3$ is still not-optimal).
5. Call procedure COMBINE4 that combines the sets $\mathcal{F}^3$ and $\mathcal{B}^3$ to derive the set $\mathcal{N}$.

If GENP4 or COMBINE4 run out of memory, H^4 and the exact method terminate prematurely.

5.2. Procedure GENP4

Procedure GENP4 generates sets $\mathcal{F}$ and/or $\mathcal{B}$ in steps 3 and 4 of GENR4. GENP4 is similar to procedure GENP but applies different fathoming rules and an additional dominance rule introduced by Jepsen et al. (2008). In generating $\mathcal{B}$, procedure GENP4 applies the following rules.

Let $lb^3(\bar{P})$, $\bar{P}\in\mathcal{B}$, be a lower bound to the reduced cost with respect to the dual solution $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ found by H^3 of any route R containing path $\bar{P}$. Lower bound $lb^3(\bar{P})$ corresponds to $\tilde{lb}(\bar{P})$ described in §4.5.1 when $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$ is replaced with $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$.

FATHOMING 1. Any path $\bar{P}\in\mathcal{B}$ such that $lb^3(\bar{P}) > z(ub)-LB_3$ can be fathomed because $\bar{P}$ cannot generate any route of any VRPTW solution of cost less than or equal to $z(ub)$.

Let $\bar{lb}(\bar{P})$, $\bar{P}\in\mathcal{B}$, be a lower bound to the reduced cost with respect to the dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ of any route containing path $\bar{P}$.

FATHOMING 2. Any path $\bar{P}$ such that $\bar{lb}(\bar{P})\geqslant 0$ is fathomed.

Lower bound $\bar{lb}(\bar{P})$ can be computed as described in the following lemma.

LEMMA 3. *Let functions $f(NG,t,i)$ be computed with the arc costs $\bar{d}_{ij}=d_{ij}-(1/2)(\bar{u}_i+\bar{u}_j)-\sum_{S\in\mathcal{S}_{ij}}\bar{v}_S$. Let $\bar{c}(\bar{P})$ be the cost of path $\bar{P}$ using arc costs $\bar{d}_{ij}$, and let $i=\sigma(\bar{P})$. Lower bound $\bar{lb}(\bar{P})$ can be computed as follows*:

$$\bar{lb}(\bar{P})=\bar{c}(\bar{P})-\sum_{\substack{C\in\mathcal{C}\ s.t.\\ |C\cap V(\bar{P})\setminus\{i\}|\geqslant 2}}\bar{g}_C + \min_{\substack{NG\subseteq N_i\ s.t.\ NG\cap V(\bar{P})=\{i\}\\ t'\leqslant t(\bar{P})}}\left\{f(NG,t',i)-\sum_{\substack{C\in\mathcal{C}\ s.t.\\ |C\cap NG|\geqslant 2}}\bar{g}_C\right\}. \quad (26)$$

PROOF. The proof is provided in the e-companion to this paper. □

The dominance rule of Jepsen et al. (2008) when applied to $\mathcal{B}$ is as follows.

DOMINANCE 1. Let $\bar{P},\bar{P}'\in\mathcal{B}$, be two backward paths such that $\sigma(\bar{P}')=\sigma(\bar{P})$, $t(\bar{P}')\geqslant t(\bar{P})$, $q(\bar{P})\geqslant q_{\min}$, $q(\bar{P}')\geqslant q_{\min}$, and $V(\bar{P}')\subseteq V(\bar{P})$. $\bar{P}'$ dominates $\bar{P}$ if

$$\bar{c}(\bar{P}')\leqslant\bar{c}(\bar{P})-\sum_{\substack{C\in\mathcal{C}\ \text{s.t.}\\ |C\cap V(\bar{P})\setminus V(\bar{P}')|\geqslant 2}}\bar{g}_C. \quad (27)$$

Similar rules are applied by procedure GENP4 to generate the path set $\mathcal{F}$.

5.3. Procedure COMBINE4

This procedure generates the route set $\mathcal{N}$ of negative reduced costs with respect to the dual solution $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ of the master problem combining the path sets $\mathcal{F}^3$ and $\mathcal{B}^3$ as defined in §5.1. COMBINE4 corresponds to procedure COMBINE (see §4.5.2) replacing $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{g}})$ with $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$, setting $\gamma=-\varepsilon$, and adding the following fathoming rule to reduce the set $\mathcal{N}$.

FATHOMING 3. Let $c^3(R)$ be the reduced cost of route R with respect to $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ computed as $c^3(R)=c(R)-\sum_{i\in V(R)}u_i^3-\sum_{S\in\mathcal{S}}\rho_\ell(S)v_S^3-\sum_{C\in\mathcal{C}\ \text{s.t.}\ |A(C)\cap A(R)|\geqslant 1}g_C^3$. A route R of negative reduced cost with respect to $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ cannot belong to $\mathcal{N}$ if $c^3(R)>z(ub)-LB_3$ as R cannot be in any solution of cost less than or equal to $z(ub)$.

5.4. Procedure GENR4 for the CVRP

The lower bound $lb^3(\bar{P})$ is computed using bounding functions $f(NG,q,j,i)$ as described in §4.5.3 for $\tilde{lb}(\bar{P})$, while the lower bound $\bar{lb}(\bar{P})$ is computed according to expression (26) but using bounding functions $f(NG,q,i)$ instead of $f(NG,t,i)$.

6. Procedure GENRF for Generating Route Set $\hat{\mathcal{R}}$

The exact method described in §3 at Step 4 asks to generate the largest subset $\hat{\mathcal{R}}\subseteq\mathcal{R}$ of routes satisfying conditions (15), whenever $\mathcal{R}^3$ is not-optimal.

For this purpose, we use a two-phase procedure, called GENRF similar to GENR (see §4.5). In the first phase, two sets $\mathcal{F}$ and $\mathcal{B}$ of forward and backward paths are computed as described below. In the second phase, such sets are combined by procedure COMBINEF (see §6.2) to derive the final route set $\hat{\mathcal{R}}$. In the first phase, there are four cases:

(A) Both $\mathcal{F}^3$ and $\mathcal{B}^3$ generated by GENR are optimal. We set $\mathcal{F} = \mathcal{F}^3$ and $\mathcal{B} = \mathcal{B}^3$.

(B) $\mathcal{B}^3$ is optimal but $\mathcal{F}^3$ is not. We set $\mathcal{B} = \mathcal{B}^3$ and call GENPF (§6.1) to compute $\mathcal{F}$.

(C) $\mathcal{F}^3$ is optimal but $\mathcal{B}^3$ is not. We set $\mathcal{F} = \mathcal{F}^3$ and call GENPF to compute $\mathcal{B}$.

(D) Both $\mathcal{F}^3$ and $\mathcal{B}^3$ are not optimal. We call GENPF to compute both $\mathcal{F}$ and $\mathcal{B}$.

In the end, $\hat{\mathcal{R}}$ is defined *optimal* (i.e., contains any optimal VRPTW solution) if and only if $\mathcal{F}$, $\mathcal{B}$ and $\hat{\mathcal{R}}$ are such that $|\mathcal{F}| < \Delta(\mathcal{F})$, $|\mathcal{B}| < \Delta(\mathcal{B})$ and $|\hat{\mathcal{R}}| < \Delta(\mathcal{R})$.

In the following §§6.1 and 6.2, we describe procedure GENPF and COMBINEF for the VRPTW. It is obvious how to adapt them for the CVRP.

6.1. Procedure GENPF

Procedure GENPF generates one or both sets $\mathcal{F}$ and $\mathcal{B}$ required by GENRF (see cases (B), (C), and (D) in §6). GENPF is similar to procedure GENP but applies different fathoming rules.

In generating the set $\mathcal{B}$, GENPF applies the following fathoming rules. Let $lb^3(\bar{P})$ and $lb^4(\bar{P})$ be the lower bounds on the reduced costs, $c^3(R)$ and $c^4(R)$, of any route R containing path $\bar{P} \in \mathcal{B}$ with respect to $(\mathbf{u}^3, \mathbf{v}^3, \mathbf{g}^3)$ and $(\mathbf{u}^4, \mathbf{v}^4, \mathbf{g}^4)$, respectively. $lb^3(\bar{P})$ is computed as described in §5.2, and $lb^4(\bar{P})$ is computed using expression (26), where $(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{g}})$ is replaced with $(\mathbf{u}^4, \mathbf{v}^4, \mathbf{g}^4)$.

FATHOMING 4. Any path $\bar{P} \in \mathcal{B}$ such that $lb^3(\bar{P}) > z(ub) - LB_3$ or $lb^4(\bar{P}) > z(ub) - LB_4$ can be fathomed.

FATHOMING 5. Let $d^4_{ij} = d_{ij} - (1/2)(u^4_i + u^4_j) - \sum_{S \in \mathcal{S}_{ij}} v^4_s$, $\forall (i, j) \in A$. Whenever $q_{\min} = 0$, any backward path $\bar{P}$ such that

$$\sum_{(i,j) \in A(\bar{P})} d^4_{ij} - \sum_{\substack{C \in \mathcal{C} \text{ s.t.} \\ |C \cap V(\bar{P}) \setminus \{i\}| \geq 2}} g^4_C - d^4_{i0} > z(ub) - LB_4 \qquad (28)$$

can be fathomed as it cannot produce any route R of reduced cost $c^4(R) \leqslant z(ub) - LB_4$.

PROOF. The proof is provided in the e-companion to this paper. □

Similar fathoming rules are used to generate $\mathcal{F}$.

6.2. Procedure COMBINEF

This procedure generates the route set $\hat{\mathcal{R}}$ combining the path sets $\mathcal{F}$ and $\mathcal{B}$ defined in §6. COMBINEF is similar to COMBINE (see §4.5.2) but imposes that any route $R \in \hat{\mathcal{R}}$ is such that $c^3(R) \leqslant z(ub) - LB_3$ and $c^4(R) \leqslant z(ub) - LB_4$.

7. Computational Results

This section reports on the computational results of the exact method (hereafter called BMR) described in this paper. All algorithms were coded in Fortran 77 and compiled with Intel Fortran 11.0. CPLEX 12.1 (see CPLEX 2009) was used as the LP solver in H^4 and the IP solver in the exact method. All tests were run on an IBM Intel Xeon X7350 server (2.93 GHz–16 GB of RAM).

7.1. Computational Results on the VRPTW

Our exact method BMR for the VRPTW was tested on Solomon instances (see Solomon 1987), which are divided into six classes (C1, RC1, and R1 with tight time windows and strict vehicle capacity, and C2, RC2, and R2 with wide time windows and loose vehicle capacity). We considered all 100-customer instances and instances with 50 customers of classes C2, RC2, and R2.

The travel costs d_{ij} are computed as $d_{ij} = \lfloor 10 e_{ij} \rfloor / 10$, where e_{ij} is the Euclidean distance between vertices i and j; the travel times t_{ij} are integer values computed as $t_{ij} = 10(d_{ij} + s_i)$, where s_i is the service time at vertex i.

Because instances of classes C1, RC1, and R1 have tight time windows, we did not find it worth running procedure H^2. Thus, on such instances H^2 was skipped. In addition, we found it to be computationally convenient to ignore SC and WSR3 inequalities for all classes of instances.

BMR uses the best upper bounds reported in Ropke (2005) and Danna and Le Pape (2005). Such upper bounds are obtained by running the heuristic of Pisinger and Ropke (2007) with different parameter settings (Ropke 2010). Whenever BMR uses the upper bound, its computing time is added to the total computing time of BMR. For instance R211 with 100 customers, the upper bound used was found by Desaulniers et al. (2008).

BMR uses the following parameter setting: In H^2, $\Delta(N_i) = 8$; in H^3, $\Delta(N_i) = 10$, $\Delta^b = 300$, $\Delta(\mathcal{F}) = \Delta(\mathcal{B}) = 5 \times 10^7$, $\Delta(\mathcal{R}) = 1.5 \times 10^6$; in H^4, $\Delta^a = 1 \times 10^4$, $\Delta^b = 300$, $\Delta(\mathcal{F}) = \Delta(\mathcal{B}) = 5 \times 10^7$, $\Delta(\mathcal{R}) = 1.5 \times 10^6$, $\Delta(\mathcal{C}) = 20$. In H^1, H^2, and H^3, we set $Maxit1 = 100$ and $Maxit2 = 50$.

We compare BMR with the methods of Jepsen et al. (2008) and Desaulniers et al. (2008), hereafter called JPSP and DHL, respectively. Desaulniers et al. (2008) presented three versions of their algorithm. We consider the version labeled "ESPPRC SRC" because it could solve more instances than the others. According to SPEC (http://www.spec.org/benchmarks.html), our machine is three times faster than the Intel Pentium 4 3.0-GHz PC of JPSP and twice as fast as the Linux PC Dual Core AMD Opteron at 2.6 GHz of DHL.

In Table 1, we report on detailed results on instances closed for the first time by JPSP, DHL, or BMR. Complete computational results are reported in the e-companion paper.

The columns of Table 1 report the instance name (*Inst*), the optimal value (z^*—in bold if solved for the first time

Table 1. Results on selected VRPTW Solomon instances.

		Upper bound		*Proc.* H^1		*Proc.* H^2		*Proc.* H^3					*Proc.* H^4			*Final problem* $\hat{F}$		*Tot. time* (*sec*)		
Inst	z^*	$z(ub)$	*Time*	LB_1	*Time*	LB_2	*Time*	LB_3	*Time*	$\lvert\mathcal{F}^3\rvert$	$\lvert\mathcal{B}^3\rvert$	$\lvert\mathcal{R}^3\rvert$	*SR*3	LB_4	*Time*	$\lvert\hat{\mathcal{R}}\rvert$	T_{CPX}	*BMR*	*JPSP*	*DHL*
R.207.50	575.5	575.5	102	482.2	3	561.4	11	564.1	15	786	136	∘	70	575.5	101			154	34,406	n.a.
R.208.50	487.7	487.7	96	461.4	6	476.3	28	481.3	94	7,385	3,622	1,282	55	487.7	441			537	—	n.a.
R.108.100	932.1	933.7	119	905.7	6			913.0	10	1,882	2,066	∘	280	932.1	225			344	5,911	891
R.112.100	948.6	948.6	116	916.6	6			926.2	11	2,115	4,311	∘	539	946.6	721	6,214	28	865	202,803	16,073
RC.203.100	923.7	923.7	185	814.3	7	916.0	16	919.5	22	119	29	269	30	923.7	67			252	14,917	324
RC.204.100	**783.5**	783.5	290	687.4	20	771.1	235	778.4	455	1,375	339	∘	50	783.5	902			1,052	—	—
RC.206.100	1,051.1	1,051.1	172	951.5	8	1,034.4	13	1,037.7	16	276	177	∘	40	1,051.1	55			227	339	344
RC.207.100	962.9	966.6	182	865.7	9	940.6	25	945.8	62	∘	∘	∘	490	962.9	26,721			26,903	—	91,405
RC.208.100	**776.1**	777.3	215	703.9	20	761.3	114	765.8	168	3,909	3,306	∘	110	776.1	980	6,959		1,195	—	—
R.202.100	1,029.6	1,036.9	178	1,008.6	8	1,020.6	19	1,021.2	22	∘	∘	∘	115	1,027.3	1,890	575,375	1,338	3,406	8,282	1,663
R.203.100	870.8	872.4	198	845.7	9	862.8	91	865.8	165	∘	564	∘	100	870.8	1,941			2,139	54,187	641
R.204.100	**731.3**	731.3	221	688.8	22	721.3	64	724.2	194	∘	∘	∘	130	731.3	216,146			216,367	—	—
R.205.100	949.8	949.8	191	916.0	11	934.1	27	938.0	31	3,779	920	∘	553	948.3	1,240	19,034	2	1,433	—	6,904
R.206.100	875.9	880.6	201	834.0	13	860.2	34	866.3	76	∘	∘	∘	145	875.9	6,474			6,675	—	60,608
R.207.100	794.0	794.0	238	746.9	16	781.6	120	789.9	368	∘	∘	∘	35	794.0	1,163			1,401	—	11,228
R.209.100	854.8	855.7	136	818.7	15	838.0	28	840.6	59	∘	∘	∘	180	854.3	4,355	120,577	5	4,496	78,560	22,514
R.210.100	900.5	900.8	136	848.9	17	883.8	30	888.2	57	∘	∘	∘	523	900.4	39,572	53,704	3	39,711	—	400,904
R.211.100	**746.7**	751.7	168	704.9	21	729.3	133	734.1	219	∘	∘	∘	140	746.7	10,825			10,993	—	—
				93.3		**98.3**		**98.7**						**99.9**						

by BMR), the upper bound used ($z(ub)$), and the time to compute it. For each procedure H^k, $k = 1, \ldots, 4$ (if run), it is reported the lower bound (LB_k) and the cumulative computing time spent up to H^k. Columns $|\mathcal{F}^3|$, $|\mathcal{B}^3|$, and $|\mathcal{R}^3|$ report the cardinalities (in thousands) of the sets $\mathcal{F}^3$, $\mathcal{B}^3$, and $\mathcal{R}^3$; if we could not completely generate a set, an empty circle is displayed. The number of SR3 inequalities ($SR3$) added in H^4 is shown. The number of routes ($|\hat{\mathcal{R}}|$) in the final reduced problem $\hat{F}$ and the time taken by CPLEX to solve it (T_{CPX}) are shown. Finally, the total computing time in seconds (*Tot. time*) of the methods compared are reported in the last three columns of the table. *Tot. time* under BMR is equal to the sum of the time to compute the upper bound, the time spent up to H^4 and T_{CPX}.

Table 1 shows that BMR was able to solve four instances open so far. The only open Solomon instance is R.208.100, where BMR ran out of memory. Moreover, it can be noticed that the lower bounds achieved using the *ng*-routes in algorithm CCG are close to the bounds achieved using elementary routes and the time taken to perform H^2 is limited.

Table 3 compares BMR, JPSP, and DHL. For each class, Table 3 reports the class name ($Class$), the number of customers (n), the number of instances (NP), the number of instances solved by each of the three methods ($Solved$) and the average computing time in seconds ($Time$) ($n.a.$ means data are not available). In the last three rows, the average computing time of the methods over all instances, the instances solved by JPSP, and the instances solved by DHL are shown.

Table 3 shows that BMR outperforms JPSP and DHL; all instances solved by the other methods were solved by BMR, and the average time is significantly lower.

In the e-companion to this paper, we analyse the impact of parameter $\Delta(N_i)$ on procedure H^2 and the effectiveness of dominance and fathoming rules on procedures GENP4 and GENPF. The results show that (i) in the NG-route relaxation, $\Delta(N_i) = 8$ gives a good trade-off between lower bounds and computing times as $LB2$ is about 5% greater than LB_1 and about 0.5% lower than LB_3; (ii) in GENP4, the Dominance 1 and the new fathoming rules 1 and 2 eliminate about 90% of the states, and the fathoming rules eliminate 80% of the states not dominated by Dominance 1; (iii) in GENPF, both fathoming rules 5 and 4 eliminate about 90% of the states.

7.2. Computational Results on the CVRP

We considered classes A, B, E, F, M, and P, available at http://branchandcut.org/VRP/data. Cost d_{ij} is an integer value computed as $d_{ij} = \lfloor e_{ij} + 0.5 \rfloor$, where e_{ij} is the Euclidean distance between i and j. As done by Fukasawa et al. (2006), we impose that exactly m vehicles are used in the solution by simply transforming constraint (3) into an equality constraint.

For computational convenience we skip bounding procedure H^2, so the sequence of the bounding procedures is H^1, H^3, and H^4. WSR3 inequalities are used whenever $\lceil n/m \rceil \geqslant 12$.

If BMR cannot solve a problem in Step 1, we use the upper bound used by Fukasawa et al. (2006) and Baldacci et al. (2008), but, while generating the route set $\mathcal{R}^3$, whenever in GENP $|\mathcal{F}^3| > 1 \times 10^6$, we run our implementation of the tabu search algorithm of Gendreau et al. (1994) with a time limit of 180 seconds. The computing time of the tabu search is considered in the total computing time of BMR while the computing time of the initial upper bound is ignored (as done by Fukasawa et al. 2006 and Baldacci et al. 2008).

In our tests, we use the following parameter setting: in H^3, $\Delta(N_i) = 10$, $\Delta^b = 300$, $\Delta(\mathcal{F}) = 1 \times 10^7$, $\Delta(\mathcal{R}) = 1 \times 10^7$, $\Delta(\mathcal{C}) = 100$; in H^4, $\Delta^a = 1 \times 10^4$, $\Delta^b = 200$, $\Delta(\mathcal{F}) = 1 \times 10^7$, $\Delta(\mathcal{R}) = 1 \times 10^6$, $\Delta(\mathcal{C}) = 10$. In H^1, H^2, and H^3, $Maxit1 = 150$ and $Maxit2 = 100$.

BMR is compared with the methods of Baldacci et al. (2008) (BCM), Fukasawa et al. (2006) (FLL), and Lysgaard et al. (2004) (LLE). According to SPEC, our machine is three times faster than the Pentium 4 2.6-GHz PC of BCM and the Pentium 4 2.4-GHz PC of FLL, and 10 times faster than the Intel Celeron 700-MHz PC of LLE.

In Table 2, we report detailed results on difficult CVRP instances. Complete computational results are reported in the e-companion. Table 2 is similar to Table 1. Column $z(ub)$ reports the initial upper bound while column UB_4 reports the upper bound computed by our tabu search heuristic (when run). Under FLL, column (s) indicates the BCP algorithm based on s-cycle-free q-routes was used, while ($-$) indicates that the BC was used instead of the BCP. The *Tot. time* of BMR is the time spent up to H^4 (including the time to compute UB_4) plus T_{CPX}.

Table 2 shows that BMR solves three problems not solved by BCM but does not solve problem F-n135-k7 that is solved by FLL using the BC algorithm. It is worth mentioning that problem F-n135-k7 was solved for the first time by Augerat et al. (1995).

Table 4 summarizes the results of the methods on the six CVRP classes considered. For each class, the name ($Class$), the number of instances (NP), and for each exact method, the number of instances solved to optimality (Opt), the average percentage deviation of the lower bound ($\%LB$), and the average computing time in seconds ($Time$) are reported. Under FLL, column Opt_{BCP} and Opt_{BC} report the number of instances solved using BCP and BC, respectively. The last two rows indicate average values of $\%LB$ and $Time$ and the number of problems solved by each method.

Table 4 clearly shows that BMR outperforms the other methods on all classes but class F, where the BC algorithm of Lysgaard et al. (2004) outperforms the other methods.

The impact of parameter $\Delta(N_i)$ on procedure H^2 and the effectiveness of the different type of inequalities on procedures H^1 and H^3 are analysed in the e-companion to this paper. The results show that (i) increasing parameter $\Delta(N_i)$ slightly improves lower bound LB_2; (ii) LB_1 with

Table 2. Results on selected CVRP instances

			Proc. H^1		*Proc.* H^3				*Proc.* H^4				*Final problem* $\hat{F}$		*Tot. time (sec)*			
Inst	$z(ub)$	z^*	LB_1	*Time*	LB_3	*Time*	$\lvert\mathcal{F}^3\rvert$	$\lvert\mathcal{R}^3\rvert$	*SR*3	LB_4	UB_4	*Time*	$\lvert\hat{\mathcal{R}}\rvert$	T_{CPX}	*BMR*	*BCM*	*FLL (s)*	*LLE*
B-n50-k8	1,312	1,312	1,291.1	11	1,302.6	24	1,011	334	92	1,308.8	1,312	118	23,073	29	147	662	2,845 (3)	31,026
B-n66-k9	1,316	1,316	1,301.7	17	1,307.7	27	2,547	1,331	137	1,315.2	1,330	283	13,442	9	292	227	1,778 (3)	24,424
B-n68-k9	1,275	1,272	1,261.1	23	1,263.2	34	2,267	1,103	141	1,267.6	1,272	336	55,097	190	526	6,168	87,436 (3)	—
B-n78-k10	1,221	1,221	1,208.4	23	1,214.9	36	266	71	52	1,221.0		91	11,131	6	97	229	1,053 (3)	87,408
E-n76-k7	682	682	667.2	5	669.8	28	910	∘	193	680.6		151	3,597	1	152	3,370	46,520 (2)	118,683
E-n76-k8	735	735	724.3	13	725.1	45	218	1,249	160	734.0		81	1,765	1	82	873	22,891 (2)	—
E-n76-k10	830	830	815.4	18	816.5	54	317	1,476	143	826.2		81	10,696	33	114	174	80,722 (3)	—
E-n76-k14	1,021	1,021	1,004.3	15	1,007.0	36	109	255	93	1,014.7		42	8,965	10	52	44	48,637 (3)	—
E-n101-k8	815	815	801.8	30	808.8	218	1,584	∘	118	815.0	818	579			579	—	801,963 (3)	—
E-n101-k14	1,071	1,067	1,049.6	10	1,052.9	45	2,272	∘	198	1,062.3	1,067	319	56,692	134	453	1,230	116,284 (3)	—
F-n135-k7	1,162	1,162	1,158.0	1,057	m.o.										—		7,065 (–)	3,092
M-n121-k7	1,034	1,034	1,028.5	94	1,032.5	611	2,961	12	36	1,033.4	1,035	1,247	12,440	2	1,249	2,448	25,678 (3)	—
P-n50-k8	649	631	613.4	6	615.9	15	47	70	113	625.1	631	64	2,993	4	68	596	9,272 (3)	—
P-n55-k10	699	694	677.8	5	680.6	15	112	314	84	689.1		19	18,061	10	29	66	9,076 (3)	—
P-n70-k10	834	827	811.5	13	813.6	34	147	483	129	823.0	827	150	7,925	16	166	774	24,039 (3)	—
P-n76-k5	627	627	614.1	5	619.5	87	905	3,707	137	627.0		282	401	<0.1	282	—	14,546 (–)	10,970
P-n101-k4	681	681	668.5	13	678.7	73	4,475	8,811	106	681.0	681	1,154	458	1	1,155	—	1,253 (–)	281
			98.5		**98.9**					**99.7**								

Note. m.o.: GENP runs out of memory.

Table 3. VRPTW Solomon instances: summary.

			Solved			*Time*		
Class	*n*	*NP*	*BMR*	*JPSP*	*DHL*	*BMR*	*JPSP*	*DHL*
C2	50	8	8	7	n.a.	8	79	n.a.
RC2	50	8	8	7	n.a.	27	268	n.a.
R2	50	11	11	9	n.a.	124	7,086	n.a.
C1	100	9	9	9	9	25	468	18
RC1	100	8	8	8	8	276	11,004	2,150
R1	100	12	12	12	12	251	27,412	2,327
C2	100	8	8	7	8	40	2,795	2,093
RC2	100	8	8	5	6	3,767	3,204	15,394
R2	100	11	10	4	8	28,680	35,292	63,068
Avg						3,955	9,767	12,920
Solved by JPSP						261	9,767	
Solved by DHL						1,825		12,920

Table 4. CVRP instances: summary.

		BMR			*BCM*			*FLL*					*LLE*		
Class	*NP*	*Opt*	*%LB*	*Time*	*Opt*	*%LB*	*Time*	*Opt*	Opt_{BCP}	Opt_{BC}	*%LB*	*Time*	*Opt*	*%LB*	*Time*
A	22	22	99.9	30	22	99.8	118	22	20	2	99.2	1,961	15	97.9	6,638
B	20	20	99.9	67	20	99.8	417	20	6	14	99.5	4,763	19	99.4	8,178
E-M	12	9	99.8	303	8	99.4	1,025	9	7	2	98.9	126,987	3	97.7	39,592
F	3	2	100.0	164				3	0	3	99.9	2,398	3	99.9	1,046
P	24	24	99.8	85	22	99.7	187	24	16	8	99.2	2,892	16	97.7	11,219
Avg			99.9	92		99.7	323				99.3	17,409		98.4	9,935
Tot	81	77			72			78	49	29			56		

CCS is very close to LB_3 without WSR3s; (iii) the increase of LB_2 with respect to LB_1 achieved by H^1 with CCS is very small and is not worth the extra computing time required by procedure H^2; (iv) CCS and WSR3s substantially increase lower bounds LB_1 and LB_3, respectively.

8. Conclusions

In this paper, we described an exact method for solving the VRPTW and CVRP based on the set partitioning formulation strengthened with valid inequalities. We introduced a new route relaxation, called *ng*-route, that improves other nonelementary route relaxations proposed in the literature and a new strategy for solving the pricing problem in a column-and-cut generation procedure that involves the use of multiple dual solutions.

We reported computational results showing that the proposed method solves four of the five open Solomon VRPTW instances and is significantly faster than the state-of-the-art algorithms for both VRPTW and CVRP.

9. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/.

Acknowledgments

The authors thank Stefan Røpke for providing them with the VRPTW upper bounds and Simon Spoorendonk for his helpful comments on a preliminary version of this paper. Thanks are also due to two anonymous referees for making several suggestions that improved the presentation of the paper.

References

Augerat, P., J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, G. Rinaldi. 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 1 RR949-M, ARTEMIS-IMAG, Grenoble, France.

Baldacci, R., L. D. Bodin, A. Mingozzi. 2006. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Comput. Oper. Res.* **33**(9) 2667–2702.

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

Baldacci, R., E. Bartolini, A. Mingozzi, R. Roberti. 2010. An exact solution framework for a broad class of vehicle routing problems. *Comput. Management Sci.* **7**(3) 229–268.

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

Chabrier, A. 2006. Vehicle routing problem with elementary shortest path based column generation. *Comput. Oper. Res.* **33**(10) 2972–2990.

Christofides, N., A. Mingozzi, P. Toth. 1981. Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxation. *Math. Programming* **10**(1) 255–280.

CPLEX. 2009. IBM ILOG CPLEX 12.1 callable library. ILOG.

Danna, E., C. Le Pape. 2005. Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows. G. Desaulniers, J. Desrosiers, M. M. Solomon, eds. *Column Generation*. Springer, New York, 90–130.

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

Desrochers, M., J. Desrosiers, M. M. Solomon. 1992. A new optimization algorithm for the vehicle-routing problem with time windows. *Oper. Res.* **40**(2) 342–354.

Feillet, D., P. Dejax, M. Gendreau, C. Gueguen. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* **44**(3) 216–229.

Fukasawa, R., H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R. F. Werneck. 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Programming* **106**(3) 491–511.

Gendreau, M., A. Hertz, G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. *Management Sci.* **40**(10) 1276–1290.

Irnich, S., D. Villeneuve. 2006. The shortest-path problem with resource constraints and k-cycle elimination for $k \geqslant 3$. *INFORMS J. Comput.* **18**(3) 391–406.

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

Kohl, N., O. B. G. Madsen. 1997. An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation. *Oper. Res.* **45**(3) 395–406.

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

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

Mingozzi, A., N. Christofides, E. A. Hadjiconstantinou. 1994. An exact algorithm for the vehicle routing problem based on the set partitioning formulation. Technical report, University of Bologna, Bologna, Italy.

Pisinger, D., S. Ropke. 2007. A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **34**(8) 2403–2435.

Righini, G., M. Salani. 2008. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks* **51**(3) 155–170.

Ropke, S. 2005. Heuristic and exact algorithms for vehicle routing problems. Ph.D. thesis, Computer Science Department, University of Copenhagen (DIKU), Copenhagen.

Ropke, S. 2010. Private communication.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35**(2) 254–265.