

**Tabu Search, Generalized k -Path
Inequalities, and Partial
Elementarity for the Vehicle Routing
Problem with Time Windows**

G. Desaulniers, F. Lessard,
A. Hadjar

G-2006-45

July 2006

Les textes publiés dans la série des rapports de recherche HEC n'engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d'une subvention du Fonds québécois de la recherche sur la nature et les technologies.

Tabu Search, Generalized k -Path Inequalities, and Partial Elementarity for the Vehicle Routing Problem with Time Windows

Guy Desaulniers
François Lessard
Ahmed Hadjar

*GERAD and
Département de mathématiques et de génie industriel
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada H3C 3A7
{guy.desaulniers;francois.lessard, ahmed.hadjar}@gerad.ca*

July 2006

Les Cahiers du GERAD

G-2006-45

Copyright © 2006 GERAD

Abstract

The vehicle routing problem with time windows consists in delivering goods at minimum cost to a set of customers using an unlimited number of capacitated vehicles housed in a single depot. Each customer must be visited within a prescribed time window. The most recent successful solution approaches for this problem are branch-and-price-and-cut methods where the column generation subproblem is an elementary shortest path problem with resource constraints (ESPPRC). In this paper, we propose accelerating strategies for such a methodology. First, for solving the ESPPRC, we develop a tabu search heuristic that allows to rapidly generate negative reduced cost columns in most iterations. Second, we introduce a generalization of the k -path inequalities and highlight that these generalized inequalities can, in theory, be stronger than the traditional ones. Third, we propose a new subproblem type, the partially elementary shortest path problem with resource constraints (PESPPRC), which imposes elementarity requirements only on a subset of the nodes. The PESPPRC is easier to solve than the ESPPRC and can yield lower bounds that are comparable in quality with the ones provided by the ESPPRC. Finally, combining these ideas with the most recent advances published in the literature, we present a wide variety of computational results on the Solomon's 100-customer benchmark instances. In particular, we report solving five previously unsolved instances.

Key Words: Vehicle routing, time windows, column generation, partial elementarity, tabu search, generalized k -path inequalities.

Résumé

Le problème de tournées de véhicule avec fenêtres de temps consiste à minimiser les coûts totaux encourus pour livrer de la marchandise à un ensemble de clients en utilisant un nombre illimité de véhicules de capacité restreinte issus d'un même dépôt. Chaque client doit être visité dans une fenêtre de temps prédéterminée. Les plus récentes approches de résolution efficaces pour ce problème sont des méthodes de génération de colonnes et de plans coupants imbriquées dans une méthode d'énumération implicite dans lesquelles le sous-problème de génération de colonnes est un problème de plus court chemin élémentaire avec contraintes de ressource (ESPPRC). Dans cet article, nous proposons des stratégies d'accélération pour une telle méthode. Premièrement, pour résoudre le ESPPRC, nous développons une heuristique de type tabou qui permet de générer rapidement des colonnes de coût réduit négatif à la plupart des itérations. Deuxièmement, nous introduisons une généralisation des inégalités de type k -chemins et mettons en évidence que ces inégalités généralisées peuvent être, en théorie, plus fortes que les inégalités traditionnelles. Troisièmement, nous proposons un nouveau type de sous-problème, soit le problème de plus court chemin partiellement élémentaire avec contraintes de ressource (PESPPRC), qui impose des conditions d'élémentarité seulement sur un sous-ensemble des nœuds. Le PESPPRC est plus facile à résoudre que le ESPPRC et peut mener à des bornes inférieures semblables à celles fournies par le ESPPRC. Finalement, en combinant ces idées avec les avancements les plus récents publiés dans la littérature, nous présentons une grande variété de résultats numériques sur les instances de 100 clients de Solomon. En particulier, nous parvenons à résoudre cinq instances qui n'étaient pas encore résolues.

Mots clés : Tournées de véhicules, fenêtres de temps, génération de colonnes, élémentarité partielle, recherche tabou, inégalités de type k -chemins généralisées.

1 Introduction

The vehicle routing problem with time windows (VRPTW) is a generalization of the well-known capacitated vehicle routing problem (see Toth and Vigo, 2002). Given an unlimited number of vehicles housed in a single depot, it consists in determining a set of least-cost feasible routes to deliver goods to a set of customers. Each customer must be visited exactly once by a vehicle within a prescribed time interval. A route starts from the depot and visits in order a sequence of customers before returning to the depot. It is feasible if the total amount of goods delivered to the visited customers does not exceed the vehicle capacity and if it respects the time window of each visited customer. The cost of a route is given by the sum of the traveling costs between the consecutive locations visited along the route.

Several exact approaches for the VRPTW have been developed in the past fifteen years. Among them, branch-and-price approaches have produced the best results. A branch-and-price method consists of a branch-and-bound procedure where lower bounds are computed by column generation. The column generation technique separates the VRPTW linear relaxation into a master problem and a subproblem which corresponds to an elementary shortest path problem with resource constraints (ESPPRC). Because the ESPPRC is an NP -hard problem (see Dror, 1994), several researchers have replaced it by a relaxation which allows cycles, namely, the shortest path problem with resource constraints (SPPRC). Such an SPPRC-based decomposition method produces good results when the time windows are not too wide. However, when they are very wide, it provides weak lower bounds that are not tight enough to yield manageable branch-and-bound search trees.

For tackling VRPTW instances with wide time windows, two main research streams were recently explored. First, dynamic programming algorithms for solving the ESPPRC were developed and improved. These algorithms, which can be relatively efficient for some of the difficult instances, offer the possibility to model the subproblem as an ESPPRC in branch-and-price approaches, yielding tight lower bounds. On the other hand, they can be quite inefficient against very hard-to-solve ESPPRC subproblems which can occur almost at any iteration during the column generation process. Second, valid inequalities for the VRPTW were introduced to strengthen its formulation. These cutting planes have been used in branch-and-price approaches and, more recently, in branch-and-cut approaches that rely on a compact (non-decomposed) formulation of the problem.

In this paper, we propose new ideas in both research streams and integrate them into an exact branch-and-price-and-cut approach. First, to avoid as much as possible having to solve very difficult ESPPRC subproblems using dynamic programming, we introduce a simple tabu search algorithm that succeeds to generate negative reduced cost columns most of the time. Second, we develop a generalized version of the k -path valid inequalities introduced by Kohl et al. (1999) and show that they can dominate the traditional k -path inequalities. However, given the complexity of the separation algorithm, we only applied

the generalized 2-path inequalities in our tests. Third, we propose a new subproblem type, called the partially elementary shortest path problem with resource constraints, in which elementarity is only imposed for a subset of the nodes. This subset is generated dynamically during the solution process. This new subproblem type offers a compromise between the difficulty of solving the ESPPRC and the quality of the lower bounds.

As a final contribution, we present extensive computational results for the 100-customer VRPTW benchmark instances created by Solomon (1987). To the best of our knowledge, 10 of these 56 instances were not yet solved to optimality prior to this paper. Combining the proposed ideas with the most recent advances appearing in the literature, we succeeded to close 5 of these 10 unsolved instances and also to obtain much faster solution times for the most difficult instances when compared to those reported in the most recent papers.

Let us point out that the VRPTW has played and is still playing an important role in the development and improvement of solution methodologies for the class of deterministic time constrained vehicle routing and crew scheduling problems (see Desaulniers et al., 1998). Several research teams (see the literature review below) have been working actively on this problem in the past few years. This effervescence is certainly due to the fact that most of the 56 Solomon's 100-customer instances have now been solved to optimality and it seems reasonable to think that the unsolved ones will be solved in a near future. Our contribution is significant in this respect since we are closing half of the last 10 open instances.

The rest of this paper is structured as follows. A literature review on the VRPTW is presented in the next section. A mathematical model and an overview of the solution method are provided in Section 3. Then, in Section 4, we detail the various algorithms used to generate columns, including the tabu search method, and describe how they are used at each column generation iteration. In Section 5, we discuss branching and cutting strategies with an emphasis on the generalized k -path cuts. Next, we introduce the concept of partial elementarity in Section 6 before presenting extensive computational results in Section 7. The paper ends with a short conclusion.

2 Literature review

The VRPTW has been extensively studied in the literature. Recent surveys on this problem can be found in Cordeau et al. (2001) and Kallehauge et al. (2005). Here, we briefly review the papers published on exact solution methodologies. The reader interested by heuristics is referred to the recent survey papers by Bräysy and Gendreau (2005a, b).

2.1 Branch-and-price approaches based on the SPPRC

Desrochers et al. (1992) were the first to propose an exact solution approach capable of solving VRPTW instances of respectable size (up to 100 customers). Their branch-and-price approach was based on an SPPRC subproblem involving two resources: one for the

time and another for the vehicle load. To get better lower bounds, they applied when solving the subproblem a 2-cycle elimination procedure which disallows routes that visit the same customer i immediately before and immediately after visiting a customer j (that is, customers $i - j - i$ consecutively). Kohl and Madsen (1997) proposed a similar approach where column generation is replaced by Lagrangian relaxation to compute lower bounds. A two-phase method (subgradient first, bundle second) is applied for solving the Lagrangian dual problem.

An important contribution was made by Kohl et al. (1999) who introduced the k -path inequalities for the VRPTW. Given a set S of customers, such a valid inequality imposes that at least k vehicles be used to serve the customers in S if it can be proven that they cannot be served with only $k - 1$ vehicles. Thus, the separation algorithm consists in verifying the feasibility of a traveling salesman problem with time windows when $k = 2$ and of a relatively small VRPTW with a fixed number of vehicles (namely, $k - 1$) when $k \geq 3$. Given the complexity of the separation algorithm for $k \geq 3$, Kohl et al. (1999) only tested the use of 2-path cuts and obtained very good results with them. Using parallel processors and a different algorithm to enumerate the sets S , Cook and Rich (1999) showed that applying k -path inequalities with $k \geq 3$ can substantially improve lower bound values.

To improve the lower bound quality, Irnich and Villeneuve (2003) devised a k -cycle elimination procedure for the SPPRC for arbitrary integer values of k . For a given value of k , this procedure forbids the generation of routes that contain at least one l -cycle with $l \in \{2, 3, \dots, k\}$, where an l -cycle is a subpath starting and ending at the same node that contains l arcs. Irnich and Villeneuve (2003) tested this elimination procedure for $k \leq 5$ and showed that this procedure can considerably increase the lower bound values and reduce the overall solution times for difficult VRPTW instances.

2.2 Branch-and-price approaches based on the ESPPRC

Feillet et al. (2004) were the first to propose a branch-and-price approach in which the subproblem is modeled as an ESPPRC and solved by dynamic programming (see Desrochers, 1986, and Irnich and Desaulniers, 2005). They define one resource for each customer that indicates whether or not the customer has been visited along a path. To improve the efficiency of the dominance procedure used in the dynamic programming algorithm, they propose to set also at 1 in a label the resource variables associated with all unvisited customers that cannot be reached anymore from the current partial path, because of the time windows or the vehicle capacity. Other refinements for the dynamic programming algorithm have also been suggested by Feillet et al. (2005) and Chabrier (2006).

Righini and Salani (2004) proposed a bounded bi-directional search dynamic programming algorithm for solving the ESPPRC. Also, Righini and Salani (2005) and Boland et al. (2006) independently developed very similar efficient dynamic programming algorithms for the ESPPRC which start by solving the corresponding SPPRC. If the computed shortest

path is non-elementary, customer resource variables are added and a hybrid SPPRC-ESPPRC is solved. This process is repeated until finding an elementary shortest path. In a column generation context for the VRPTW, this iterative process can be stopped after solving a hybrid subproblem when either negative reduced cost elementary paths are found or the length of the shortest path is non-negative. Salani (2005) tested these improvements for solving the ESPPRC subproblems of a branch-and-price approach for the VRPTW and obtained very good results.

To obtain integer solutions throughout the solution process, Danna and Le Pape (2005) proposed a cooperation scheme between a branch-and-price method, a MIP heuristic solver, and a metaheuristic (for instance, large neighborhood search or guided tabu search). In this scheme, the branch-and-price algorithm is the leading method while the other two are invoked at regular time intervals to generate integer solutions. Cooperation occurs when the metaheuristic provides new interesting columns for the master problem (i.e., columns which can complement those in the master problem to build good feasible solutions) and when the branch-and-price approach or the MIP solver find new initial solutions for the metaheuristic.

Recently, Jepsen et al. (2006) showed how particular Chvatal-Gomory cuts can be used in a branch-and-price method for the VRPTW. These so-called subset row inequalities are very efficient for tightening the master problem linear relaxation of the VRPTW instances. However, they require additional resources in the subproblem to handle them. Further details on these inequalities are given in Section 5.2.

2.3 Pure branch-and-cut approaches

Pure branch-and-cut approaches for the VRPTW have also been developed in the last years by Bard et al. (2002), Lysgaard (2004), and Kallehauge and Boland (2005). These approaches are based on an arc-flow formulation of the problem and, typically, require for obtaining good lower bounds a large number of valid inequality families: subtour elimination, two-path, infeasible path, reachability, successor, predecessor, odd CAT, etc. It should be noted that several of these inequality families are implied when using a path-flow formulation solved by branch-and-price, especially when the subproblem is an ESPPRC.

3 Model and headlines of the solution approach

The VRPTW can be modeled as a set partitioning problem. Let \mathcal{N} be the set of customers and Ω the set of all feasible routes. With each route $p \in \Omega$, associate the following parameters: c_p represents its cost and v_{ip} , $\forall i \in \mathcal{N}$, takes value 1 if route p visits customer i and 0 otherwise. Furthermore, define a binary variable θ_p for each route $p \in \Omega$ which takes value 1 if route p is chosen in the solution, and 0 otherwise.

Using this notation, the VRPTW can be formulated as follows.

$$\text{Minimize} \quad \sum_{p \in \Omega} c_p \theta_p \quad (1)$$

$$\text{subject to:} \quad \sum_{p \in \Omega} v_{ip} \theta_p = 1, \quad \forall i \in \mathcal{N} \quad (2)$$

$$\theta_p \text{ binary}, \quad \forall p \in \Omega. \quad (3)$$

The objective function (1) aims at minimizing total cost. Set partitioning constraints(2) ensure that each customer is visited exactly once by a vehicle. Finally, binary requirements (3) are imposed on the variables.

In practice, model (1)–(3) contains a huge number of variables. Like several authors (see Section 2), we propose to solve it using a branch-and-price-and-cut approach. In such an approach (see Barnhart et al., 1998, Desaulniers et al., 1998), column generation is applied to compute lower bounds, cutting planes are generated to improve the quality of these bounds, and a branch-and-bound procedure is used, when needed, to derive integer solutions.

To compute a lower bound at a node of the search tree, we apply column generation for solving the linear relaxation of model (1)–(3) augmented by the appropriate branching decisions and cutting planes. Column generation (Dantzig and Wolfe, 1960; see also Desrosiers and Lübbecke, 2005) is an iterative process that solves alternately a restricted master problem and a subproblem. The restricted master problem is simply the linear relaxation of the augmented model restricted to a small subset of its variables. It is solved by the simplex algorithm to provide a primal and a dual solution. The subproblem corresponds to an ESPPRC which can typically be solved by dynamic programming (see Feillet et al., 2004, and Irnich and Desaulniers, 2005) to generate negative reduced cost columns (variables) that are added to the restricted master problem before beginning another iteration. The solution process stops when the subproblem cannot produce any negative reduced cost column.

The ESPPRC subproblem is defined over a network $G = (\mathcal{N} \cup \{o, d\}, \mathcal{A})$, where $\mathcal{N} \cup \{o, d\}$ and \mathcal{A} are the sets of nodes and arcs, respectively. Set $\mathcal{N} \cup \{o, d\}$ contains a node for each customer in \mathcal{N} , as well as a source node o and a sink node d representing the depot at the beginning and the end of the horizon, respectively. Set \mathcal{A} contains start arcs (o, j) , $j \in \mathcal{N}$, end arcs (i, d) , $i \in \mathcal{N}$, and travel arcs (i, j) , $i, j \in \mathcal{N}$, when customer j can be visited immediately after customer i in at least one feasible route. A cost c_{ij} is associated with each arc $(i, j) \in \mathcal{A}$.

Each feasible vehicle route can be represented by a path in G . Resource constraints (see Irnich and Desaulniers, 2005) are however required in the subproblem definition to

ensure the feasibility of the generated paths with regards to time windows, vehicle capacity, and elementarity. In its simplest form, a resource is a quantity that accumulates along a path and is restricted to a resource interval at each of its nodes. When the ESPPRC is a subproblem for the VRPTW, $|\mathcal{N}| + 2$ resources are needed: one for the time (denoted *time*), another for the vehicle load (denoted *load*), and one for each customer $n \in \mathcal{N}$ to ensure that a customer is not visited more than once (denoted *cust_n*). Let \mathcal{R} denote the set of these resources, t_{ij}^r the consumption of resource $r \in \mathcal{R}$ along arc $(i, j) \in \mathcal{A}$, and $[a_i^r, b_i^r]$ the resource interval at node $i \in \mathcal{N}$ for resource $r \in \mathcal{R}$. For $r = \textit{time}$, t_{ij}^r is equal to the travel time between locations i and j plus the service time at i if $i \in \mathcal{N}$, while $[a_i^r, b_i^r]$ corresponds to the customer time window for $i \in \mathcal{N}$ and to the planning horizon for $i \in \{o, d\}$. For $r = \textit{load}$, t_{ij}^r is equal to the load l_j picked up at j if $j \in \mathcal{N}$ and 0 otherwise, while $[a_i^r, b_i^r] = [0, Q]$ for all nodes $i \in \mathcal{N} \cup \{o, d\}$. Finally, for $r = \textit{cust}_n$, $n \in \mathcal{N}$, t_{ij}^r is equal to 1 if $j = n$ and 0 otherwise, while $[a_i^r, b_i^r] = [0, 1]$ for all nodes $i \in \mathcal{N} \cup \{o, d\}$. As it is often the case in practice, we assume that all resource consumptions satisfy the triangle inequality, that is, for all pairs of arcs (i, j) and (j, k) in \mathcal{A} , arc (i, k) exists and $t_{ij}^r + t_{jk}^r \geq t_{ik}^r$ for all resources $r \in \mathcal{R}$.

Let X_{ij} be a binary variable indicating whether or not arc $(i, j) \in \mathcal{A}$ is part of the computed shortest path for the ESPPRC and T_i^r be the value of resource $r \in \mathcal{R}$ accumulated along this path from the source node o up to node $i \in \mathcal{N} \cup \{o, d\}$. To compute the feasible path with the least reduced cost, we also need the dual variable values π_i of constraints (2) associated with the restricted master problem solution of the current iteration.

Given that notation, the ESPPRC subproblem can be formulated as follows:

$$\text{Minimize} \quad \sum_{j \in \mathcal{N}} c_{oj} X_{oj} + \sum_{(i,j) \in \mathcal{A}: i \in \mathcal{N}} (c_{ij} - \pi_i) X_{ij} \quad (4)$$

$$\text{subject to:} \quad \sum_{j \in \mathcal{N}} X_{oj} = 1, \quad (5)$$

$$\sum_{j: (i,j) \in \mathcal{A}} X_{ij} - \sum_{j: (j,i) \in \mathcal{A}} X_{ji} = 0, \quad \forall i \in \mathcal{N} \quad (6)$$

$$X_{ij}(T_i^r + t_{ij}^r - T_j^r) \leq 0, \quad \forall (i, j) \in \mathcal{A}, r \in \mathcal{R} \quad (7)$$

$$a_i^r \leq T_i^r \leq b_i^r, \quad \forall i \in \mathcal{N} \cup \{o, d\}, r \in \mathcal{R} \quad (8)$$

$$X_{ij} \text{ binary}, \quad \forall (i, j) \in \mathcal{A}. \quad (9)$$

The objective function (4) aims at minimizing the path reduced cost. Constraints (5)–(6) define a path structure, while constraints (7)–(8) ensure the feasibility of this path through the use of resource constraints. Finally, binary requirements are provided by (9).

Note that branching decisions and cutting planes can modify the ESPPRC subproblem in different ways. For instance, they may reduce the arc set \mathcal{A} (when branching on arcs) or change the variable cost coefficients through additional dual variables (when cutting planes are added as constraints in the restricted master problem). See Desaulniers et al. (1998) for further details.

In the next section, we present the multiple-algorithm procedure used to generate columns at each iteration of the branch-and-price-and-cut approach. Branching and cutting strategies are discussed afterwards in Section 5.

4 Column generators

As mentioned in the introduction, the ESPPRC subproblem is an *NP*-hard problem. Some of its instances can be very hard to solve (more than 10 hours of computational time for certain instances involving 100 customers). Thus, it is common practice to try to avoid solving the subproblem to optimality at each column generation iteration as long as it is possible to generate negative reduced cost columns. On the other hand, to prove the optimality of the current restricted master problem solution and obtain a lower bound for the current branch-and-bound node, the subproblem must be solved to optimality at least in the last iteration. A possible alternative to obtain a lower bound, which avoids solving to optimality the subproblem, consists in computing the so-called Lagrangean bound (see Desaulniers et al., 1998) using a relaxation of the ESPPRC subproblem (for instance, using the SPPRC subproblem instead). Since this bound is, in general, weaker than the bound obtained by completing the column generation process, we compute (when needed) the latter bound in our approach.

In this section, we begin by briefly describing the dynamic programming algorithm used to solve the ESPPRC subproblem to optimality. Next, we discuss how this algorithm can be speeded up by heuristic strategies. Then, we present a fast tabu search method for the ESPPRC that can generate quite efficiently negative reduced cost columns. Finally, we provide the details of the procedure that combines these algorithms for generating columns.

4.1 Exact dynamic programming

Exact dynamic programming algorithms can be used to solve the ESPPRC. In our approach, we use an implementation of the label setting algorithm developed by Righini and Salani (2005). Here, we summarize the basic features of this algorithm.

Typical label setting algorithms rely on three main concepts: label, label extension, and label dominance. A label represents a feasible partial path from the source node o to any node in $\mathcal{N} \cup \{o, d\}$. It is associated with the ending node of the partial path. For the ESPPRC subproblem of the VRPTW, a label has $|\mathcal{N}| + 3$ components, one for its cost, and one for each resource in \mathcal{R} . A label L_i at node i can thus be denoted by

$L_i = (Z_i, T_i^{time}, T_i^{load}, T_i^{cust_1}, \dots, T_i^{cust_{|\mathcal{N}|}})$, where Z_i is the cost of the partial path. The solution process begins with a single label $L_o = (0, a_o^{time}, 0, 0, \dots, 0)$ at node o . Labels are then extended along the arcs of \mathcal{A} to yield new feasible partial paths. The extension of a label L_i along an arc (i, j) consists in computing a new label L_j associated with node j that represents the partial path obtained by appending arc (i, j) to the partial path represented by L_i . In our case, label $L_j = (Z_j, T_j^{time}, T_j^{load}, T_j^{cust_1}, \dots, T_j^{cust_{|\mathcal{N}|}})$ is computed as follows:

$$\begin{aligned} Z_j &= Z_i + c_{ij} \\ T_j^r &= \max\{a_j^r, T_i^r + t_{ij}^r\}, \quad \forall r \in \mathcal{R}. \end{aligned}$$

When such an extension does not produce a feasible partial path, that is, $T_j^r > b_j^r$ for at least one resource $r \in \mathcal{R}$, label L_j is not created.

To avoid enumerating all feasible partial paths from node o to all other nodes in G , a label dominance procedure is applied during the solution process on the generated labels associated with the same node. It identifies among them the Pareto-optimal labels and discards all the others. A label is Pareto-optimal if it is not dominated by any other label. A label $L_1 = (Z_1, T_1^{time}, T_1^{load}, T_1^{cust_1}, \dots, T_1^{cust_{|\mathcal{N}|}})$ is dominated by a label $L_2 = (Z_2, T_2^{time}, T_2^{load}, T_2^{cust_1}, \dots, T_2^{cust_{|\mathcal{N}|}})$ if

$$Z_2 \leq Z_1 \tag{10}$$

$$T_2^r \leq T_1^r, \quad \forall r \in \mathcal{R} \tag{11}$$

and at least one of these inequalities is strict.

As proposed by Feillet et al. (2004), a stronger dominance rule can be obtained by redefining the customer resources as follows. In a label L , the binary resource corresponding to customer $n \in \mathcal{N}$ takes value 1 if this customer has already been visited along the partial path represented by L or if it cannot be reached from this path while maintaining feasibility. This new definition allows to discard more labels and speeds up the solution process.

To further reduce the number of generated labels, Righini and Salani (2005) developed a bounded bidirectional search algorithm that proceeds in three steps: forward labeling, backward labeling, label junction. Let $H = b_d^{time} - a_o^{time}$ be the horizon length. The forward labeling step is a label setting algorithm that extends the labels as described above, starting from the initial label associated with node o . A label is however not extended when its time component T^{time} is greater than or equal to $H/2$. The backward labeling step is symmetric to the forward labeling step and starts from an initial label associated with node d . Finally, in the third step, pairs of forward and backward labels associated with the same node are joined together to derive complete o - d paths when these junctions yield negative reduced cost feasible paths.

Another efficient accelerating strategy was proposed independently by Righini and Salani (2005) and Boland et al. (2006). This iterative strategy, called decremental search

space by the first authors, begins by solving the corresponding SPPRC, that is, without any customer resources. If the computed shortest path is non-elementary, the resource associated with the most visited customer in this path is added to the problem, yielding a hybrid SPPRC-ESPPRC. The next iteration consists in solving this modified problem and analyzing its solution. The process iterates in this way until the computed shortest path is elementary. In a column generation context, this iterative process can be stopped as soon as a negative reduced cost elementary path is found (even if it is not a shortest path) or when the computed shortest path has a non-negative reduced cost. In our approach, instead of starting decremental search with an empty set of customer resources at each column generation iteration, we start it using the customer resources of the preceding iteration. According to the results of preliminary tests, this seems to be much more efficient because there is an important intersection between the sets of customer resources generated at different column generation iterations when an empty initial set is used. Finally, in our implementation, 2-cycles are also eliminated when solving the hybrid SPPRC-ESPPRC subproblem using the procedure described in Irnich and Villeneuve (2003).

4.2 Heuristic dynamic programming

To speed up the exact dynamic programming algorithm, we use two different heuristic strategies that were also used by several other researchers. The first strategy consists in eliminating a priori a certain number of arcs that do not seem promising. This elimination procedure depends on the current dual variable values and is, therefore, applied at every column generation iteration when heuristic dynamic programming is used. First, for each node, it ranks separately its incoming arcs and its outgoing arcs in increasing order of their cost ($c_{ij} - \pi_i$ if $i \neq o$, c_{oj} otherwise). Then, given a minimum number A_{min} of incoming and outgoing arcs to keep for each node in \mathcal{N} , it eliminates every arc $(i, j) \in \mathcal{A}$ such that $i \neq o$, $j \neq d$, and the rank of (i, j) is greater than A_{min} for the incoming arcs of j and also for the outgoing arcs of i . Therefore, we keep all start and end arcs, and at least A_{min} incoming and A_{min} outgoing arcs for each node in \mathcal{N} .

In the second heuristic strategy, an aggressive dominance rule is applied to eliminate a larger number of labels in the dominance procedure. Given a maximum number R_{max} of customer resources to use in the dominance test, we simply apply the dominance test stated in Subsection 4.1 on a restricted subset of R_{max} customer resources. These R_{max} customers are selected at each iteration as follows. First, we compute for each customer covering constraint (2) the difference between the value of the corresponding dual variable at the current iteration and that at the preceding iteration. Then, we select the R_{max} customers associated with the R_{max} highest differences.

4.3 Tabu search

Another heuristic alternative to rapidly generate negative reduced cost columns is to use tabu search for solving the ESPPRC. Tabu search (see Glover and Laguna, 1997) is a well-known metaheuristic that has been successful at solving difficult combinatorial optimization problems, namely the VRPTW (see Bräysy and Gendreau, 2005b). It starts with an initial solution and applies a sequence of moves to modify it. Allowable moves are defined by a set of relatively simple operators. For instance, an operator can correspond to the reassignment of a customer i to a different route p . The various choices of i and p define different moves for this operator. A solution that can be obtained from a given solution using a single allowable move is called a neighbor. A tabu search method is iterative and computes at each iteration the best neighbor of the current solution. It then replaces the current solution by this neighbor even if the objective value is deteriorated. To avoid cycling, it uses a tabu list that contains the inverse moves of the latest moves performed. These inverse moves are simply forbidden for a certain number of iterations. The tabu list thus allows to escape local minima. Tabu search can also benefit greatly from a diversification strategy which is invoked occasionally to move the search to a different region of the solution space. Various diversification strategies can be applied.

The tabu search method that we propose for the ESPPRC relies on two operators: inserting a customer into the current path (solution) and deleting a customer from the current path. To simplify computations, the solution space is restricted to the set of feasible solutions. Therefore, at each iteration of the method, given the subset $\tilde{\mathcal{N}}$ of customers in the current path p , we evaluate all non-tabu feasible neighbors of p . To generate these neighbors, we first scan the list of customers in $\tilde{\mathcal{N}}$ and delete them individually from p . Then, we scan the list of customers in $\mathcal{N} \setminus \tilde{\mathcal{N}}$ and try to insert individually each of them into the path at every possible position. For each neighbor derived from the insertion operator, path feasibility is checked for path elementarity, customer time windows, and vehicle capacity. No feasibility checks are needed for the deletion operator.

To diversify the search, we use a multiple start procedure, that is, we start the tabu search method with different initial solutions and limit to a maximum number I_{max} the number of iterations to perform from each initial solution. The set of initial solutions is given by the paths associated with the basic variables of the current restricted master problem solution. All these paths are good initial candidates because they have a zero reduced cost. In a column generation context, all negative reduced cost best neighbors are retained to be added to the restricted master problem. Also, the algorithm is halted either when all initial solutions have been used or when a maximum number C_{max} of negative reduced cost columns have been found.

Minor modifications to the tabu search method are needed to handle some of the features discussed in the following sections. First, when branching on the arc flow variables (see Section 5.3), sequences of customers become imposed. In this case, these sequences are seen

by the tabu search operators as aggregated customers that cannot be visited separately. Second, when a 3-customer subset row cut is added in the restricted master problem (see Section 5.2), its corresponding dual variable value must be subtracted from the reduced cost of a path when it contains at least two of these three customers. This can easily be handled by keeping track of the number of customers included in the path that are part of these 3-customer subsets. Finally, when the elementarity requirements is relaxed for certain customers (see Section 6), the tabu search feasibility check for elementarity is modified to allow cycles with these customers (except 2-cycles).

4.4 Column generators usage

The column generators described in the previous subsections have different characteristics. For hard-to-solve VRPTW instances, the exact dynamic programming algorithm is very slow but efficient in the sense that it always generates negative reduced cost columns when some exist. Because of its speed, it should be avoided as much as possible. Various heuristic dynamic programming algorithms can be devised by setting the parameters A_{min} and R_{max} to different values. Smaller values generally yield a fast but less efficient algorithm, while larger values can considerably increase solution times and efficiency. It should be noted that these algorithms are negatively affected by large (unstable) dual variable values because such values favor cycles. Since these values usually appear in approximately the first half of the column generation iterations when solving a linear relaxation (see du Merle et al., 1999), the use of heuristic dynamic programming algorithms in those iterations should be limited. Finally, the computational time spent by the tabu search method can be controlled by the value of I_{max} . Therefore, it can be very fast to very slow. Its efficiency depends on the available computational time. On the one hand, it is very efficient in the first few column generation iterations when there exist plenty of negative reduced cost columns even when I_{max} takes a small value. On the other hand, it might fail to find negative reduced cost columns when there are very few such columns in the last column generation iterations.

Given these characteristics, we propose the following usage of the different column generators. First, we divide the column generation process into a predefined number of phases (typically, two or three phases). Intuitively, the first phase should include the iterations with unstable dual values, while the last phase those with very few negative reduced cost columns. Middle phases, if any, correspond to transitional phases. A phase is defined by an ordered sequence of algorithms (for instance, two or three) and parameters associated with them. The sequence may contain the same algorithm more than once, but with different parameter values. In a sequence, the algorithms are ordered from the fastest to the slowest. In any phase except the first, the sequence contains at least one algorithm that is more efficient than those of the preceding phase. For instance, in a two-phase approach, the first phase may involve a tabu search algorithm with $I_{max} = 30$ followed by a heuristic dynamic programming algorithm with $A_{min} = 10$ and $R_{max} = 5$,

while the second phase may comprise a tabu search algorithm with $I_{max} = 100$, a heuristic dynamic programming algorithm with $A_{min} = 15$ and $R_{max} = 10$, and the exact dynamic programming algorithm.

Then at each column generation iteration of a phase, the algorithms are applied one after the other (respecting their order and always starting from the first) to the subproblem until one of the algorithm generates at least one negative reduced cost column. When none of the algorithms succeed to generate a column, it indicates that the current algorithms lost their efficiency and we move on to the next phase. When all phases have been completed, the current restricted master problem solution is optimal for the linear relaxation bearing that the last phase algorithm sequence contains the exact dynamic programming algorithm. As it will be shown in the computational results, this type of approach permits to avoid in most iterations solving the *NP*-hard ESPPRC subproblem with the exact dynamic programming algorithm, yielding a much faster solution process for the hardest instances.

5 Cutting and branching strategies

This section describes the cutting and branching strategies used to derive integer solutions in the proposed branch-and-price-and-cut approach. We also discuss a stopping criterion which forces a premature termination of the linear relaxation solution process in order to avoid solving difficult ESPPRC subproblems.

5.1 Generalized k -path inequalities

The k -path inequalities for the VRPTW were introduced by Kohl et al. (1999). Given a set $S \subseteq \mathcal{N}$ of customers and the minimum number $k(S)$ of vehicles needed to service them, the corresponding k -path inequality is given by

$$X(S) = \sum_{(i,j) \in \delta^-(S)} X_{ij} \geq k(S), \quad (12)$$

where $X(S)$ is the total flow entering S , $\delta^-(S) \subset \mathcal{A}$ is the subset of arcs (i, j) entering S (that is, $i \notin S$ and $j \in S$) and

$$X_{ij} = \sum_{p \in \Omega} w_{ijp} \theta_p \quad (13)$$

is the total flow on arc $(i, j) \in \mathcal{A}$. In this expression, w_{ijp} is a binary parameter indicating whether or not arc (i, j) is part of path $p \in \Omega$ and X_{ij} can only take value 0 or 1. The separation algorithm consists in generating sets S and, for each of them, computing $k(S)$ by solving a VRPTW where only the customers in S need to be serviced and the objective solely aims at minimizing the number of vehicles used. Most researchers have focussed on the 2-path inequalities because they are easier to identify. Indeed, given a set $S \subseteq \mathcal{N}$, to determine if $k(S) \geq 2$, one only needs to verify if the capacity of a single vehicle is

sufficient for servicing all customers in S and, if so, to solve a traveling salesman problem with time windows (TSPTW). When capacity is insufficient or the TSPTW is infeasible, we can conclude that $k(S) \geq 2$. In this case, a cut with a right-hand side of 2 (even if $k(S) \geq 3$) can be generated when $X(S)$ takes a value less than 2 in the current linear relaxation solution. For the rest of this paper, we will call such a cut a *traditional k -path cut*.

We propose to generalize these k -path inequalities to obtain what we call *generalized k -path inequalities*. Let $S \subseteq \mathcal{N}$ and consider the partition $\delta_1^-(S) \cup \dots \cup \delta_{|S|}^-(S)$ of $\delta^-(S)$ such that an arc (i, j) is in $\delta_\ell^-(S)$ if ℓ is the minimum number of vehicles required to service all customers in S when (i, j) is used by one vehicle. Note that such a partition can be obtained by solving, for each arc $(i, j) \in \delta^-(S)$, a VRPTW in which a vehicle must use (i, j) . For a given $k \in \{1, 2, \dots, |S|\}$, let $\Delta_k^-(S) = \delta_k^-(S) \cup \dots \cup \delta_{|S|}^-(S)$. The generalized k -path inequalities are given by

$$\sum_{\ell=1}^{k-1} \frac{1}{\ell} \sum_{(i,j) \in \delta_\ell^-(S)} X_{ij} + \frac{1}{k} \sum_{(i,j) \in \Delta_k^-(S)} X_{ij} \geq 1, \quad \forall S \subseteq \mathcal{N}. \quad (14)$$

Proposition 1 *The generalized k -path inequalities (14) are valid for the VRPTW.*

Proof: Let us consider an arbitrary feasible solution $X = \bar{x}$ of the VRPTW and a subset of customers $S \subseteq \mathcal{N}$. Let $\delta_{\bar{m}}^-(S)$ denote the subset of arcs $(i, j) \in \delta^-(S)$ such that $\bar{x}_{ij} = 1$ and \bar{m} its cardinality. Since \bar{x} is a feasible solution that requires at most \bar{m} vehicles to service all customers in S , we deduce that $\delta_\ell^-(S) \cap \delta_{\bar{m}}^-(S) = \emptyset, \forall \ell > \bar{m}$. Thus, $\bar{x}_{ij} = 0$ for all $(i, j) \in \delta_\ell^-(S)$ such that $\ell > \bar{m}$, and

$$\sum_{\ell=1}^{k-1} \frac{1}{\ell} \sum_{(i,j) \in \delta_\ell^-(S)} \bar{x}_{ij} + \frac{1}{k} \sum_{(i,j) \in \Delta_k^-(S)} \bar{x}_{ij} \geq \frac{1}{\bar{m}} \sum_{\ell=1}^{\bar{m}} \sum_{(i,j) \in \delta_\ell^-(S)} \bar{x}_{ij} = \frac{\bar{m}}{\bar{m}} = 1. \quad \square$$

Let us make three remarks about these generalized inequalities. First, for a given set $S \subseteq \mathcal{N}$, when $\Delta_k^-(S) = \delta^-(S)$, the generalized k -path inequality (14) is equivalent to the traditional k -path inequality (12). Second, a traditional k -path inequality is strictly dominated by the corresponding generalized k' -path inequality whenever there exists an integer $k' > k$ such that $\Delta_{k'}^-(S) \neq \emptyset$. Finally, it may happen that, for a given k , a fractional-valued solution satisfies all traditional k -path inequalities while it violates some generalized k -path inequalities.

Note that k -path inequalities symmetric to (14) can also be defined by replacing $\delta_\ell^-(S)$ and $\Delta_k^-(S)$ in (14) with the corresponding sets $\delta_\ell^+(S)$ and $\Delta_k^+(S)$ of arcs exiting S . The inequalities (14) (resp. the symmetric ones) will thus be referred to as the generalized entering (resp. exiting) k -path inequalities. Contrarily to the traditional k -path inequalities,

the generalized entering and exiting k -path inequalities associated with a set S are not necessarily equivalent.

Given the difficulty of computing the partition of $\delta^-(S)$, we restricted our attention to the generalized entering 2-path inequalities (and their exiting counterparts) and used in our tests the following weaker version of them:

$$\sum_{(i,j) \in \delta^-(S) \setminus \tilde{\Delta}_2^-(S)} X_{ij} + \frac{1}{2} \sum_{(i,j) \in \tilde{\Delta}_2^-(S)} X_{ij} \geq 1, \quad \forall S \subseteq \mathcal{N}, \quad (15)$$

where $\tilde{\Delta}_2^-(S)$ is the subset of entering arcs which have been identified as belonging to $\Delta_2^-(S)$. To identify if arc (i, j) belongs to $\Delta_2^-(S)$, one can solve an ESPPRC (using the exact dynamic programming algorithm described in Section 4.1) defined over the subnetwork of G containing all arcs of \mathcal{A} except those in $\delta^-(S) \setminus \{(i, j)\}$. All arcs with a head node in S have a cost of -1, while all the others bear a cost of 0. With this cost structure, an optimal value greater than $-|S|$ for the corresponding ESPPRC means that all customers in S cannot be serviced by a single vehicle if it enters S using arc (i, j) . In this case, (i, j) belongs to $\Delta_2^-(S)$ and can be put in $\tilde{\Delta}_2^-(S)$. The treatment of each individual entering arc in $\delta^-(S)$ can be time consuming. An alternative is to group entering arcs and treat them simultaneously by solving an ESPPRC containing all these arcs. In this case, if the cost of the shortest path exceeds $-|S|$, then all these arcs belong to $\Delta_2^-(S)$ and can be put in $\tilde{\Delta}_2^-(S)$.

Let us denote by x_{ij} and $x(S)$ ($S \subseteq \mathcal{N}$) the values taken in the current fractional-valued solution by X_{ij} and $X(S)$, respectively. The separation algorithm that we implemented is as follows. Starting from the empty set, we generate with a recursive procedure the customer sets by adding to the current set S an additional customer i which is linked to S by an arc belonging to the master problem solution supporting graph (i.e., there exists an arc (i', j') such that $x_{i',j'} > 0$ and either $i' = i$ and $j' \in S$ or $j' = i$ and $i' \in S$). Such a customer can be added only if $|S| < S_{max}$, a predefined parameter, and if $x(S \cup \{i\}) < 2$. For each generated set S with $x(S) < 2$, we solve the following ESPPRCs. Let $\delta_P^-(S)$ (resp. $\delta_P^+(S)$) be the subset of arcs (i, j) in $\delta^-(S)$ (resp. $\delta^+(S)$) such that $x_{ij} > 0$. The first ESPPRC solved involves only the entering arcs in $\delta_P^-(S)$ and the exiting arcs in $\delta_P^+(S)$, the second the arcs in $\delta_P^-(S)$ and in $\delta^+(S) \setminus \delta_P^+(S)$, the third the arcs in $\delta^-(S) \setminus \delta_P^-(S)$ and in $\delta_P^+(S)$, and the fourth the arcs in $\delta^-(S) \setminus \delta_P^-(S)$ and in $\delta^+(S) \setminus \delta_P^+(S)$. When the optimal value of any of these ESPPRCs is greater than $-|S|$, we add the entering arcs to $\tilde{\Delta}_2^-(S)$ (and the exiting arcs to the corresponding exiting set). Otherwise, we can retrieve some arcs which definitely belong to $\delta_1^-(S)$ (or $\delta_1^+(S)$) and verify, with this updated information, if the corresponding generalized k -path inequalities are already satisfied. If this is the case, we stop solving the sequence of ESPPRCs and move on to the next customer set S . The advantage of using this procedure is that we can often conclude that no generalized k -path inequalities are violated for set S just after solving the first ESPPRC, which can be solved quite rapidly because of the small size of the sets $\delta_P^-(S)$ and $\delta_P^+(S)$.

When none of these four ESPPRCs has an optimal value of $-|S|$, then we can conclude that $\delta_1^-(S)$ is empty and a traditional 2-path cut (12) has been found. Otherwise, we solve an ESPPRC for each arc in $\delta_P^-(S)$ which have not yet been identified as belonging to $\delta_1^-(S)$ (and, similarly, for the exiting arcs) to identify additional arcs to put in $\tilde{\Delta}_2^-(S)$. Again, after identifying an arc that belongs to $\delta_1^-(S)$, the corresponding inequality is checked to determine if it is already satisfied. When all arcs in $\delta_P^-(S)$ have been classified either in $\delta_1^-(S)$ or in $\tilde{\Delta}_2^-(S)$ and the inequality is still not satisfied, then a generalized k -path cut (15) has been found because all unclassified arcs have a null flow. By default, all these unclassified arcs are put in $\delta^-(S) \setminus \tilde{\Delta}_2^-(S)$. However, to further strengthen the definition of the violated inequality, we can also solve additional ESPPRCs, involving groups of arcs in $\delta^-(S) \setminus \delta_P^-(S)$. When the optimal value of such an ESPPRC exceeds $-|S|$, then all arcs in the corresponding group are added to $\tilde{\Delta}_2^-(S)$. The number of such ESPPRCs to solve depends on a parameter U_{max} and the number of arcs in $\delta^-(S) \setminus \delta_P^-(S)$. These arcs are first sorted in decreasing order of their earliest arrival time in the set S . Then, respecting this order, they are divided into U_{max} groups of equal size if U_{max} is greater than the number of arcs. Otherwise, each group contains a single arc. A smaller value for U_{max} yields a faster separation algorithm, but produces weaker cuts because $\tilde{\Delta}_2^-(S)$ contains less arcs.

5.2 Subset row inequalities

Inspired by the clique and odd hole inequalities for the set packing problem, the subset row inequalities of Jepsen et al. (2006) are Chvatal-Gomory inequalities given by

$$\sum_{p \in \Omega} \lfloor \frac{1}{k} \sum_{i \in S} v_{ip} \rfloor \theta_p \leq \lfloor \frac{|S|}{k} \rfloor, \quad \forall S \subseteq \mathcal{N}, 2 \leq k \leq |S|, \quad (16)$$

where S is a subset of customers. Like in Jepsen et al. (2006), we focus on the cuts defined for subsets of three customers (or, more precisely, the rows (2) associated with them) because they are easy to find. These cuts can be rewritten as follows:

$$\sum_{p \in \Omega_S} \theta_p \leq 1, \quad \forall S \subseteq \mathcal{N} \text{ such that } |S| = 3, \quad (17)$$

where $\Omega_S \subseteq \Omega$ is the subset of paths visiting at least two customers in S .

In a column generation approach, the addition of a set Q of subset row cuts to the restricted master problem requires the following adjustments to the subproblem. The dual variable μ_q of such a cut $q \in Q$ defined for the 3-customer subset S_q must be subtracted from a path reduced cost when this path visits at least two customers in S_q . This subtraction can easily be incorporated in all the column generators described in Section 4 by computing the number of customers in S_q visited along a path. In dynamic programming algorithms, this number is computed using an additional resource src_q . Hence, a large number of subset row cuts considerably increases label dimension and can substantially impede the

efficiency of these algorithms. To alleviate this difficulty, Jepsen et al. (2006) proposed to modify the dominance rule (see Section 4.1) by omitting the resources src_q , $q \in Q$, in inequalities (11) and replacing inequality (10) by

$$Z_2 - \sum_{q \in Q_{1,2}} \mu_q \leq Z_1 \quad (18)$$

where $Q_{1,2} \subseteq Q$ is the subset of subset row cuts q for which T^{src_q} is equal to 1 in label L_2 and to 0 in label L_1 . With this modified rule, more labels can be dominated.

Even with this modification, dynamic programming algorithms can be quite slow when too many subset row cuts are present in the restricted master problem. To reduce this negative impact, we use the following three rules when generating subset row cuts. First, a maximum of Q_{max} cuts can be added simultaneously (the most violated ones are selected). Second, a customer can be included in at most N_{max} of the subsets defining these cuts. Third, cuts are added only if the most violated one is violated by at least V_{min} . Furthermore, when using a heuristic dynamic programming algorithm, the set $Q_{1,2}$ in (18) is replaced by the set $Q_{1,2} \cap \tilde{Q}$, where $\tilde{Q} \subseteq Q$ is updated at each column generation iteration and contains a maximum of M_{max} cuts, namely, those with the most negative dual variables. Like in Jepsen et al. (2006), the separation algorithm that we use consists in enumerating all 3-customer subsets and checking if the corresponding inequalities are violated.

5.3 Branching strategy

When no cuts can be added to strengthen a linear relaxation and branching is required, we branch on a fractional-valued variable X_{ij} (as defined by (13)). On one branch, this variable is set to 1 by removing from network G all arcs $(i, j') \in \mathcal{A}$ such that $j' \neq j$ if $i \neq o$, and all arcs $(i', j) \in \mathcal{A}$ such that $i' \neq i$ if $j \neq d$. On the other branch, it is set to 0 by removing from G the arc (i, j) . Columns present in the restricted master problem that do not respect the imposed decision are also removed prior to solving the modified linear relaxation. The branching variable X_{ij} is selected as the one yielding the highest value of the expression $c_{ij}(\min\{x_{ij}, 1 - x_{ij}\} + \sum_{(i', j') \in B_{ij}} x_{i'j'})$, where $B_{ij} \subset \mathcal{A}$ is the subset of arcs

incompatible with (i, j) and adjacent to it (that is, either $j' = i$ or $i' = j$). An arc (i', j') with $j' = i$ (resp. $i' = j$) is said to be incompatible with arc (i, j) if $(i', j') = (j, i)$ or the subpath $i' - j' - j$ (resp. $i - i' - j'$) is infeasible because of the time windows or vehicle capacity. The branch-and-bound tree is explored using a best-first strategy.

5.4 Early cutting and branching

Each time that cutting planes are added to a linear relaxation, the column generation process should be restarted and completed to find an optimal solution to the modified

linear relaxation and its corresponding lower bound. Because of the well-known tailing-off effect of the column generation process (the objective value does not change much in the last iterations), these reoptimizations can require a large amount of computational time, especially when difficult subproblems must be solved. To accelerate these reoptimizations, we propose to prematurely halt column generation at an iteration of the last phase of the multi-phase approach (see Section 4.4) when the following two conditions hold: i) the current linear relaxation was obtained by adding cutting planes; and ii) the current objective value is smaller than the value of the best integer solution found so far. In this case, to preserve the exactness of the branch-and-bound scheme, the bound associated with a node is the one computed before adding any cutting planes. The first condition ensures that this bound is valid. The second condition allows to compute a valid lower bound after adding cuts when there is a chance to prune the current node.

6 Partial elementarity

As mentioned in the literature review, different column generation subproblem types have been proposed for the VRPTW, namely, the SPPRC, the SPPRC with k -cycle elimination, and the ESPPRC. On the one hand, the SPPRC and ESPPRC subproblems provide the weak and strong lower bounds, respectively, while the SPPRC with k -cycle elimination yields intermediate bounds that improve with the value of k . On the other hand, the computational time required for solving one subproblem type generally increases with the quality of the bounds it produces. Because the ESPPRC can be very difficult to solve in some cases, we were not even able to solve in a reasonable computational time the first linear relaxation of a few VRPTW instances. Hence, solving these instances require either improving the solution methodology for solving the ESPPRC or using a weaker subproblem. Our efforts for improving the solution methodology described in Section 4.1 were not successful. Therefore, we turned our attention to the second option and asked ourselves the following question: can bounds of the same quality as those yielded by the ESPPRC be computed if elementarity requirements are imposed only on a subset of the nodes?

To answer this question, we introduce a new subproblem type that we call the partially elementary shortest path problem with resource constraints (PESPPRC). The PESPPRC is simply a hybrid SPPRC-ESPPRC similar to the ones encountered in the decremental search space strategy proposed for the ESPPRC (see Section 4.1), that is, elementarity is required only for a subset of the nodes. To solve the PESPPRC, one can easily adapt all the algorithms described in Section 4. Note also that the PESPPRC with k -cycle elimination can also be considered. In fact, we used the PESPPRC with 2-cycle elimination for our tests.

To use the PESPPRC as a subproblem, customers subject to elementarity requirements must be defined. Like in Righini and Salani (2005) and Boland et al. (2006), we propose to identify these customers dynamically. However, instead of identifying them at each column generation iteration, they are determined after solving each linear relaxation unless

Algorithm 1 Procedure for identifying customers with elementarity requirements

- 1: Let \mathcal{E}_0 be the set of customers previously identified as elementary
 - 2: $\mathcal{E} \leftarrow \emptyset$
 - 3: **for all** positive-valued variables θ_p in the current master problem solution **do**
 - 4: **if** $|\mathcal{E}_0| + |\mathcal{E}| < E_{max}$ **then**
 - 5: Find \mathcal{E}_p the set of customers visited more than once in path p and e_p the first of these customers encountered along p
 - 6: **if** $\mathcal{E}_p \cap \mathcal{E} = \emptyset$ **then**
 - 7: $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_p\}$
-

either the linear relaxation solution does not contain any cycle or a maximum number E_{max} of customers with elementarity requirements has already been identified. Note that the addition of elementarity requirements has priority on cutting planes and branching decisions. Note also that these requirements can be seen as cutting planes imposed at the subproblem level.

We tested several strategies to determine which customers should become elementary when a linear relaxation solution contains at least one cycle. The sequential procedure described in Algorithm 1 seems to be the best one. In this procedure where \mathcal{E} contains at the end the identified customers, at most one customer (visited more than once) per non-elementary path is selected. Preliminary tests showed that it is important to select the first one encountered along a path.

7 Computational results

This section reports the results derived from a series of computational experiments. First, we describe the instances used for our tests. Then, we discuss the efficiency of the tabu search column generator while solving linear relaxations. Next, we compare the quality of the bounds that can be obtained using various subproblem types (SPPRC, ESPPRC, PESPPRC, ...) and using generalized 2-path cuts. Finally, we present results for the complete solution of VRPTW instances.

All our tests were run on a Linux PC with a Dual Core AMD Opteron processor of 2.6 GHz, using a customized version of the Gencol software (version 4.5), which is an implementation of a branch-and-price-and-cut approach commercialized by Kronos Inc. Gencol relied on the ILOG Cplex solver (version 9.0) to solve the restricted master problems.

7.1 Test instances

Our tests were performed using the well-known VRPTW benchmark instances created by Solomon (1987). These 100-customer instances are divided into three problem classes which differ by the geographical distribution of the customers: they are clustered in the C instances, randomly located in the R instances, and partly clustered, partly randomly located in the RC instances. Each class is divided into two series. In the 100-series instances, time windows are, in general, narrower, yielding easier-to-solve instances. In this series, a route can contain approximately up to 25 customers. In the 200-series instances where time windows are wider, approximately up to 50 customers can be visited in a single route. There are 29 and 27 instances in the series 100 and 200, respectively. Instances are denoted as in the following example: RC204 stands for the 4th instance in the series 200 of the RC class. To allow comparisons with recent papers, we applied the convention that travel times and distances are calculated with one decimal point and truncation.

Note that smaller instances built from the 100-customer instances by keeping only the first 25 or 50 customers have also been studied in the literature. Since all these instances have now been solved to optimality, we restricted our study to the 100-customer instances. Out of these 56 instances, 10 of them (all in the series 200) remained open prior to this paper.

7.2 Linear relaxation results

To evaluate the efficiency of the tabu search column generator when the column generation approach uses an ESPPRC subproblem, we performed the following experiments. First, we selected 12 VRPTW instances (4 from the series 100 and 8 from the series 200) that are among the most difficult to solve. Then, we solved the linear relaxation of each instance using two different multiple-phase procedures as described in Section 4.4. In each phase of the first procedure (see Table 1 in which *hdp* stands for heuristic dynamic programming), the algorithm sequence starts with a tabu search algorithm. The second procedure is identical to the first except that the tabu search algorithms with $I_{max} = 30, 150$ and 200 are replaced by very fast heuristic dynamic programming algorithms with $(A_{min}, R_{max}) = (5, 0), (10, 10),$ and $(10, 10),$ respectively. Notice that the procedures have been adjusted for each series of instances (two phases for the series 100 and three for the series 200).

The computational results obtained from these experiments are reported in Table 2. For each tested instance and each procedure, this table indicates the total CPU time (in seconds) for solving the linear relaxation, the total number of column generation iterations (*nb it*), and the total numbers of iterations where the last algorithm used was a tabu search algorithm (*nb tabu*), a heuristic dynamic programming algorithm (*nb hdp*), or the exact dynamic programming algorithm (*nb edp*), respectively. As one can observe, the tabu search algorithm is more efficient than the heuristic dynamic programming algorithm for avoiding the use of the exact dynamic programming algorithm. In fact, in 9 of the

Table 1: Algorithm sequences for the procedures using tabu search algorithms

Phase	100-series instances	200-series instances
I	1- tabu with $I_{max} = 30$ and $C_{max} = 300$ 2- hdp with $A_{min} = 10$ and $R_{max} = 0$ 3- hdp with $A_{min} = 15$ and $R_{max} = 10$	1- tabu with $I_{max} = 30$ and $C_{max} = 500$ 2- hdp with $A_{min} = 5$ and $R_{max} = 5$
II	1- tabu with $I_{max} = 150$ and $C_{max} = 300$ 2- hdp with $A_{min} = 20$ and $R_{max} = 3$ 3- hdp with $A_{min} = 100$ and $R_{max} = 100$	1- tabu with $I_{max} = 30$ and $C_{max} = 500$ 2- hdp with $A_{min} = 10$ and $R_{max} = 10$ 3- hdp with $A_{min} = 10$ and $R_{max} = 30$
III		1- tabu with $I_{max} = 200$ and $C_{max} = 500$ 2- hdp with $A_{min} = 20$ and $R_{max} = 40$ 3- hdp with $A_{min} = 100$ and $R_{max} = 100$

Table 2: Linear relaxation results

Instance	with tabu					without tabu			
	CPU (s)	nb it	nb tabu	nb hdp	nb edp	CPU (s)	nb it	nb hdp	nb edp
RC104	36	77	57	19	1	35	152	149	3
RC108	26	56	40	14	2	42	136	131	5
R108	26	84	66	17	1	23	102	99	3
R112	30	61	47	12	2	23	61	59	2
RC203	95	117	103	13	1	1131	608	599	9
RC206	68	140	116	23	1	78	449	448	1
RC207	148	133	104	28	1	397	374	371	3
R203	172	214	174	39	1	758	613	601	12
R205	97	170	135	34	1	77	351	349	2
R206	210	183	137	45	1	864	511	503	8
R209	181	204	164	39	1	278	405	403	2
R210	269	184	141	41	2	1420	451	438	13

12 instances, the exact dynamic programming algorithm was invoked only once in the last iteration to prove optimality. Tabu search also allows to reduce the total number of iterations since it generates much more columns per iteration in the early iterations than the heuristic dynamic programming algorithms. This efficiency results in substantial computational time reductions for the most difficult instances, while it does not really change the solution times for the others. The procedures detailed in Table 1 was, therefore, used for the rest of our tests.

7.3 Bounds

This section focusses on the quality of the bounds that can be obtained with various sub-problem types and the generalized 2-path cuts. In particular, we compare the lower bounds yielded by the PESPPRC and the ESPPRC, as well as those derived using the generalized

2-path cuts instead of the traditional ones. Note that, for the 2-path cuts, we used the separation procedure described in Section 5.1 with the following parameter values for the 100-series instances: $S_{max} = 15$ and $U_{max} = 100$. For the 200-series instances, we were not successful at generating more than a few generalized 2-path cuts on some of the instances even when the value of S_{max} was set to 25. Since the bound improvements were negligible (less than 0.1) when these cuts were added and the times needed to compute them were rather large, we decided not to use them for the 200-series instances.

Tables 3 and 4 report lower bounds for all 100-series and 200-series instances, respectively. These lower bounds are those derived by using the SPPRC, the SPPRC with 2-cycle elimination (2-CE), the SPPRC with 3-cycle elimination (3-CE) as reported in Irnich and Villeneuve (2003), the PESPPRC with 2-cycle elimination and a maximum number of elementary customers (20 and 30 for series 100, and 40 and 50 for series 200), the ESPPRC, the ESPPRC with the traditional 2-path cuts (2-PC), and the ESPPRC with the generalized 2-path cuts (G2-PC). The last column of each table provides an upper bound which corresponds to the instance optimal value (without asterisk) or the cost of its best known feasible solution (see Danna and LePape, 2005, for instances with one asterisk and Section 7.5 for those with two asterisks). Columns in these tables have been ordered in increasing value of the bounds. Dashes (-) identify cases for which the lower bound could not be computed in less than 5 hours of computational time. Finally, *id* (for identical) indicates that the previous bound (on the same line) could not be improved using a stronger subproblem or a new type of cuts. For example, for instance C109, the solution computed using PESPPRC with $E_{max} = 20$ required 16 elementary customers. Therefore, the bound provided by this solution could not be improved using $E_{max} = 30$, nor the ESPPRC. Furthermore, since the computed solution was integer, it was optimal and satisfied all generalized 2-path cuts.

From the results in Tables 3 and 4, we make the following observations. First, the integrality gaps between the upper bounds and the first lower bounds are quite large for most instances, those in the series 200 (with large time windows) yielding the largest gaps. Second, 2-cycle elimination is very helpful to improve the quality of the bounds derived with SPPRC. Third, 3-cycle elimination is also useful but does not yield the same bound quality as that obtained with the ESPPRC or the PESPPRC, especially for the most difficult instances. Fourth, using the ESPPRC closes a large part of the gaps, especially for the 200-series instances. Fifth, the results with the PESPPRC show that, to obtain the same bound as the one yielded by the ESPPRC, elementarity is not needed for all customers. In fact, in most cases, a maximum of 30 and 50 elementary customers is sufficient for the 100-series and 200-series instances, respectively. It should be noticed that the solution approach requires, in general, less customer resources with the PESPPRC than with the ESPPRC even when the same bound value is obtained. For example, for instance C204, 4 customer resources were needed with the PESPPRC, while 30 customer resources were activated by the decremental search space strategy when using the ESPPRC. Sixth, as shown in Kohl et al. (1999), the traditional 2-path cuts substantially improve the quality

Table 3: Bound comparison for the 100-series instances

Instance	SPPRC	SPPRC 2-CE	SPPRC 3-CE	PESPPRC $E_{max} = 20$	PESPPRC $E_{max} = 30$	ESPPRC	ESPPRC 2-PC	ESPPRC G2-PC	UB
C101	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
C102	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
C103	826.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	826.3
C104	821.6	822.9	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	822.9
C105	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
C106	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
C107	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
C108	817.4	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
C109	809.3	822.9	827.3	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	827.3
RC101	1567.5	1584.1	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	1617.4	1617.5	1619.8
RC102	1380.2	1403.6	1404.9	1406.3	<i>id</i>	<i>id</i>	1441.9	1442.0	1457.4
RC103	1170.3	1218.5	1221.7	1224.6	1225.5	<i>id</i>	1245.4	1245.9	1258.0
RC104	1052.6	1094.3	1097.2	1098.7	1101.1	1101.8	1116.8	1117.8	1132.3
RC105	1453.9	1471.2	1471.8	1471.9	<i>id</i>	<i>id</i>	1509.8	1510.6	1513.7
RC106	1249.0	1308.8	1317.2	1318.0	1318.8	<i>id</i>	1343.1	1347.2	1372.7
RC107	1117.3	1170.7	1181.1	1182.8	1183.4	<i>id</i>	1196.4	1200.0	1207.8
RC108	1035.9	1063.0	1066.8	1068.3	1072.9	1073.4	1105.2	1107.7	1114.2
R101	1631.1	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	1634.0	1634.0	1637.7
R102	1466.6	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	1466.6
R103	1203.2	1206.3	1206.5	1206.8	<i>id</i>	<i>id</i>	1206.8	1207.1	1208.7
R104	937.1	949.1	955.3	955.4	956.9	956.9	957.2	957.5	971.5
R105	1341.2	1346.1	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	1349.3	1349.4	1355.3
R106	1212.3	1226.4	1226.4	1226.9	<i>id</i>	<i>id</i>	1227.9	1228.1	1234.6
R107	1037.0	1051.8	1052.2	1053.2	<i>id</i>	<i>id</i>	1054.4	1054.7	1064.6
R108	891.7	907.2	911.5	913.3	913.5	<i>id</i>	915.1	915.5	932.1
R109	1097.5	1130.6	1134.3	1134.3	<i>id</i>	<i>id</i>	1135.1	1135.6	1146.9
R110	1021.3	1048.5	1052.5	1055.4	1055.6	<i>id</i>	1056.0	1056.2	1068.0
R111	1005.9	1032.0	1034.2	1034.7	1034.7	<i>id</i>	1034.9	1035.1	1048.7
R112	892.5	919.2	923.6	926.0	926.7	926.7	927.9	929.2	948.6

of the bounds for several 100-series instances. However, additional generalized 2-path cuts do not have a major impact on the bound quality even, if in certain cases, up to 100 of these cuts were generated.

The reader interested in the quality of the bounds yielded by the subset row cuts can consult the paper of Jepsen et al. (2006). Also, as indicated in the next section by the number of branch-and-bound nodes required for solving the instances, these cuts often completely close the integrality gap.

Table 4: Bound comparison for the 200-series instances

Instance	SPPRC	SPPRC 2-CE	SPPRC 3-CE	PESPPRC $E_{max} = 40$	PESPPRC $E_{max} = 50$	ESPPRC	UB
C201	589.1	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	589.1
C202	589.1	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	589.1
C203	585.8	588.7	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	588.7
C204	582.2	586.0	-	588.1	<i>id</i>	<i>id</i>	588.1
C205	582.4	586.4	<i>id</i>	<i>id</i>	<i>id</i>	<i>id</i>	586.4
C206	576.0	585.4	586.0	<i>id</i>	<i>id</i>	<i>id</i>	586.0
C207	570.5	582.0	585.1	585.8	<i>id</i>	<i>id</i>	585.8
C208	570.2	581.8	585.8	<i>id</i>	<i>id</i>	<i>id</i>	585.8
RC201	1108.9	1240.4	1255.4	1255.9	<i>id</i>	<i>id</i>	1261.8
RC202	882.6	1004.1	1050.9	1088.1	<i>id</i>	<i>id</i>	1092.3
RC203	698.1	815.3	-	920.6	922.5	<i>id</i>	923.7
RC204	608.6	688.3	-	772.4	777.9	-	783.5*
RC205	967.1	1055.5	1117.9	1147.6	<i>id</i>	<i>id</i>	1154.0
RC206	852.2	952.2	1005.9	1034.6	1037.4	1038.6	1051.1
RC207	769.1	866.3	886.9	931.5	937.8	947.3	962.9
RC208	627.2	704.6	-	762.2	764.4	-	776.5**
R201	1080.7	1136.2	1140.0	1140.3	<i>id</i>	<i>id</i>	1143.2
R202	934.2	1009.8	-	1022.2	<i>id</i>	<i>id</i>	1029.6
R203	756.7	846.5	-	863.7	866.9	<i>id</i>	870.8
R204	640.4	689.8	-	722.1	724.7	-	731.3**
R205	838.4	916.6	930.6	937.9	938.9	<i>id</i>	949.8
R206	749.1	834.6	-	865.9	866.5	866.9	875.9
R207	668.9	747.6	-	786.3	789.8	790.7	794.0
R208	610.7	661.7	-	-	-	-	701.2*
R209	750.4	819.8	833.9	839.5	840.9	841.4	854.8
R210	754.0	849.3	-	884.5	885.7	889.4	900.5
R211	650.8	705.8	-	731.9	734.4	-	746.7**

7.4 Integer solution results

To obtain integer solutions, we applied the cutting and branching strategies described in Section 5. Given the small bound improvement provided by the use of the generalized 2-path cuts (see Table 3), we decided to set U_{max} to 1 in order to substantially reduce the separation time of these cuts. The parameters for the subset row cuts were set as follows: $Q_{max} = 30$, $N_{max} = 5$, $V_{min} = 0.1$, and $M_{max} = 10$. All the other parameters were set to the values specified in previous sections. After solving a linear relaxation and when needed, either elementarity requirements, generalized 2-path cuts, subset row cuts, or branching decisions were added in this order.

Tables 5 and 6 report the computational results for the 100- and the 200-series instances, respectively. For each instance, we report the computational time in seconds, the number of cuts generated, and the number of branch-and-bound nodes explored for different solution

approaches described in the table headers (SRC stands for subset row cuts). Note that the last two columns of each table provide the results obtained by Jepsen et al. (2006) (except the number of cuts which is not reported in their paper) using an Intel Pentium 4 processor of 3.0 GHz. In these tables, *id* indicates that the previous solution time (on the same line) was identical because either no additional elementarity requirements were needed or the linear relaxation solution was already integer without adding any generalized 2-path cuts. Dashes (-) identify instances that could not be solved to optimality. Bold figures highlight newly solved instances.

The results in these tables show that subset row cuts are very efficient for reducing the number of branch-and-bound nodes. Indeed, a large number of these instances require a single node when subset row cuts are generated. For example, when using the ESPPRC as a subproblem, only one node, 78 subset row cuts, and 641 seconds are needed for solving instance R203, while it requires 793 nodes and 57085 seconds without the subset row cuts (these last figures are not reported in the table). Note that, when compared to the Jepsen et al. (2006) approach, our approach yields a slightly larger number of branching nodes for certain instances because the parameter V_{min} restricts generating cuts that are barely violated.

The use of the PESPPRC subproblem does not necessarily increase the number of branching nodes even though it yields slightly weaker lower bounds. On the other hand, it generally produces higher solution times, especially for smaller E_{max} values. Solution time increases are typically due to a larger number of cuts that need to be generated to compensate for weaker bounds and to the progressive addition of elementarity requirements. Nevertheless, there are exceptions among the most difficult instances (namely, C204, R206, R207, and R209) which show that it might be advantageous to reduce the difficulty of solving the subproblem when the quality of the lower bound is almost the same as the one obtained with the ESPPRC subproblem.

The application of generalized 2-path cuts for the 100-series instances does not seem to be very beneficial except for instances RC103, RC104, RC108, and R104. In some cases (instances R109 and R112), it even considerably deteriorates solution times.

When compared to the results of Jepsen et al. (2006) who ran their tests on a slightly faster computer, we obtain very substantial time reductions (up to 99% for C104, C203, and R203) for the most difficult instances when using the same ingredients (ESPPRC and SRC). For some of the easiest instances, our approach performs slightly worse than the Jepsen et al.'s approach because the subset row cuts are added more carefully in our approach, yielding more rounds of cuts and, thus, additional reoptimizations. Notice that the high efficiency of the tabu search heuristic (and the overall column generation procedure described in Section 4.4) for the ESPPRC subproblem is again shown by the fast solution times (in comparison with Jepsen et al.) obtained for the 200-series instances of class C for which no branching decisions nor cuts were needed. A minimal time reduction of 93%

Table 5: Integer programming results for the 100-series instances

Instance	DESAULNIERS, LESSARD, HADJAR												JEPSEN et al.	
	PESPPRC $E_{max} = 20$ G2-PC SRC			PESPPRC $E_{max} = 30$ G2-PC SRC			ESPPRC G2-PC SRC			ESPPRC SRC			ESPPRC SRC	
	CPU	nb	nb	CPU	nb	nb	CPU	nb	nb	CPU	nb	nb	CPU	nb
	(s)	cuts	BB	(s)	cuts	BB	(s)	cuts	BB	(s)	cuts	BB	(s)	BB
C101	2	0	1	<i>id</i>	<i>id</i>	<i>id</i>	2	0	1	<i>id</i>	<i>id</i>	<i>id</i>	3	1
C102	8	0	1	<i>id</i>	<i>id</i>	<i>id</i>	8	0	1	<i>id</i>	<i>id</i>	<i>id</i>	13	1
C103	16	0	1	<i>id</i>	<i>id</i>	<i>id</i>	28	0	1	<i>id</i>	<i>id</i>	<i>id</i>	34	1
C104	22	0	1	<i>id</i>	<i>id</i>	<i>id</i>	86	0	1	<i>id</i>	<i>id</i>	<i>id</i>	4113	1
C105	3	0	1	<i>id</i>	<i>id</i>	<i>id</i>	3	0	1	<i>id</i>	<i>id</i>	<i>id</i>	5	1
C106	4	0	1	<i>id</i>	<i>id</i>	<i>id</i>	4	0	1	<i>id</i>	<i>id</i>	<i>id</i>	7	1
C107	4	0	1	<i>id</i>	<i>id</i>	<i>id</i>	4	0	1	<i>id</i>	<i>id</i>	<i>id</i>	7	1
C108	8	0	1	<i>id</i>	<i>id</i>	<i>id</i>	7	0	1	<i>id</i>	<i>id</i>	<i>id</i>	14	1
C109	20	0	1	<i>id</i>	<i>id</i>	<i>id</i>	16	0	1	<i>id</i>	<i>id</i>	<i>id</i>	21	1
RC101	18	148	1	<i>id</i>	<i>id</i>	<i>id</i>	17	148	1	19	87	1	12	1
RC102	118	224	1	119	212	1	124	225	1	120	193	3	77	1
RC103	390	267	5	341	239	5	295	247	5	541	262	5	2706	3
RC104	6022	440	27	3707	396	17	2874	345	17	11773	437	21	65807	7
RC105	17	58	1	<i>id</i>	<i>id</i>	<i>id</i>	14	57	1	33	79	1	27	1
RC106	3115	914	73	3045	846	63	3268	1017	83	3916	755	71	15892	37
RC107	157	186	1	92	117	1	135	158	1	161	158	1	154	1
RC108	315	201	1	306	205	1	279	138	1	635	228	1	3365	1
R101	8	16	15	<i>id</i>	<i>id</i>	<i>id</i>	8	16	15	8	19	15	2	3
R102	3	0	1	<i>id</i>	<i>id</i>	<i>id</i>	3	0	1	<i>id</i>	<i>id</i>	<i>id</i>	4	1
R103	32	72	1	<i>id</i>	<i>id</i>	<i>id</i>	21	76	1	20	53	1	24	1
R104	3334	417	12	2612	460	11	2260	406	11	3103	391	11	32343	3
R105	26	144	3	<i>id</i>	<i>id</i>	<i>id</i>	25	144	3	36	144	3	43	5
R106	135	169	3	<i>id</i>	<i>id</i>	<i>id</i>	122	162	3	87	144	3	75	1
R107	623	272	7	281	193	3	359	205	5	416	227	5	1310	3
R108	1054	383	1	921	345	1	875	325	1	891	296	1	5912	1
R109	1393	587	59	1383	596	55	1406	655	69	1127	588	65	1432	19
R110	490	280	5	484	227	9	459	219	9	426	219	5	1068	3
R111	6183	946	105	5145	968	95	5411	881	101	5738	736	111	83931	39
R112	17984	723	26	24409	727	31	19313	605	21	16073	574	19	202804	9

was obtained for these instances. Furthermore, we could solve instance C204, while Jepsen et al. were unable to do so.

Finally, we would like to point out that, with different variants of our approach, we were able to solve five open instances of the Solomon benchmark dataset, namely, instances

Table 6: Integer programming results for 200-series instances

Instance	DESAULNIERS, LESSARD, HADJAR									JEPSEN et al.	
	PESPPRC $E_{max} = 40$			PESPPRC $E_{max} = 50$			ESPPRC			ESPPRC	
	SRC			SRC			SRC			SRC	
	CPU (s)	nb cuts	nb BB	CPU (s)	nb cuts	nb BB	CPU (s)	nb cuts	nb BB	CPU (s)	nb BB
C201	11	0	1	<i>id</i>	<i>id</i>	<i>id</i>	9	0	1	203	1
C202	103	0	1	<i>id</i>	<i>id</i>	<i>id</i>	49	0	1	3483	1
C203	282	0	1	<i>id</i>	<i>id</i>	<i>id</i>	122	0	1	13070	1
C204	539	0	1	<i>id</i>	<i>id</i>	<i>id</i>	16416	0	1	-	-
C205	56	0	1	<i>id</i>	<i>id</i>	<i>id</i>	15	0	1	417	1
C206	162	0	1	<i>id</i>	<i>id</i>	<i>id</i>	24	0	1	595	1
C207	220	0	1	<i>id</i>	<i>id</i>	<i>id</i>	84	0	1	1241	1
C208	390	0	1	<i>id</i>	<i>id</i>	<i>id</i>	26	0	1	555	1
RC201	143	53	3	<i>id</i>	<i>id</i>	<i>id</i>	92	55	3	229	3
RC202	246	34	1	<i>id</i>	<i>id</i>	<i>id</i>	89	39	1	313	1
RC203	1240	96	1	842	62	1	324	47	1	14917	1
RC204	-	-	-	-	-	-	-	-	-	-	-
RC205	207	48	1	183	33	1	111	32	1	221	1
RC206	580	85	1	541	68	1	344	73	1	340	1
RC207	-	-	-	464209	290	5	91405	210	5	-	-
RC208	-	-	-	-	-	-	-	-	-	-	-
R201	94	32	1	<i>id</i>	<i>id</i>	<i>id</i>	78	52	1	139	1
R202	1365	156	19	<i>id</i>	<i>id</i>	<i>id</i>	1663	152	17	8282	13
R203	1323	97	1	1102	72	1	641	78	1	54187	1
R204	-	-	-	-	-	-	-	-	-	-	-
R205	14340	381	11	21708	530	17	6904	345	9	-	-
R206	29766	270	1	26940	245	1	60608	171	1	-	-
R207	36966	158	1	4214	81	1	11228	24	1	-	-
R208	-	-	-	-	-	-	-	-	-	-	-
R209	27691	319	3	10270	307	3	22514	248	3	78560	3
R210	-	-	-	-	-	-	400904	266	5	-	-
R211	-	-	-	-	-	-	-	-	-	-	-

RC207, R205, R206, R207, and R210. Optimal solutions for these instances are given in Table 7.

7.5 Unsolved instances

To conclude this section on computational results, we present in Table 8 lower and upper bounds (LB and UB) on the optimal value of the remaining unsolved instances. These bounds have either been published in the literature (see Danna and LePape, 2005) or obtained during the numerous tests that we made for this paper.

Table 7: Optimal solutions for instances RC207, R205, R206, R207, and R210

Solution for RC207. Total cost = 962.9. Number of vehicles = 6.	
Cost	Route
157.2	82, 99, 52, 22, 19, 21, 23, 25, 77, 75, 58, 74, 86, 57, 90
168.2	61, 72, 71, 93, 94, 81, 42, 44, 40, 36, 35, 37, 38, 39, 43, 41, 54, 96
141.2	88, 78, 73, 79, 7, 6, 2, 8, 5, 3, 1, 45, 46, 4, 100, 70, 68
155.3	69, 98, 53, 12, 14, 47, 17, 16, 15, 11, 9, 87, 59, 97, 13, 10, 60, 55
154.9	92, 95, 67, 62, 33, 30, 31, 29, 27, 28, 26, 32, 34, 50, 56, 91, 80
186.1	65, 83, 64, 84, 85, 63, 51, 49, 18, 76, 89, 48, 20, 24, 66
Solution for R205. Total cost = 949.8. Number of vehicles = 5.	
Cost	Route
221.4	95, 59, 92, 98, 14, 42, 15, 2, 73, 72, 39, 67, 23, 75, 41, 22, 74, 56, 4, 25, 55, 54, 24, 80, 68, 77
248.2	28, 12, 29, 33, 65, 71, 9, 81, 51, 30, 76, 50, 3, 79, 78, 34, 35, 66, 20, 32, 10, 70, 1
201.8	27, 69, 31, 88, 62, 11, 63, 90, 64, 49, 19, 7, 18, 8, 46, 48, 17, 60, 89
239.2	52, 82, 47, 36, 45, 83, 5, 84, 61, 16, 44, 38, 86, 85, 99, 6, 94, 97, 87, 57, 43, 91, 100, 37, 93, 96, 13, 58
39.2	53, 40, 21, 26
Solution for R206. Total cost = 875.9. Number of vehicles = 5.	
Cost	Route
206.4	52, 7, 82, 45, 48, 47, 36, 19, 11, 64, 49, 46, 8, 18, 83, 84, 17, 5, 60, 89
134.0	27, 69, 31, 88, 62, 63, 90, 30, 51, 9, 81, 78, 79, 76, 12, 26, 40, 53
202.0	92, 98, 37, 42, 15, 14, 44, 38, 86, 16, 61, 99, 96, 6, 94, 97, 87, 2, 57, 43, 100, 91, 85, 93, 59, 95, 13, 58
169.8	28, 50, 33, 3, 29, 34, 71, 65, 35, 66, 20, 32, 10, 70, 1
163.7	21, 73, 72, 39, 67, 23, 41, 22, 74, 75, 56, 4, 25, 55, 54, 24, 80, 68, 77
Solution for R207. Total cost = 794.0. Number of vehicles = 4.	
Cost	Route
204.5	27, 69, 1, 50, 33, 51, 30, 20, 65, 9, 81, 79, 78, 34, 35, 71, 66, 32, 90, 10, 70, 31, 52
193.5	28, 76, 54, 39, 67, 23, 56, 75, 41, 22, 74, 72, 73, 21, 4, 55, 25, 24, 29, 3, 77, 68, 80, 12, 26
239.8	89, 18, 45, 46, 36, 47, 48, 7, 88, 62, 11, 63, 64, 49, 19, 82, 8, 83, 60, 5, 84, 17, 61, 91, 100, 37, 98, 93, 59, 95, 13, 58
156.2	94, 92, 42, 57, 15, 43, 14, 44, 38, 86, 16, 85, 99, 96, 6, 97, 87, 2, 40, 53
Solution for R210. Total cost = 900.5. Number of vehicles = 6.	
Cost	Route
147.8	27, 69, 1, 30, 51, 33, 71, 65, 66, 20, 32, 90, 10, 70, 31
144.5	28, 12, 76, 3, 79, 29, 78, 81, 9, 35, 34, 24, 80, 68, 77, 50
228.4	52, 7, 82, 48, 47, 36, 19, 88, 62, 11, 63, 64, 49, 46, 8, 84, 17, 85, 98, 37, 100, 91, 93, 5, 60, 89
57.4	6, 94, 96, 99, 59, 87, 97, 13, 58
186.0	18, 83, 45, 61, 16, 86, 44, 38, 14, 43, 41, 22, 74, 56, 4, 55, 25, 54, 26
136.4	95, 92, 42, 15, 57, 2, 23, 67, 39, 75, 72, 73, 21, 40, 53

Table 8: Best known bounds for the unsolved instances

Instance	LB	UB
RC204	778.0	783.5
RC208	764.5	776.5
R204	724.8	731.3
R208	680.1	701.2
R211	734.4	746.7

8 Conclusions

This paper has presented three new ideas for improving the efficiency of branch-and-price-and-cut approaches applied to the VRPTW. First, the tabu search method for heuristically solving the subproblem and rapidly generating negative reduced cost columns showed to be very efficient. Second, the generalization of the k -path inequalities brought an interesting theoretical contribution for the VRPTW but did not prove to be very useful in practice. Third, the PESPPRC subproblem provided a compromise between lower bound quality and subproblem complexity. For certain instances, it showed to be very useful for improving solution times. Combining these ideas with the most recent advances from the literature, we succeeded to solve 5 of the 10 previously unsolved Solomon’s (1987) benchmark instances with 100 customers, and to obtain substantial time reductions for the most difficult solvable 100-customer instances when compared to the best results found in the literature.

Future research ensuing from this paper will focus on improving the tabu search method, better exploiting the generalized k -path inequalities, and better determining the customers on which to impose elementarity requirements when the PESPPRC is used as a subproblem.

References

- Bard, J.F., G. Kontoravdis, and G. Yu (2002). A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science* 36, 250–269.
- Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research* 46, 316–329.
- Boland, N., J. Dethridge, and I. Dumitrescu (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters* 34, 58–68.
- Bräysy, O., and M. Gendreau (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science* 39, 104–118.
- Bräysy, O., and M. Gendreau (2005b). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science* 39, 119–139.

- Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research* 33, 2972–2990.
- Cordeau, J.-F., G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis (2001). The VRP with time windows. In: P. Toth and D. Vigo (eds), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, 157–194.
- Cook, W., and J.L. Rich (1999). A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Department of Computational and Applied Mathematics, Rice University.
- Danna, E., and C. Le Pape (2005). Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In: *Column Generation*, G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds), Springer, New-York, 99–129.
- Dantzig, G.B., and P. Wolfe (1960). Decomposition principle for linear programs. *Operations Research* 8, 101–111.
- Desaulniers, G., J. Desrosiers, I. Ioachim, F. Soumis, M.M. Solomon, and D. Villeneuve (1998). A unified framework for time constrained vehicle routing and crew scheduling problems. In: *Fleet Management and Logistics*, T.G. Crainic and G. Laporte (eds), Kluwer, Norwell, MA, 57–93.
- Desrochers, M., J. Desrosiers, and M.M. Solomon (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40, 342–354.
- Desrochers, M. (1986). *La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes*. Ph.D. dissertation, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada. Publication #470, in French.
- Desrosiers, J., and M.E. Lübbecke (2005). A primer in column generation. In: *Column Generation*, G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds), Springer, New-York, 1–32.
- Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research* 42, 977–978.
- du Merle, O., D. Villeneuve, J. Desrosiers, and P. Hansen (1999). Stabilized Column Generation. *Discrete Mathematics* 194, 229–237.
- Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44, 216–229.
- Feillet, D., M. Gendreau, and L.-M. Rousseau (2005). New refinements for the solution of vehicle routing problems with branch and price. Technical Report, Centre de recherche sur les transports, Université de Montréal.
- Glover, F., and M. Laguna (1997). *Tabu search*. Kluwer, Norwell, MA.

- Irnich, S., and G. Desaulniers (2005). Shortest Path Problems with Resource Constraints. In: *Column Generation*, G. Desaulniers, J. Desrosiers and M.M. Solomon (eds.), Springer, New-York, NY, 33–65.
- Irnich, S., and D. Villeneuve (2003). The shortest path problem with resource constraints and k -cycle elimination for $k \geq 3$. *Les Cahiers du GERAD* G-2003-55, HEC Montréal, Montréal. To appear in *INFORMS Journal on Computing*.
- Jepsen, M., B. Petersen, S. Spoorendonk, and D. Pisinger (2006). A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. Technical Report 06-03, Department of Computer Science, University of Copenhagen, Denmark.
- Kallehauge, B., and N. Boland (2005). Path inequalities for the vehicle routing problem with time windows. Technical Report, Center for Traffic and Transport, Technical University of Denmark.
- Kallehauge, B., J. Larsen, O.B.G. Madsen, and M.M. Solomon (2005). Vehicle routing problem with time windows. In: *Column Generation*, G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds), Springer, New-York, 67–98.
- Kohl, N., and O.B.G. Madsen (1997). An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean relaxation. *Operations Research* 45, 395–406.
- Kohl, N., J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science* 33, 101–116.
- Lysgaard, J. (2004). Reachability cuts for the vehicle routing problem with time windows. Technical Report, Logistics/SCM Research Group, Aarhus School of Business, Denmark.
- Righini, G., and M. Salani (2005). New dynamic programming algorithms for the resource-constrained elementary shortest path problem. Technical Report No. 69, University of Milano, Italy.
- Righini, G., and M. Salani (2004). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Technical Report No. 66, University of Milano, Italy.
- Ropke, S. (2003). A general heuristic for vehicle routing problems. *Route 2003, International Workshop on Vehicle Routing*, Skodsborg, Denmark, June 22–25.
- Salani, M. (2005). *Branch-and-price algorithms for vehicle routing problems*. Ph.D. thesis, University of Milano, Italy.
- Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research* 35, 254–265.
- Toth, P., and D. Vigo, editors (2002). *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications.