The Vehicle Routing Problem with Time Windows and Temporal Dependencies

Anders Dohn, Matias Sevel Rasmussen, and Jesper Larsen
Department of Management Engineering, Technical University of Denmark, Kgs. Lyngby, Denmark

In this article, we formulate the vehicle routing problem with time windows and temporal dependencies. The problem is an extension of the well studied vehicle routing problem with time windows. In addition to the usual constraints, a scheduled time of one visit may restrain the scheduling options of other visits. Special cases of temporal dependencies are synchronization and precedence constraints. Two compact formulations of the problem are introduced and the Dantzig–Wolfe decompositions of these formulations are presented to allow for a column generation-based solution approach. Temporal dependencies are modeled by generalized precedence constraints. Four different master problem formulations are proposed and it is shown that the formulations can be ranked according to the tightness with which they describe the solution space. A tailored time window branching is used to enforce feasibility on the relaxed master problems. Finally, a computational study is performed to quantitatively reveal strengths and weaknesses of the proposed formulations. It is concluded that, depending on the problem at hand, the best performance is achieved either by relaxing the generalized precedence constraints in the master problem, or by using a time-indexed model, where generalized precedence constraints are added as cuts when they become severely violated. © 2011 Wiley Periodicals, Inc. NETWORKS, Vol. 58(4), 273–289 2011

Keywords: vehicle routing with time windows; temporal dependency; generalized precedence constraints; time-window branching; relaxation; column generation; branch-and-price; branch-and-cut-and-price; set partitioning; set covering; integer programming

1. INTRODUCTION

The vehicle routing problem with time windows and temporal dependencies (VRPTWTD) is an extension of the vehicle routing problem with time windows (VRPTW). Given is a fixed set of customers with individual demands and with time windows specifying when each customer accepts service. The objective is to find routes for a number of vehicles, all starting and ending at a central depot in such a way that the total cost is minimized. The extension that we present here is concerned with temporal dependencies between customers. A temporal dependency which is often encountered in practical instances and that has received the most attention in the literature, is the rather strict requirement of synchronization between two visits. Synchronization on visits is also used to model rendezvous between vehicles. Other, less restrictive, dependencies are constraints on minimum overlap between visits and limits on minimum or maximum gaps between visits.

In this article, a context-free approach to VRPTWTD is presented for the first time. We apply time window branching combined with time window reductions to restore feasibility with respect to temporal dependencies. We prove that the standardized modeling of temporal dependencies as generalized precedence constraints does not affect the efficiency of the solution method. Along with a direct formulation and a relaxed formulation, we introduce a time-indexed formulation with an implicit representation of generalized precedence constraints. We are able to rank the formulations theoretically, according to the tightness with which they describe the solution space. For computational testing, we introduce a fourth model, which is a hybrid of the relaxed formulation and the time-indexed formulation. Finally, we introduce a new set of context-free benchmark instances which enables a thorough quantitative analysis and which we hope will facilitate future research in this area. The main contribution of this article is the formulation and comparison of models for VRPTWTD along with a generic and efficient solution approach.

There is a vast amount of literature on VRPTW and its variants. VRPTW is known to be NP-hard [35]; nevertheless exact solution of the problem has received a lot of attention. The most successful approach is based on a Dantzig–Wolfe decomposition [11] of the mathematical model using column generation in a branch-and-cut-and-price framework. The method was first proposed by Desrochers et al. [13]. The most promising recent work is based on solution of the subproblem as an elementary shortest path problem with

Received October 2009; accepted July 2011
Correspondence to: A. Dohn; e-mail: adohn@man.dtu.dk
Contract grant sponsor: Elite Forsk-grant of The Danish Ministry of Science, Technology and Innovation
DOI 10.1002/net.20472
Published online 29 October 2011 in Wiley Online Library (wileyonlinelibrary.com).

time windows and capacity constraints (ESPPTWCC). Feillet et al. [17] were the first to apply this idea and were followed by Chabrier [6], Danna and Pape [10], Jepsen et al. [22], and Desaulniers et al. [12] among others. The approach that we present here for VRPTWTD builds on the same idea. See Ref. [24] for a recent review of the literature and a thorough description of the technique.

The motivation behind this work is the many practical applications of VRPTWTD. With the inclusion of temporal dependencies in the model, we are able to describe numerous concrete problems. As Kilby et al. [25] point out, there is a need for more sophisticated models for the vehicle routing problem. They mention synchronization and precedence constraints as some of the relevant extensions.

Ioachim et al. [20] describe a fleet assignment and routing problem with synchronization constraints. The problem is solved by column generation. A similar problem with synchronization is described by Bélanger et al. [2]. Rousseau et al. [33] present the synchronized vehicle dispatching problem, which is a dynamic vehicle routing problem with synchronization between vehicles. Constraint programming and local search are applied to arrive at high-quality feasible solutions. Lim et al. [29] and Li et al. [28] study a problem from the Port of Singapore, where technicians are allocated to service jobs. For each job, a certain combination of technicians with individual skills is needed. The technicians must be present at the same time, and hence, the schedule for each technician must respect a number of synchronization constraints with other schedules. The problem is solved using metaheuristics. Another application with synchronization between visits is in ground handling at airports. Teams drive around at the airport and are assigned tasks on the parked aircraft. Dohn et al. [15] describe this setup and present exact solutions to the instances considered. Oron et al. [32] consider ground handling with synchronization constraints as well, and present computational results for a tailored heuristic applied to data instances from an in-flight caterer in Malaysia. Bredström and Rönnqvist [4] present an application of vehicle routing with synchronization constraints in home health care. A branch-and-price algorithm is applied to a realistic home care routing problem and yields promising results.

The generalization of synchronization to other temporal dependencies has been described for a few applications. Lesaint et al. [27] present a workforce scheduling software from a practical perspective. In the problem described, both synchronization and various other sequencing constraints occur. Fügenschuh [18] describes a problem in school bus routing. Busses must wait for each other at various intermediate stops and hence precedence relations are introduced for such stops. Fügenschuh refers to the problem as the vehicle routing problem with coupled time windows. Doerner et al. [14] describe an application in blood collection from satellite locations for a central blood bank. Multiple visits at each location have to be scheduled with a certain slack between them. They refer to the vehicle problem as having interdependent time windows. Bredström and Rönnqvist [5] modeled temporal dependencies for a home care routing problem in a mixed integer programming model (MIP) which was solved with a standard MIP solver. In Justesen and Rasmussen [23] a similar application is described and solved using branch-and-price. Bredström and Rönnqvist [5] have also continued their work in this direction. An application with general temporal dependencies in machine scheduling is described by Van den Akker et al. [37]. Column generation is used to solve the problem. The pricing problem is primarily solved heuristically by local search and occasionally to optimality using a standard solver on an integer programming formulation of the pricing problem. Van den Akker et al. [38] and Bigras et al. [3] describe machine scheduling problems and propose to apply column generation approaches to time-indexed formulations. Hence, their models have some similarities to the time-indexed formulation presented in this paper.

The article is organized as follows. In section 2, we present two valid compact formulations of VRPTWTD. Possible decompositions of the compact formulations are presented and compared in section 3. For the decomposed models, a tailored branching method is required, which is described in section 4. A set of test instances are introduced in section 5 and the test results for these are found in section 6. Finally, we conclude on our findings and discuss possible areas for future research in section 7.

2. MODEL

In the following, we present two valid model formulations for VRPTWTD, namely a mixed-integer formulation that we refer to as the direct formulation and a time-indexed formulation. The mixed-integer formulation is an extension of the model commonly used for VRPTW, whereas a time-indexed model has not received the same amount of attention.

In the traditional VRPTW, the objective is to find the cheapest set of routes to a set, $\mathcal{C}$, of n customers. Given is a fleet of identical vehicles, $\mathcal{V}$, which are located at a central depot. Typically, the depot is represented as two locations, namely a start depot, 0, and an end depot, $n+1$. Together with all customers, they form the set, $\mathcal{N}$. All vehicles have a capacity of q. Each customer, i, has a demand, d_i, and a time window, where it accepts service $[\alpha_i, \beta_i]$. α_i is the first possible service time. The vehicle is allowed to arrive before this time, but must then wait at the customer for the time window to open. β_i is the latest possible time of initiation at customer i. $[\alpha_0, \beta_0]$ denotes the scheduling horizon of the problem. Vehicles start at the depot at time α_0 and must return to the end depot no later than β_0. τ_{ij} gives the travel time between any two customers, i and j. This may include service time at customer i. Traveling between the two customers also incurs a certain cost given by c_{ij}. We assume that q, d_i, α_i, β_i, and c_{ij} are nonnegative integers and that τ_{ij} are positive integers, respecting the triangular inequality.

2.1. Direct Formulation

The mathematical model of VRPTW is presented below. x_{ijk} are binary variables with $x_{ijk} = 1$, if vehicle k drives

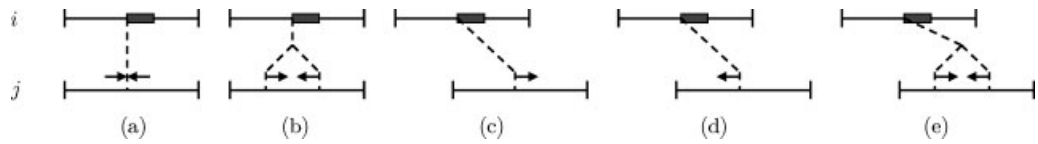

FIG. 1. Five kinds of temporal dependencies that are often encountered in practice. Each of the five subfigures shows the time windows of two customers i and j with a temporal dependency between them. Assuming some start time for customer i, the dashed line together with the arrows give the corresponding feasible part of the time window of customer j. (a) synchronization, (b) overlap, (c) minimum difference, (d) maximum difference, and (e) minimum + maximum difference.

directly from customer i to customer j, $x_{ijk} = 0$, otherwise. s_{ik} are continuous variables and are defined as the start time for service at customer i, if the customer is serviced by vehicle k. Otherwise, $s_{ik} = 0$. Without restricting the model, we can fix $s_{0k} = \alpha_0, \forall k \in \mathcal{V}$ and $s_{n+1,k} = \beta_0, \forall k \in \mathcal{V}$.

$$\min \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} c_{ij} x_{ijk} \tag{1}$$

$$\sum_{j\in\mathcal{N}:j\neq i} \sum_{k\in\mathcal{V}} x_{ijk} = 1 \qquad \forall i \in \mathcal{C} \tag{2}$$

$$\sum_{i\in\mathcal{C}} d_i \sum_{j\in\mathcal{N}} x_{ijk} \leq q \qquad \forall k \in \mathcal{V} \tag{3}$$

$$\sum_{j\in\mathcal{N}} x_{0jk} = 1 \qquad \forall k \in \mathcal{V} \tag{4}$$

$$\sum_{i\in\mathcal{N}} x_{ihk} - \sum_{j\in\mathcal{N}} x_{hjk} = 0 \qquad \forall h \in \mathcal{C}, \forall k \in \mathcal{V} \tag{5}$$

$$\sum_{i\in\mathcal{N}} x_{i,n+1,k} = 1 \qquad \forall k \in \mathcal{V} \tag{6}$$

$$s_{ik} + \tau_{ij} - M(1 - x_{ijk}) \leq s_{jk} \qquad \forall i,j \in \mathcal{N}, \forall k \in \mathcal{V} \tag{7}$$

$$\alpha_i \sum_{j\in\mathcal{N}} x_{ijk} \leq s_{ik} \leq \beta_i \sum_{j\in\mathcal{N}} x_{ijk} \qquad \forall i \in \mathcal{C}, \forall k \in \mathcal{V} \tag{8}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in \mathcal{N}, \forall k \in \mathcal{V} \tag{9}$$

The objective is to minimize the total cost of all edges traveled (1). All customers must be visited by exactly one vehicle (2), and the route for each vehicle must respect the capacity of that vehicle (3). (4) and (6) ensure that each route starts and ends at the depot. We also need to ensure that routes are not segmented, that is, if a vehicle arrives at a customer, it eventually leaves that customer again (5). If a vehicle is set to travel between two customers, there has to be enough time between the two visits (7). Finally, we need to make sure that all time windows are respected (8). (8) also ensure that $s_{ik} = 0$ when vehicle k does not visit customer i. (9) are the integrality constraints on x_{ijk}.

In VRPTWTD, we furthermore have a number of temporal dependencies between customers. We are able to express all of these by generalized precedence constraints. We introduce the parameter δ_{ij} which specifies the minimum difference in time from customer i to customer j. The set Δ defines all customer pairs (i,j) for which a temporal dependency exists. The generalized precedence constraints are formulated as follows, where $\sum_{k\in\mathcal{V}} s_{ik}$ is the start time of service at customer i.

$$\sum_{k\in\mathcal{V}} s_{ik} + \delta_{ij} \leq \sum_{k\in\mathcal{V}} s_{jk} \qquad \forall (i,j) \in \Delta \tag{10}$$

Constraint (10) can be used to model all the temporal dependencies that were observed in the literature review. There may be dependencies between several customers, for example, synchronization of three or more customers. Such dependencies are modeled by applying the corresponding pair wise dependencies. In this article, we will focus on five kinds of temporal dependencies that are commonly found in practice. These are visualized in Figure 1.

It is straight forward to model the temporal dependencies of Figure 1 using constraints (10). The correct values for δ_{ij} and δ_{ji} are listed in Table 1.

2.2. Time-Indexed Formulation

Time-indexed formulations have not received much attention in the column generation context of VRPTW. A time-indexed formulation is usually disregarded because of its vast size. It is, however, popular in the formulation of machine scheduling problems, as it gives a tight description of precedence constraints. Here, we present the time-indexed model of VRPTWTD, as it will be used to strengthen the bounds in the branch-and-price algorithm. We introduce the index $t \in \mathcal{T}$ on the x-variable, with $\mathcal{T} = \{\alpha_0, \ldots, \beta_0\}$. x_{ijkt} is defined as: $x_{ijkt} = 1$, if vehicle k services customer i at time t and then drives directly to customer j. $x_{ijkt} = 0$, otherwise. Further, define the auxiliary sets $\mathcal{T}^{\tau}_{tij} = \{\alpha_0, \ldots, \min\{\beta_0, t + \tau_{ij} - 1\}\}$ and $\mathcal{T}^{\delta}_{tij} = \{\alpha_0, \ldots, \min\{\beta_0, t + \delta_{ij} - 1\}\}$.

$$\min \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} \sum_{t\in\mathcal{T}} c_{ij} x_{ijkt} \tag{11}$$

TABLE 1. Parameter values for the five temporal dependencies of Figure 1.

	Temporal dependency	δ_{ij}	δ_{ji}
(a)	Synchronization	0	0
(b)	Overlap	$-\text{dur}_j$	$-\text{dur}_i$
(c)	Minimum difference	$\text{diff}_{\min}$	N/A
(d)	Maximum difference	N/A	$-\text{diff}_{\max}$
(e)	Minimum + maximum difference	$\text{diff}_{\min}$	$-\text{diff}_{\max}$

dur_i is the service time at customer i. $\text{diff}_{\min}$ and $\text{diff}_{\max}$ are, respectively, the minimum and maximum differences required.

$$\sum_{j\in\mathcal{N}:j\neq i}\sum_{k\in\mathcal{V}}\sum_{t\in\mathcal{T}} x_{ijkt} = 1 \qquad \forall i \in \mathcal{C} \tag{12}$$

$$\sum_{i\in\mathcal{C}} d_i \sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{T}} x_{ijkt} \leq q \qquad \forall k \in \mathcal{V} \tag{13}$$

$$\sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{T}} x_{0jkt} = 1 \qquad \forall k \in \mathcal{V} \tag{14}$$

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}} x_{ihkt} - \sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{T}} x_{hjkt} = 0 \qquad \forall h \in \mathcal{C}, \forall k \in \mathcal{V} \tag{15}$$

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}} x_{i,n+1,kt} = 1 \qquad \forall k \in \mathcal{V} \tag{16}$$

$$\sum_{k\in\mathcal{V}}\sum_{t'=t,\ldots,\beta_0} x_{ijkt'} + \sum_{h\in\mathcal{N}}\sum_{k\in\mathcal{V}}\sum_{t'\in\mathcal{T}^{\tau}_{tij}} x_{jhkt'} \leq 1 \qquad \forall i,j \in \mathcal{N}, \forall t \in \mathcal{T} \tag{17}$$

$$\sum_{h\in\mathcal{N}}\sum_{k\in\mathcal{V}}\sum_{t'=t,\ldots,\beta_0} x_{ihkt'} + \sum_{h\in\mathcal{N}}\sum_{k\in\mathcal{V}}\sum_{t'\in\mathcal{T}^{\delta}_{tij}} x_{jhkt'} \leq 1 \qquad \forall (i,j) \in \Delta, \forall t \in \mathcal{T} \tag{18}$$

$$x_{ijkt} = 0 \qquad \forall i \in \mathcal{C}, j \in \mathcal{N}, \forall k \in \mathcal{V}, \forall t \in \{\alpha_0, \ldots, \alpha_i - 1\} \cup \{\beta_i + 1, \ldots, \beta_0\} \tag{19}$$

$$x_{ijkt} \in \{0,1\} \qquad \forall i,j \in \mathcal{N}, \forall k \in \mathcal{V}, \forall t \in \mathcal{T} \tag{20}$$

Constraints (11)–(16) are similar to Constraints (1)–(6), where we now sum over the time index as well. Constraints (17) provide the required travel time between customers. If any vehicle goes directly from customer i to customer j, and if customer i is scheduled at time t or later, then j cannot be scheduled at time $t + \tau_{ij} - 1$ or earlier. The strength of this model lies in the formulation of generalized precedence constraints (18). Constraints (18) are the equivalent of Constraints (10) of the former model. Similarly to the former constraints, these constraints state that if customer i is scheduled anywhere from time t and onward, then customer j is not scheduled before time $t + \delta_{ij}$. This is valid for all $t \in \mathcal{T}$. Constraints (19) enforce the time windows and (20) are the integrality constraints.

3. DECOMPOSITION

As described earlier, Dantzig–Wolfe decomposition has been very successful in exact optimization of VRPTW. The decomposition splits the problem into a set-partitioning master problem and a resource constrained shortest path subproblem. See, for example, Kallehauge et al. [24] for a thorough exposition. In the traditional VRPTW formulation, Constraints (2) are the only constraints that link the vehicles. Without these, we can solve the problem separately for each vehicle. Hence, the problem is split into a subproblem, where feasible routes are generated and a master problem, where these routes are combined.

3.1. Master Problem

We propose four applicable formulations of the master problem and rank them according to the tightness with which they describe the solution space.

3.1.1. Direct Formulation. The introduced generalized precedence constraints apply to routes from separate vehicles, and hence, these will be part of the new master problem. In the full master problem, we have the set of all feasible routes, $\mathcal{R}$. Each route has a cost of c_r and is defined by the customers visited and the time of each such visit, described by two parameters, a_i^r and s_i^r. For each route, r, and each customer, i, if customer i is in the route r, we set $a_i^r = 1$ and set s_i^r equal to the time of that visit. If the customer is not in the route, $a_i^r = 0$ and $s_i^r = 0$. In column generation, the variables of the master problem are generated iteratively and the set of variables available in a specific iteration is denoted $\mathcal{R}'$. Decision variables for the master problem are denoted λ_r, with $\lambda_r = 1$, if route r is used, and $\lambda_r = 0$, otherwise. The Linear Programming (LP) relaxation of the master problem defined by a subset of the decision variables, $\mathcal{R}'$, is denoted the restricted master problem and is formulated below. The master problem is obtained by decomposing the compact direct formulation (1)–(10).

$$\min \sum_{r\in\mathcal{R}'} c_r\lambda_r \tag{21}$$

$$\sum_{r\in\mathcal{R}'} a_i^r\lambda_r = 1 \qquad \forall i \in \mathcal{C} \tag{22}$$

$$\sum_{r\in\mathcal{R}'} s_i^r\lambda_r + \delta_{ij} \leq \sum_{r\in\mathcal{R}'} s_j^r\lambda_r \qquad \forall (i,j) \in \Delta \tag{23}$$

$$\lambda_r \geq 0 \qquad \forall r \in \mathcal{R}' \tag{24}$$

The corresponding subproblem is that of generating negative reduced cost routes for the master problem (21)–(24). In this context, we refer to the model as the direct formulation. The main disadvantage of the model is that it introduces linear time costs in the subproblem, namely the dual variables of Constraints (23). Hence, the subproblem is a resource constrained shortest path problem with linear node costs. Another issue is that s_i^r is a nonbinary parameter, and the introduction of nonbinary parameters in the master problem is usually a feature that leads to highly fractional solutions.

3.1.2. Time-Indexed Formulation. In the time-indexed formulation, the master problem contains only binary parameters. Constraints (12) and (18) link the vehicles and must therefore remain in the master problem. The parameters of the time-indexed master problem are defined as $a_{it}^r = 1$ if customer i is scheduled at time t in route r, and $a_{it}^r = 0$ otherwise. The decision variable λ_r has the same definition as in

the previous model. The relation to the decision variables of model (11)–(20) is: $\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}} x_{ijkt} = \sum_{r\in\mathcal{R}'} a^r_{it}\lambda_r, \forall i \in \mathcal{C}, \forall t \in \mathcal{T}$. The restricted master problem of the time-indexed formulation is:

$$\min \sum_{r\in\mathcal{R}'} c_r\lambda_r \tag{25}$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} a^r_{it'}\lambda_r = 1 \qquad \forall i \in \mathcal{C} \tag{26}$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'=t,\ldots,\beta_0} a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}^{\delta}_{ij}} a^r_{jt'}\lambda_r \le 1 \qquad \forall (i,j)\in\Delta, \forall t\in\mathcal{T} \tag{27}$$

$$\lambda_r \ge 0 \qquad \forall r \in \mathcal{R}' \tag{28}$$

The obvious problem with the time-indexed restricted master problem (25)–(28) is the number of constraints of type (27). The scheduling horizon is usually large enough to make this model intractable in realistic problems. The subproblem is a resource constrained shortest path problem with time-dependent costs. The costs may be different for each time step. This is very unlikely, however. Most of the constraints of type (27) will be nonbinding and this leaves the corresponding dual variables equal to 0. For the same reason, we may choose to introduce them, only when they become violated.

3.1.3. Relaxed Formulation. A third way of approaching the problem is to simply disregard the temporal dependencies in the master problem. The dependencies must then be enforced by the branching scheme. This approach is used for synchronization by Dohn et al. [15] and Bredström and Rönnqvist [4] and for generalized precedence constraints by Justesen and Rasmussen [23]. It leaves the following master problem, which is identical to the master problem of the VRPTW decomposition. Here, we refer to it as the relaxed formulation.

$$\min \sum_{r\in\mathcal{R}'} c_r\lambda_r \tag{29}$$

$$\sum_{r\in\mathcal{R}'} a^r_i\lambda_r = 1 \qquad \forall i \in \mathcal{C} \tag{30}$$

$$\lambda_r \ge 0 \qquad \forall r \in \mathcal{R}' \tag{31}$$

3.1.4. Limited Time-Indexed Formulation. In the time-indexed formulation (25)–(28), it is possible to include only a subset of Constraints (27) and this idea is implemented in a limited version of the time-indexed formulation. The formulation can be seen as a hybrid of the time-indexed formulation and the relaxed formulation. Obviously, all generalized precedence constraints must be respected in a feasible solution. Therefore, if a violation occurs for a generalized precedence constraint, which is not in the subset of included constraints, the constraint is instead enforced by branching, like in the relaxed formulation.

In our case, we have chosen to define the subset of generalized precedence constraints dynamically. More specifically, we only add cuts if they are maximally violated, that is, if the left hand side of constraint (27) is equal to 2. When a cut has been added it stays in the model. Smaller violations are handled by the branching scheme. How to identify violated constraints is described in more detail in section 3.2.

3.1.5. Strength of the Formulations. The relaxed formulation is obviously a relaxation of both the direct formulation, the full time-indexed formulation, and the limited time-indexed formulation. An interesting result is that the direct formulation is also a relaxation of the time-indexed formulation, and we are hence able to rank the models according to their strength.

Proposition 1 (The time-indexed master problem formulation is a stronger formulation than the direct master problem formulation)**.**

Proof. In the following, we assume that we have a solution to (25)–(28) and prove that the solution is also feasible for Constraints (21)–(24). For all problems with a feasible solution, it holds that $\alpha_0 + \delta_{ij} - 1 \le \beta_0, \forall (i,j) \in \Delta$ and hence a special case of (27) with $t = \alpha_0$ is:

$$\sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,\beta_0} a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,\alpha_0+\delta_{ij}-1} a^r_{jt'}\lambda_r \le 1 \qquad \forall (i,j)\in\Delta \tag{32}$$

Using (26) this entails the rather obvious:

$$\sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,\alpha_0+\delta_{ij}-1} a^r_{jt'}\lambda_r = 0 \qquad \forall (i,j)\in\Delta \tag{33}$$

$$\Downarrow$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,t} a^r_{jt'}\lambda_r = 0 \qquad \forall (i,j)\in\Delta, t = \alpha_0,\ldots,\alpha_0+\delta_{ij}-2 \tag{34}$$

$$\Downarrow$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,t} a^r_{jt'}\lambda_r = 1 \qquad \forall (i,j)\in\Delta, t = \alpha_0,\ldots,\alpha_0+\delta_{ij}-2 \tag{35}$$

Summing Constraints (26), (35), and (27) over t, we get the following for $(i,j) \in \Delta$:

$$\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} a^r_{it'}\lambda_r = 1$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,t} a^r_{jt'}\lambda_r = 1 \quad t=\alpha_0,\ldots,\alpha_0+\delta_{ij}-2$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'=t,\ldots,\beta_0} a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,\min\{\beta_0,t+\delta_{ij}-1\}} a^r_{jt'}\lambda_r \le 1 \quad t=\alpha_0,\ldots,\beta_0$$

$$\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} (t'-\alpha_0+1+\delta_{ij})a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} (\beta_0+\delta_{ij}-t')a^r_{jt'}\lambda_r \le \beta_0-\alpha_0+\delta_{ij}+1$$

Therefore, for any feasible solution of (25)–(28), we have for $(i,j) \in \Delta$:

$$0 \le \beta_0-\alpha_0+\delta_{ij}+1-\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}}(t'-\alpha_0+1+\delta_{ij})a^r_{it'}\lambda_r - \sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}}(\beta_0+\delta_{ij}-t')a^r_{jt'}\lambda_r \tag{36}$$

$$=\beta_0-\alpha_0+\delta_{ij}+1-\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} t'a^r_{it}\lambda_r - (-\alpha_0+1+\delta_{ij})\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} a^r_{it'}\lambda_r - (\beta_0+\delta_{ij})\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} a^r_{jt'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} t'a^r_{jt'}\lambda_r \tag{37}$$

$$=\beta_0-\alpha_0+\delta_{ij}+1-\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} t'a^r_{it'}\lambda_r+\alpha_0-1-\delta_{ij}-\beta_0-\delta_{ij}+\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} t'a^r_{jt'}\lambda_r \tag{38}$$

$$=\sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} t'a^r_{jt'}\lambda_r - \sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}} t'a^r_{it'}\lambda_r - \delta_{ij} \tag{39}$$

$$=\sum_{r\in\mathcal{R}'} s^r_j\lambda_r - \sum_{r\in\mathcal{R}'} s^r_i\lambda_r - \delta_{ij} \tag{40}$$

The result in (38) is based on (26). The final result in (40) comes from the following relation between the parameters of the models (21)–(24) and (25)–(28): $s^r_i = \sum_{t\in\mathcal{T}} t a^r_{it}$. The result in (40) proves that any feasible solution of (25)–(28) also respects (23). (22) is trivially respected as $a^r_i = \sum_{t'\in\mathcal{T}} a^r_{it'}$, and hence (21)–(24) is a relaxation of (25)–(28).

To illustrate that the two formulations are not equally strong, we consider the following small example. Take two customers $i=1$ and $j=2$ with $\delta_{12}=2$. Three simple routes cover these two customers with $a^1_1=1$, $a^2_2=1$, $a^3_2=1$ and $s^1_1=1$, $s^2_2=2$, $s^3_2=4$ for model (21)–(24). In model (25)–(28) this corresponds to $a^1_{11}=1$, $a^2_{22}=1$, $a^3_{24}=1$. A solution with $\lambda_1=1$, $\lambda_2=0.5$, $\lambda_3=0.5$ is feasible in (21)–(24) but not in (25)–(28). This is verified by inspecting (23) for $i=1, j=2$:

$$\sum_{r\in\mathcal{R}'} s^r_1\lambda_r + \delta_{12} \le \sum_{r\in\mathcal{R}'} s^r_2\lambda_r \Rightarrow 1+2\le 3$$

and (27) for $i=1, j=2, t=1$:

$$\sum_{r\in\mathcal{R}'}\left(a^r_{11}\lambda_r + a^r_{12}\lambda_r + a^r_{13}\lambda_r + a^r_{14}\lambda_r\right) + \sum_{r\in\mathcal{R}'}\left(a^r_{21}\lambda_r + a^r_{22}\lambda_r\right) = 1+0.5 \not\le 1$$

■

Using the above result, we can conclude that the full time-indexed formulation is a stronger formulation than the direct formulation. The direct formulation in turn is stronger than the relaxed formulation. In the same way, we also know that the full time-indexed formulation is a stronger formulation than the limited time-indexed formulation, which is stronger than the relaxed formulation. It is not possible to rank the direct formulation and the limited time-indexed formulation.

A nice property of the time-indexed model is that it has only been relaxed with respect to integrality and this means

that if we can restore integrality, we have a feasible solution. In this article, we will only consider branching to restore integrality, and hence the advantage may not seem immediate. For VRPTW, a significant amount of work has been done on cut generation to remove fractional solutions. Such cuts could be added to the time-indexed model of VRPTWTD as well, and this may restore integrality without the need of branching. See, for example, the work of Kohl et al. [26], Cook and Rich [9], Lysgaard et al. [31], and Jepsen et al. [22] for more on the subject.

3.2. Identifying Violated Cuts

As described earlier, the generalized precedence constraints (27) of the time-indexed master problem (25)–(28) are only represented implicitly. The constraints are added as cuts, as they become violated. The constraint is repeated below.

$$\sum_{r\in\mathcal{R}'}\sum_{t'=t,\ldots,\beta_0} a^r_{it'}\lambda_r + \sum_{r\in\mathcal{R}'}\sum_{t'\in\mathcal{T}^\delta_{ij}} a^r_{jt'}\lambda_r \le 1$$

$$\forall (i,j)\in\Delta, \forall t\in\mathcal{T} \quad (27)$$

In theory, we have to check for violations for all $t \in \mathcal{T}$, but actually it is possible to do with significantly less. As a^r_{it} is a binary parameter and $\lambda_r \ge 0$, the sum $\sum_{r\in\mathcal{R}'}\sum_{t'=t,\ldots,\beta_0} a^r_{it'}\lambda_r$ is nonincreasing for increasing t. Correspondingly, the sum $\sum_{r\in\mathcal{R}'}\sum_{t'=\alpha_0,\ldots,\min\{\beta_0,t+\delta_{ij}-1\}} a^r_{jt'}\lambda_r$ is nondecreasing. Constraints (27) are never violated for $t = \alpha_0$ as such violations are prevented by preprocessing the time windows, see section 4.1. Therefore, for customer i, we only need to check for violations with any t where $\exists r \in \mathcal{R}' : a^r_{it}\lambda_r > 0$, that is, any point in time where customer i is scheduled (possibly with a fractional value). It is easy to generate a list of all t where $\exists r \in \mathcal{R}' : a^r_{it}\lambda_r > 0$, by running through the routes of all variables with positive values and registering the time of service for each customer. By separating cuts as described, we are not adding all violated cuts, but we are sure to add at least one cut for each customer, if any cuts are violated for that customer.

3.3. Subproblem

The subproblem of the Dantzig–Wolfe decomposition of VRPTW can be solved as an ESPPTWCC. Any feasible solution of the subproblem with negative cost represents a column with negative reduced cost in the master problem and may therefore enter the basis. The subproblem consists of Constraints (3)–(9). The variables are defined as in the compact formulation, but now for the single vehicle under consideration, that is, the index k has been removed. The objective function of the subproblem becomes:

$$\min \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} (c_{ij} - \pi_i)x_{ij} \quad (41)$$

$\pi_i, i \in \mathcal{N}$, are the dual variables of Constraints (30) of the VRPTW master problem. Dror [16] proves that ESPPTW is NP-hard in the strong sense and hence no pseudo-polynomial algorithms are likely to exist. The subproblem is usually solved with a dynamic label setting algorithm. Desrochers et al. [13] presented a dynamic algorithm for the nonelementary version of the subproblem. This algorithm was adjusted to handle the elementary problem by Feillet et al. [17] and superior results based on this method have been presented recently, see, for example, Desaulniers et al. [12]. The idea in the label setting algorithm is to represent partial paths by labels. Given a label for some partial path, it is possible to expand the path by creating new labels in nodes that can possibly extend the current partial path. The length of the path is increased by one, and the process continues iteratively.

The subproblem of the direct formulation must consider the dual variables of Constraints (22), π_i, and additionally the dual variables of Constraints (23), σ_{ij}, and the objective function becomes:

$$\min \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} (c_{ij} - \pi_i)x_{ij} - \sum_{(i,j)\in\Delta} \sigma_{ij}s_i + \sum_{(j,i)\in\Delta} \sigma_{ji}s_i \quad (42)$$

As described previously, the subproblem is now a resource constrained shortest path problem with linear node costs, which makes it much harder to solve. Ioachim et al. [21] describe a dynamic algorithm to solve the acyclic version of this problem. A similar cyclic problem is solved as a subproblem by Christiansen and Nygreen [7].

The subproblem of the time-indexed formulation has the following objective function which we split in three parts for easy reference:

$$\min \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} (c_{ij} - \pi_i)\sum_{t\in\mathcal{T}} x_{ijt} \quad (43a)$$

$$- \sum_{(i,j)\in\Delta}\sum_{t\in\mathcal{T}}\sum_{t'=\alpha_0,\ldots,t} \rho_{ijt'}x_{ijt} \quad (43b)$$

$$- \sum_{(j,i)\in\Delta}\sum_{t\in\mathcal{T}}\sum_{t'=\max\{\alpha_0,t-\delta_{ji}+1\},\ldots,\beta_0} \rho_{jit'}x_{ijt} \quad (43c)$$

where π_i are the dual variables of Constraints (26) and ρ_{ijt} are the nonpositive dual variables of Constraints (27). In the worst case, this objective function introduces a distinct cost for each time step. In a label setting algorithm, this means that we have to create a label for each time step and hence the number of labels explodes immediately. In practice, only a few constraints of type (27) are binding and therefore only few ρ_{ijt} have nonzero values.

The idea in the basic label setting algorithm is to create and keep only labels that are not dominated by better labels. With the objective function (43), we have a lot of potential labels. It is, however, only an advantage to postpone a visit, if it can possibly decrease the objective value. As $\rho_{ijt} \le 0$ and $x_{ijt} \in \{0, 1\}$, adjusting time for a certain visit can only decrease the objective, if it removes terms in (43b) or (43c). (43a) is neutral to service time of customers, that is, for a given transition (i, j) the contribution to the objective function coefficient of x_{ijt} is the same for all $t \in \mathcal{T}$. The smallest contribution from (43b) is

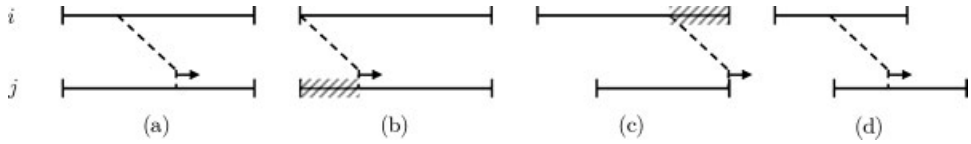

FIG. 2. Time window reductions. (a) The original time windows. (b) Using the generalized precedence constraint, the first part of the time window of customer j is removed. (c) In a similar way, the last part of the time window of customer i is removed. (d) The time windows after the reduction.

obtained by the smallest possible value of t as $\sum_{t'=\alpha_0,\ldots,t} \rho_{ijt'}$ is nonincreasing over t for given i,j. Therefore, for customer i, we need a label for the earliest possible time t^0. Only the value of (43c) will decrease as t is increased (for given i,j) and only when terms with $\rho_{jit'} < 0$ are excluded. The value of (43c) for t is lower than the corresponding sum for $t-1$ when $\rho_{ji(t-\delta_{ji})} < 0$, that is:

$$\rho_{ji(t-\delta_{ji})} < 0 \Rightarrow \sum_{t'=\max\{\alpha_0,t-\delta_{ji}\},\ldots,\beta_0} \rho_{jit'} < \sum_{t'=\max\{\alpha_0,t-\delta_{ji}+1\},\ldots,\beta_0} \rho_{jit'}$$

The full objective function possibly decreases for such t, and hence, we need one label for each $t \in \{t^0+1,\ldots,\beta_i\}$, where $\exists(j,i) \in \Delta : \rho_{ji(t-\delta_{ji})} < 0$. For all other potential labels, there will always be a label earlier in time with the same or less cost.

A small improvement, that we found to have a significant effect, is to include knowledge of mutually exclusive customers. Some temporal dependencies like, for example, synchronization and overlap make it impossible to include both customers in the same route. In these methods, such a restriction is imposed in the master problem or in the branching scheme. Hence, routes could be generated that would never occur in a feasible solution. By excluding the occurrence of mutually exclusive customers in all routes generated in the subproblem, the LP-bounds get stronger and as a consequence the algorithm is more efficient. The dominance scheme in the subproblem solver is also modified to utilize this knowledge. By visiting one of two mutually exclusive customers, the other becomes unreachable. Hence, by updating the set of unreachable customers appropriately, two labels which have visited two different mutually exclusive costumers can still be compared in the dominance check, as their possible extensions are identical.

4. BRANCHING

The master problem models presented in the previous section are relaxations. Therefore, we may need to apply branch-and-bound to get to a feasible solution. The λ-variables of the master problems were introduced as binary variables, but the integer property has been relaxed to allow solution by an LP-solver. Therefore, integrality needs to be restored by branching. A lot of work has already been done for VRPTW in this respect. See, for example, Kallehauge et al. [24] for a review. In the relaxed formulation, the generalized precedence constraints have also been relaxed. Therefore, in this model, we need a branching method that will also restore feasibility with respect to temporal dependencies.

Gélinas et al. [19] proposed to branch on time variables to arrive at integer-feasible solutions. This type of branching was also used to enforce synchronization by Ioachim et al. [20], Dohn et al. [15], and Bredström and Rönnqvist [4], and for general temporal dependencies by Justesen and Rasmussen [23]. Time window branching is not complete with respect to integer feasibility and hence has to be complemented by another branching scheme, for example, traditional flow variable branching.

4.1. Time Window Reduction

Before describing the actual branching scheme, we introduce a simple reduction technique based on the generalized precedence constraints. For any two customers, i and j with $(i,j) \in \Delta$, it is possible to reduce the time windows as follows:

	Customer i	Customer j
Old time windows	$[\alpha_i, \beta_i]$	$[\alpha_j, \beta_j]$
New time windows	$[\alpha_i, \min\{\beta_i, \beta_j - \delta_{ij}\}]$	$[\max\{\alpha_j, \alpha_i + \delta_{ij}\}, \beta_j]$

These reductions are illustrated in Figure 2. The reductions are used to preprocess the time windows and may also be used anywhere in the branching tree. This technique is essential when applying time window branching.

4.2. Time Window Branching

In a feasible solution of VRPTWTD, all visits are scheduled at exactly one point in time and all generalized precedence constraints are respected. In the relaxed formulation, a solution may be integer feasible, but could still violate precedence constraints. In the direct formulation and the time-indexed formulation, an integer feasible solution will also respect precedence constraints. As for VRPTW, we may still use time window branching to get integral solutions. In the following, we use the relaxed formulation as a basis for introducing time window branching, but it transfers easily to the other models.

Figure 3 shows a violation of the precedence constraint between customers j and i, in routes r_1 and r_2. By branching

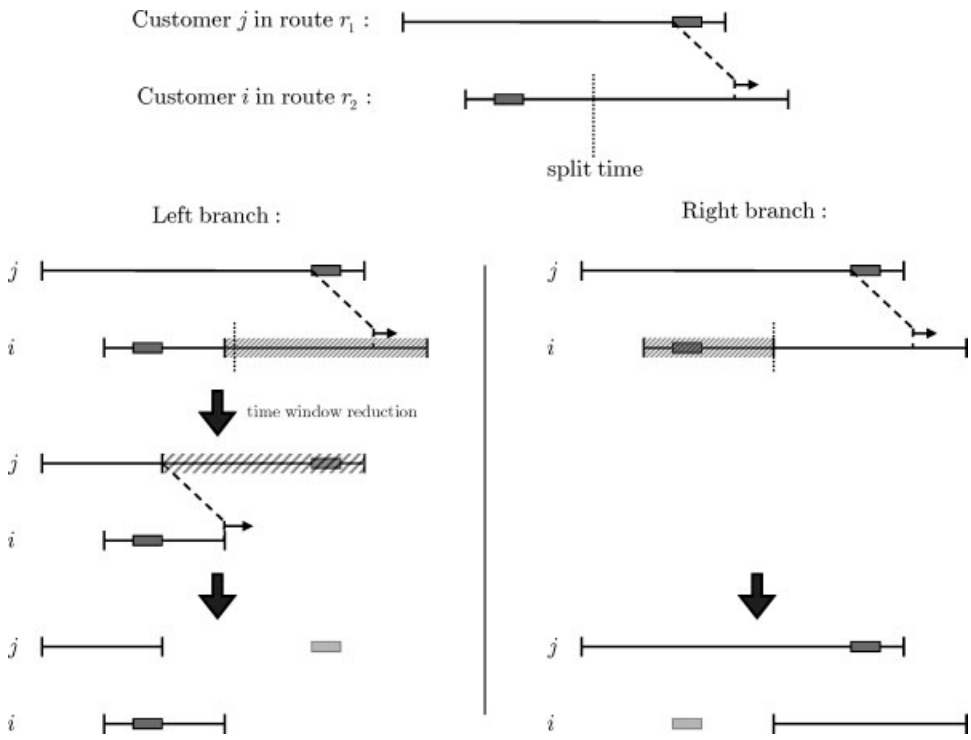

FIG. 3. Branching to avoid a violation of a precedence constraint.

on the time window of customer i and using the time window reduction rule of section 4.1 for (j, i), r_1 and r_2 are prohibited in the left and right branch, respectively. Note that there is no overlap between the time window of customer i in the left branch and the corresponding time window in the right branch.

Later, we describe a strategy to wisely select the point in time, t^s, where the time window is split. If $s_i + 1 \leq t^s \leq s_j + \delta_{ji}$, the current solution will be excluded from the solution space by using the reduction rule for (j, i). The modified time windows of customer i become: $[\alpha_i, t^s - 1]$ in the left branch and $[t^s, \beta_i]$ in the right branch.

The tightest formulation is reached if time windows are reduced as much as possible. Therefore in both branches, we run through all relevant precedence constraints with the new time window of customer i and reduce time windows where possible. This may also reduce the time windows of other customers than i and j, and this process is repeated iteratively, until no further reduction is possible.

An interesting result is that this branching strategy is as strong as the one formerly proposed specifically for synchronization, by, for example, Ioachim et al. [20]. In the less general context, the time windows of two synchronized customers are, naturally, always identical. Branching is done on the two time windows simultaneously, so they always stay identical. Synchronization modeled by two generalized precedence constraints, also has this property when time window reductions are applied. Assume that we have a synchronization constraint between customers i and j, that is, $\delta_{ij} = 0$ and $\delta_{ji} = 0$ and hence $\alpha_j = \alpha_i \wedge \beta_j = \beta_i$. For a given split time t^s for customer i, the time windows become:

	Customer i	Customer j
Old time windows	$[\alpha_i, \beta_i]$	$[\alpha_i, \beta_i]$
TW (left branch)	$[\alpha_i, t^s - 1]$	$[\alpha_i, \min\{\beta_i, t^s - 1 - \delta_{ji}\}]$ $= [\alpha_i, t^s - 1]$
TW (right branch)	$[t^s, \beta_i]$	$[\max\{\alpha_i, t^s + \delta_{ij}\}, \beta_i]$ $= [t^s, \beta_i]$

This is illustrated in Figure 4. The time windows of i and j are identical in each of the branches after applying time window reduction.

Usually, there are several branching candidates to choose from and we need a strategy to choose one of these. Gélinas et al. [19] elaborate further on this subject. When using strong branching (see, e.g., Ref. [1]) a few candidates are chosen for further probing. In any case, we need to specify a priority ordering of candidates. First, we need to find the potential branching candidates. In theory, we could branch on any time window and split it at an arbitrary position. In practice, however, we limit this choice. We do not want to consider candidates where the branching is without effect in one of the branches, that is, where one of the branches does not prohibit any columns of the current solution. Also, many of the remaining candidates have an identical effect on the current solution. They may still have a different effect on new columns, but it is very hard to predict this impact. Figure 5a depicts some of the potential branching candidates in the time window of customer i. Customer i is a part of two routes that have been included in the solution with fractional values and hence it appears at multiple positions within its own time

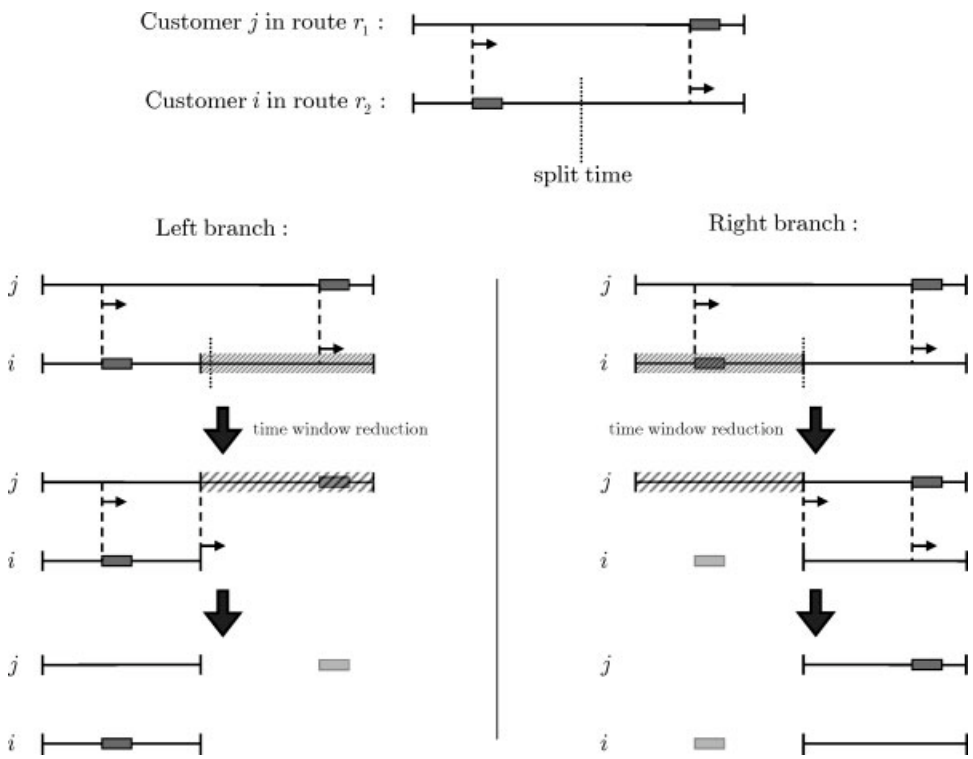

FIG. 4. Branching on a generalized precedence constraint of a synchronization constraint.

window. In this example we assume that the routes r_2 and r_3 are both in the solution with a value of 0.5 and r_1 and r_4 with a value of 1. The effect on the current solution of each candidate is shown in Table 2. Candidates 1 and 6 are examples of ineffective candidates. Candidates 2 and 3 have an identical effect on the current solution.

In our approach, when choosing between candidates with an identical immediate effect, it is optimal to select the candidate which splits at the latest possible time. For customer i, that split time coincides with either s_i^r or with $s_j^r + \delta_{ji}$ for a route r, which is in the current basis of the master problem.

In Figure 5a, we prefer candidate 3 to candidate 2 as there is less chance that r_2 can be adapted to the new time window of the right branch. We prefer candidate 4 to candidate 3 as it excludes the same or more in both branches. Figure 5b shows the three candidates that we would actually consider for the time window of customer i. The candidates $1'$, $4'$, and $5'$ get the same values as 1, 4, and 5, respectively, in Table 2. Remember that these are just the candidates of customer i. There will be similar candidates for each of the other customers. We find the candidates for customer i by running through all routes that are included in the solution with a positive value. If customer i is in the route, the start time of the customer is a candidate ($t^s = s_i$). If the route includes a customer j, where $(j, i) \in \Delta$ the route contributes with a candidate for customer i with split time $t^s = s_j + \delta_{ji}$.

The preference of late split times is due to the following algorithmic considerations: the label setting algorithm, which is used to generate routes, schedules visits at the earliest possible time in any route. Hence, it is impossible to schedule the visit earlier without rerouting. In the left branch, the service

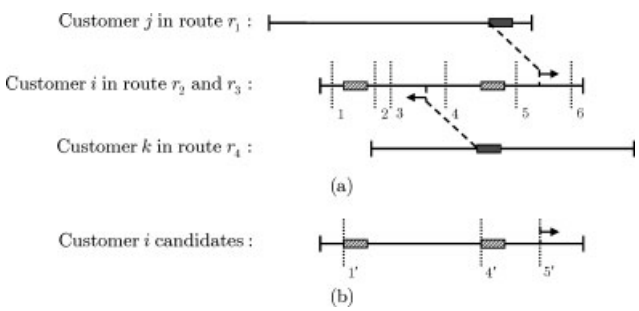

FIG. 5. Some potential branching candidates in the time window of customer i.

TABLE 2. Effect of the branching candidates of Figure 5.

	Infeasible routes		Sum of excluded variables		
	Left branch	Right branch	Left branch	Right branch	Preference
Candidate 1	r_1, r_2, r_3		2	0	0
Candidate 2	r_1, r_3	r_2	1.5	0.5	0.5
Candidate 3	r_1, r_3	r_2	1.5	0.5	0.5
Candidate 4	r_1, r_3	r_2, r_4	1.5	1.5	1.5
Candidate 5	r_1	r_2, r_3, r_4	1	2	1
Candidate 6		r_2, r_3, r_4	0	2	0

time of a customer will be forced to decrease by at least one time unit and hence rerouting is required. In the right branch, the current conflict is resolved, as the start of the new time window is equal to s_i^r or $s_j^r + \delta_{ji}$ which generated the branching candidate in the first place.

In this article, the problems are solved to optimality, which means that every node in the branch-and-bound tree must be either explored or pruned. Hence, we aim for a small, but at the same time, balanced tree. To achieve this, we rank the branching candidates according to the corresponding sums of excluded variables in the two branches. A candidate gets the value of the minimum of the two sums, and hence only the worst of the branches counts. A large value of a candidate is equal to a high preference. Hence, we prefer branching candidates which exclude as much as possible in the least effective branch. In the example, candidate 4 is preferred, as it excludes 1.5 routes in each branch, giving it a preference ranking of 1.5.

If the aim is to get high-quality solutions, but not necessarily optimal solutions, in a short time, it may be better to choose branching candidates where one of the branches is more promising than the other. This may then be utilized in a heuristic search of the branch-and-bound tree. This idea has been used in several other contexts, see, for example, Ryan [34].

5. BENCHMARK INSTANCES

A set of benchmark instances has been used in the following quantitative analysis of the problem and in a comparison of the different models. The instances are extensions of the 56 well-known VRPTW-instances of Solomon [36]. Solomon's VRPTW-instances have been used extensively in existing literature and new solution algorithms for VRPTW are often tested on these to indicate algorithm performance. The data sets are well suited for the tests, as they represent a wide range of problems with varying structure. The locations of customers are in some instances uniformly distributed over whole area. In others, customers are located in clusters. The time windows of customers are also varied to test both very tight and very loose time window constraints. The data sets consist of a number of customers with a geographical location, a time window, service time, and a demand along with the number of available vehicles, their capacity, and the scheduling horizon. The instances are publicly available. We take the original instances and introduce temporal dependencies of various types to these instances. We have chosen to look at the instances with 25 and 50 customers, as these are small enough to allow quick solution of the basic problem. Some of them still prove hard to solve as temporal dependencies are introduced. A thorough analysis is performed on the instances with 25 customers, as most of these can be solved within 1 h. Tests on instances with 50 customers are also included, to assess how the findings for 25 customers scale to larger instances.

Five sets of instances were made: one for each of the five temporal dependencies of Figure 1, except maximum difference, and one set with a random combination of the other four (random combination). The reason for omitting maximum difference is its similarity to minimum difference. When we generate instances randomly, the two types are actually identical.

For all instances, a list of temporal dependencies is created for each of the five types. All random values are drawn from uniform distributions, and the list is generated randomly in the following way:

1. Determine the type of the next temporal dependency to be added (for random combination this choice is random, and for all other types it is fixed).
2. Choose, at random, two visits, i and j, which are not already directly or indirectly interdependent. Visits are indirectly interdependent if there is a chain of dependencies from one to the other, for example, if they both have a dependency on a certain visit, but not directly on each other. We require independency between the visits to avoid infeasible cycles of dependencies.
3. Check if it is possible to impose a temporal dependency between i and j. If not possible, go to 2. If it is impossible to add more temporal dependencies of this type, go to 1. If it is impossible to add more temporal dependencies altogether, exit.
4. Draw random values for the temporal dependency. $\text{diff}_{\min}$ and $\text{diff}_{\max}$ are random numbers drawn from all values that do not make the problem infeasible and that impose a constraint which is more strict than that already given by the time windows. For synchronization and overlap all values are fixed.
5. Set values of δ_{ij} and δ_{ji} according to Table 1.
6. Reduce time windows as explained in section 4.1. This is necessary to ensure that all instances are feasible.
7. Go to 1.

As we do not allow cycles of dependencies, it is not possible to add more than $n-1$ temporal dependencies, where n is the number of customers. In some cases, the number may be smaller than this. For example, for the very strict synchronization constraint, it is obviously not possible to impose $n-1$ synchronizations, corresponding to a synchronization of all visits, if some of the visits have nonoverlapping time windows. This results in fewer test instances for some instances sets.

The addition of temporal dependencies to the instance set leads to a very large number of new test instances. For every one of the original instances we have five sets of dependencies and for each of these sets we can choose to use from 0 to $n-1$ dependencies. Hence, for instances with 25 customers, we are able to run 125 tests for each of the original instances (except for a few cases, where it was not possible to generate $n-1$ dependencies). This totals to 7,000 tests and gives a good statistical foundation for the test results presented in the next section. The data files containing the parameters for the randomly generated temporal dependencies are available from the authors on request. We have also generated similar files for instances of size 50 and 100.

TABLE 3. Overview of the test results for instances with 25 customers.

		Time-indexed formulation (all cuts)		Time-indexed formulation (limited)		Relaxed formulation	
	Instances in total	Solved in root	Solved before timeout	Solved in root	Solved before timeout	Solved in root	Solved before timeout
Synchronization	1148	483	1027	448	1141	138	1143
Overlap	1324	351	1058	322	1207	127	1240
Minimum difference	1400	741	1350	703	1377	226	1381
Min + max difference	1400	531	1226	465	1361	105	1384
Random mix	1400	506	1271	459	1382	155	1383

Time out is 1 h.

6. TEST RESULTS

The intention of this section is to give a general overview of the complexity of vehicle routing problems with temporal dependencies. The tests are summarized in graphs that capture the trends we see in the tests overall.

The algorithms are implemented in the branch-and-cut-and-price framework of COIN-OR [8, 30]. The tests have been run on 2.2 GHz AMD (Advanced Micro Devices) processors with 2 GB RAM. Based on preliminary tests, the algorithm is set to do strong branching with three candidates and add up to five variables with negative reduced cost per iteration. For as long as possible, columns are generated by a heuristic version of the label setting algorithm similar to the one proposed by Chabrier [6].

The direct formulation is expected to lead to highly fractional solutions, as generalized precedence constraints can be respected by linearly combining routes, where a particular customer has varying start times. Furthermore, the subproblem is a resource constrained shortest path problem with linear node costs, which is significantly harder to solve, than the other subproblems presented. To our knowledge, no exact solution methods for this problem exist in the literature. The method of Ioachim et al. [21] could probably be adapted to the cyclic case, but the efficiency is questionable. Therefore, the computational experiments of this article do not include the direct formulation.

To give an idea of the overall performance of the remaining three approaches, the test results are summarized in Table 3. As described in section 5, five sets of instances were generated, and the table shows a clear tendency for all five types. The full time-indexed formulation solves the largest number of instances in the root node, as expected. The instances solved in the root node by the relaxed formulation are a subset of those solved in the root node by the limited time-indexed model which again is a subset of those of the full time-indexed formulation. The numbers in the table illustrate this relationship. When looking at the number of instances solved before timeout, the tendency is reversed. The relaxed formulation is capable of solving the largest number of instances for all five types. However, the performance of the limited time-indexed model is in all cases almost as good as that of the relaxed model. Interestingly, the Overlap instances seem harder to solve than the other types.

Table 4 summarizes the results for the 50 customer instances. Incrementing the number of temporal dependencies with one between each test, would result in 14,000 tests, as the temporal dependency generation scheme can generate 49 dependencies for each of the five types for all 56 original instances. To limit the extend of the test, we have chosen to test only for 5, 15, 25, 35, and 45 temporal dependencies. This limits the maximum number of tests to 1,400 in total. As can be observed from Table 4, the results are similar to those of Table 3. In all cases, a smaller ratio of the instances can be solved within an hour and a significant drop, in the number of instances solved in the root, is observed, compared to the instances with 25 customers. Interestingly, the limited time-indexed model now performs slightly better than the relaxed model.

In the remainder of this section, we have chosen to focus on two of the 25 customer instance sets, namely instances

TABLE 4. Overview of the test results for instances with 50 customers.

		Time-indexed formulation (all cuts)		Time-indexed formulation (limited)		Relaxed formulation	
	Instances in total	Solved in root	Solved before timeout	Solved in root	Solved before timeout	Solved in root	Solved before timeout
Synchronization	242	56	158	45	201	5	196
Overlap	276	24	121	23	156	2	150
Minimum difference	280	66	196	58	205	7	202
Min + max difference	280	50	108	44	135	3	134
Random mix	280	33	140	28	189	1	183

Time out is 1 h.

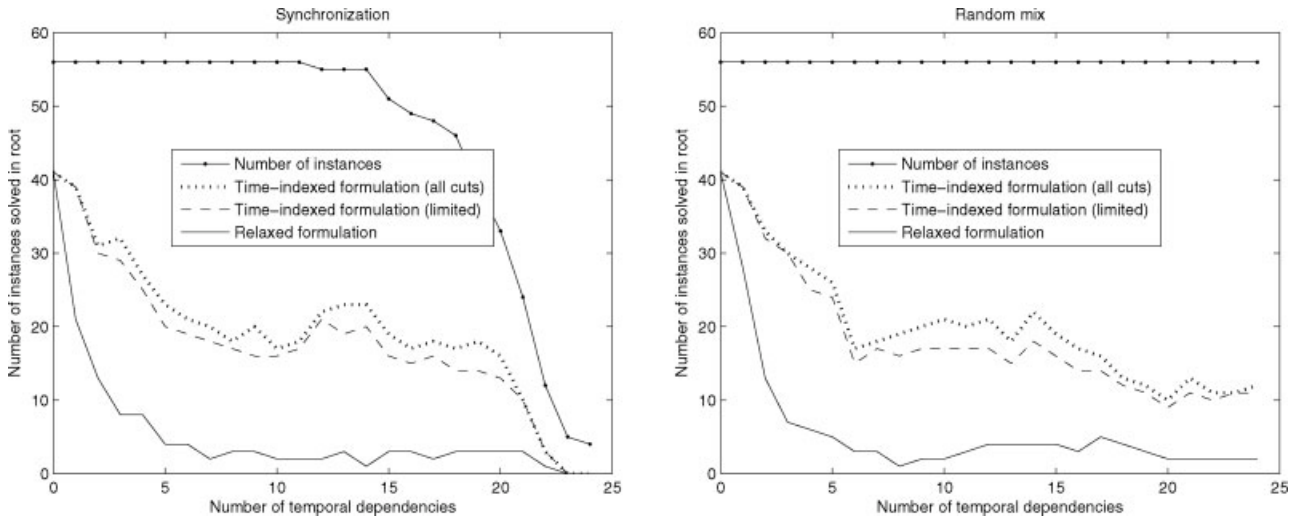

FIG. 6. Number of instances solved at the root node of the branch-and-bound tree.

with only synchronization relations and a set with a random mix of the five temporal dependencies of Table 1. These two sets have been chosen as the first represents a large group of practical applications and the latter does not hold any particular structure. The statements made in the following are in full accordance with the other instance sets.

The root node lower bound sometimes coincides with the value of the optimal solution. In such cases, we often find the optimal solution at the root node. As this results in low computation times, it is interesting to see how often it happens.

In Figure 6, the total number of instances solved at the root node is given, summarized over all 56 instances. We clearly observe the strength of the time-indexed formulation. There is a significant increase in the number of instances solved at the root node compared to the relaxed formulation. Interestingly, there is not much difference from the full formulation to the limited version. In the relaxed formulation, if a problem can be solved at the root node, it means that all temporal dependencies were respected by chance, and hence they would not have been very constraining. Figure 7 gives the number of nodes in the branch-and-bound tree (the mean over all instances) and the conclusions are the same as for Figure 6. As we would expect for the relaxed formulation, we see that the number of nodes increases with the number of temporal dependencies. This does not seem to be the case for the time-indexed formulation.

Another interesting aspect is the solution time. We examine the solution time for each of the instances individually and we also consider the general trend. The variation on solution time is large between the instances. Hence, an average of these values would emphasize the harder instances. We want all instances to count equally and therefore, we normalize the values by comparing each computation time

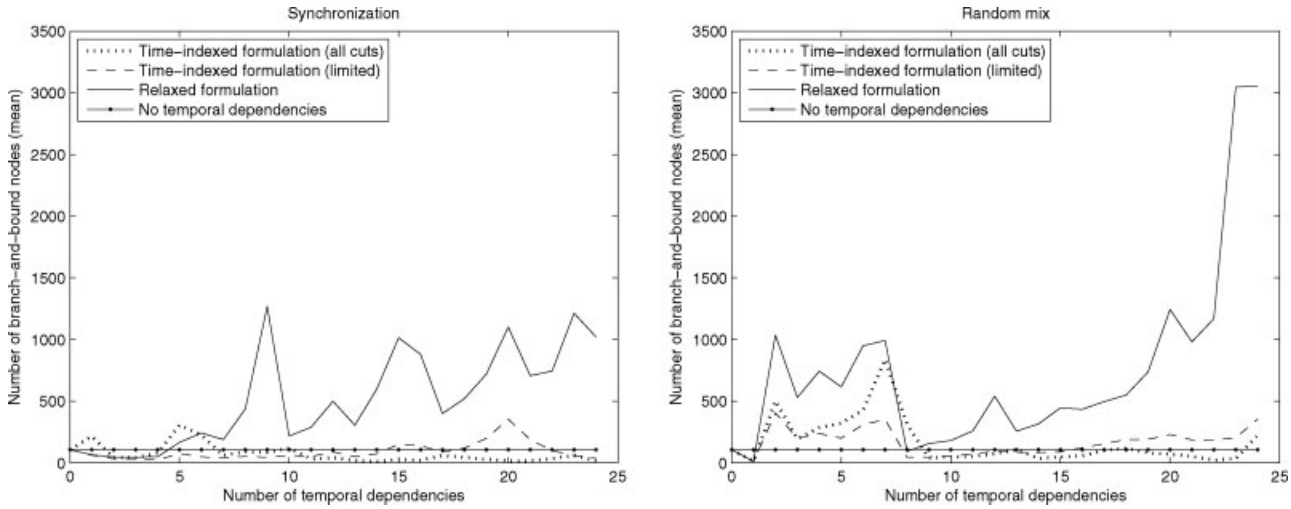

FIG. 7. Number of nodes in the branch-and-bound tree.

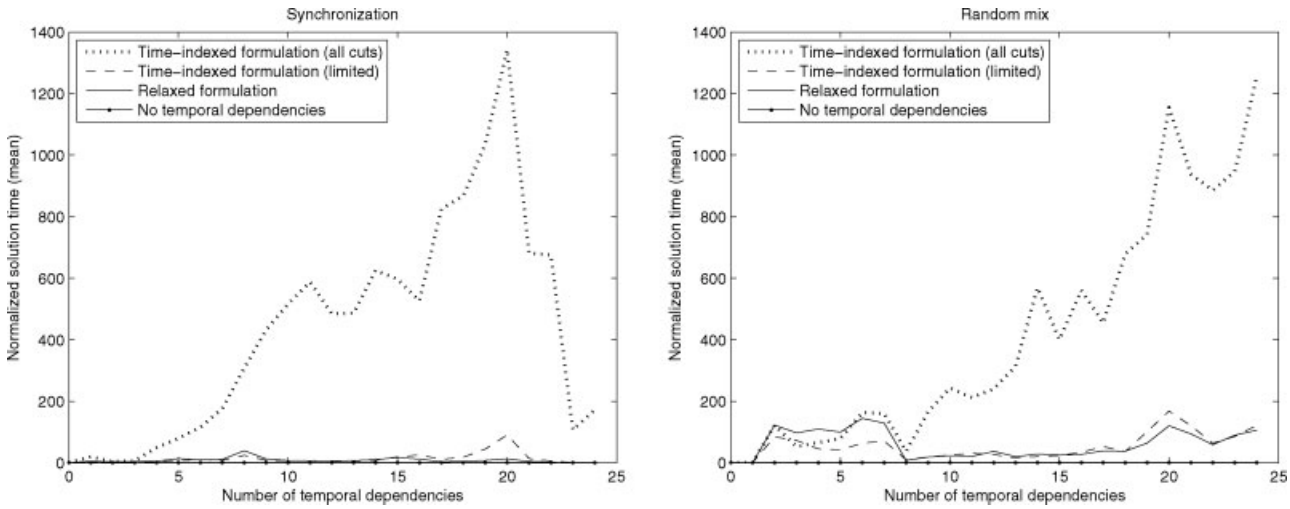

FIG. 8. Normalized solution time (mean).

to the solution time for the same problem without temporal dependencies. The mean over all instances is shown in Figure 8.

Looking at Figure 8, it is clear that the time-indexed formulation is worse than the other two with respect to solution time. Closer inspection shows that the full time-indexed formulation has a few instances where computation time is excessive and this has a major impact on the mean value.

In connection with solution time, it is also interesting to make a direct comparison between the approaches for each instance. For each number of temporal dependencies, we count the number of instances where the limited time-indexed approach is faster than the relaxed formulation and vice versa. The results are summarized in Figure 9. Looking at the instances individually, the limited time-indexed approach seems a little better.

Finally, we look at the distribution of time spent in the algorithm. This is illustrated in Figure 10. The solution procedure in each node of the branch-and-bound tree consists of three parts, namely cut generation, variable (column) generation, and solution of the LP master problem. The times of Figure 10 sum the time spent in all nodes of the tree. The branching time reported is the time used to select branching candidates. As strong branching is applied, this selection also involves the solution of a limited number of LP problems. There may be an overhead on time from memory management, primarily. Therefore, the four components illustrated of the figure do not sum to exactly 100%.

From Figure 10, we observe that for the time-indexed formulation, the portion of time spent in the LP solver increases as problems with more temporal dependencies are considered. This is due to the fact that more cuts are added

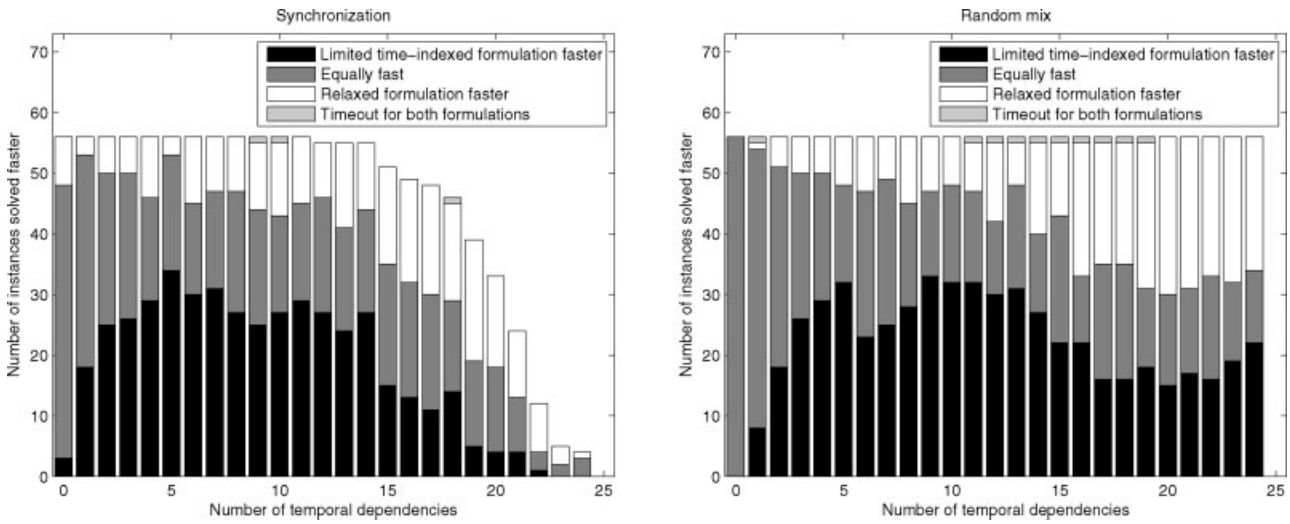

FIG. 9. Number of tests where one of the approaches is faster than the other. The two approaches are considered equally fast if they are within 20% of each other.

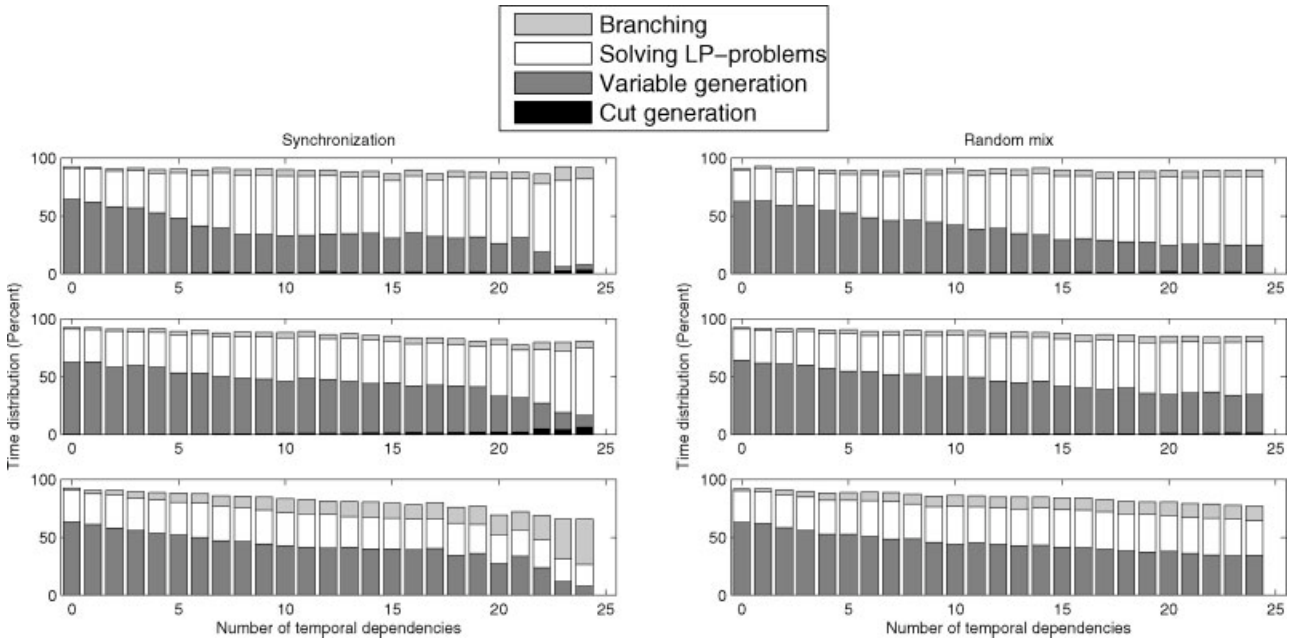

FIG. 10. Distribution of solution time for the time-indexed model (top), the limited time-indexed model (middle), and the relaxed model (bottom).

and hence the size of the LP model increases. For the relaxed formulation, the tendency is, not surprisingly, that more time is spent branching when the number of temporal dependencies increases. The share of time spent by the LP solver is, in this case, stable.

On the basis of the tests, we are able to conclude that the temporal dependencies introduce additional complexity to the problem, as expected. The time-indexed formulation has the worst immediate performance, but may be more useful for large instances with harder pricing problems. The performance of the limited time-indexed approach and the relaxed formulation is comparable. A few instances of each type turn out to be very hard to solve, no matter what method is used. The time-indexed formulation does have a number of nice features that could be utilized in future development. It has tighter bounds, both theoretically and in the practical instances that we have examined. The tighter bounds mean that more instances are solved at the root node of the branch-and-bound tree, and in these cases this formulation gives better results. Also, for instances where the solution is not found at the root node, the branch-and-bound tree is still significantly smaller than the corresponding tree for the relaxed formulation. The number of variables that has to be generated is also generally smaller for the time-indexed formulation. For most realistic problems, variable generation is the dominating factor of the overall solution time, and in these cases the time-indexed formulation may be the better choice, as the LP solution time becomes less important.

7. CONCLUSIONS AND FUTURE WORK

The VRPTWTD has been introduced. The problem has previously been treated in various practical contexts in different forms, but this is the first generic analysis presented in the literature. Four different models were presented and ranked according to their theoretical strength. The time-indexed model has the tightest formulation and hence gives the best bounds, but the number of constraints is too large for them to be included explicitly. Instead, the model was implemented in a branch-and-cut-and-price framework, where both constraints and variables are generated dynamically. As this approach is novel, it was described how to efficiently identify violated cuts and the necessary adjustments in the pricing problem were introduced. The branching scheme was presented next. The scheme is based on the traditional time window branching, where the scheme is also used to restore feasibility with respect to temporal dependencies. The branching scheme is as strong as and more general than the previously presented branching scheme for routing with synchronization. Finally, benchmark instances were introduced and a quantitative analysis was performed.

The analysis showed that, even though the time-indexed model has some nice properties, it also retains its major drawback, namely the number of constraints. As a consequence, a hybrid method was implemented, where only a limited number of the violated cuts are added. This approach kept most of the nice features of the time-indexed model, while at the same time lowering the solution time to the same level as that of the relaxed model.

The model presented in this article is general and is therefore applicable to various practical problems. Future work could be on an adaption to real world problems. Another very interesting direction for future research is to include additional cuts. Using the time-indexed formulation, we were able to solve many instances at the root node of the branch-and-bound tree, and this number could be increased by introducing additional cuts. From, for example, Desaulniers

et al. [12] it is clear that the number of nodes can be limited severely by including cuts, especially for large instances. In many cases, the problems are solved in the root node. The performance of the time-indexed model was clearly better than the relaxed model for the instances, where the optimal solution was obtained at the root node.

Acknowledgments

The authors thank the anonymous referees for constructive comments on an earlier version of this article.

REFERENCES

[1] T. Achterberg, T. Koch, and A. Martin, Branching rules revisited, Oper Res Lett 33 (2005), 42–54.

[2] N. Bélanger, G. Desaulniers, F. Soumis, and J. Desrosiers, Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues, Eur J Oper Res 175 (2006), 1754–1766.

[3] L.-P. Bigras, M. Gamache, and G. Savard, Time-indexed formulations and the total weighted tardiness problem, INFORMS J Comput 20 (2008), 133–142.

[4] D. Bredström and M. Rönnqvist, A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints, Discussion Paper 2007/7, Norwegian School of Economics and Business Administration–Department of Finance and Management Science, Norway, 2007.

[5] D. Bredström and M. Rönnqvist, Combined vehicle routing and scheduling with temporal precedence and synchronization constraints, Eur J Oper Res 191 (2008), 19–31.

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

[7] M. Christiansen and B. Nygreen, "Robust inventory ship routing by column generation," Chapter 7, Column Generation, G. Desaulniers, J. Desrosiers, and M.M. Solomon (Editors), Springer, New York, 2005, pp. 197–224.

[8] Coin, COmputational infrastructure for operations research (COIN-OR). Available at: http://www.coin-or.org/, 2006. Accessed on 10 September 2009.

[9] W. Cook and J.L. Rich, A parallel cutting-plane algorithm for the vehicle routing problem with time windows, Technical report, Rice University, 2001.

[10] E. Danna and C.L. Pape, "Branch-and-price heuristics: a case study on the vehicle routing problem with time windows," Chapter 4, Column Generation, G. Desaulniers, J. Desrosiers, and M.M. Solomon (Editors), Springer, New York, 2005, pp. 99–129.

[11] G.B. Dantzig and P. Wolfe, Decomposition principle for linear programs, Oper Res 8 (1960), 101–111.

[12] G. Desaulniers, F. Lessard, and A. Hadjar, Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows, Transp Sci 42 (2008), 387.

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

[14] K.F. Doerner, M. Gronalt, R.F. Hartl, G. Kiechle, and M. Reimann, Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows, Comput Oper Res 35 (2008), 3034–3048.

[15] A. Dohn, E. Kolind, and J. Clausen, The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach, Comput Oper Res 36 (2009), 1145–1157.

[16] M. Dror, Note on the complexity of the shortest path models for column generation in VRPTW, Oper Res 42 (1994), 977–978.

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

[18] A. Fügenschuh, The vehicle routing problem with coupled time windows, Cent Eur J Oper Res 14 (2006), 157–176.

[19] S. Gélinas, M. Desrochers, J. Desrosiers, and M.M. Solomon, A new branching strategy for time constrained routing problems with application to backhauling, Ann Oper Res 61 (1995), 91–109.

[20] I. Ioachim, J. Desrosiers, F. Soumis, and N. Belanger, Fleet assignment and routing with schedule synchronization constraints, Eur J Oper Res 119 (1999), 75–90.

[21] I. Ioachim, S. Gelinas, F. Soumis, and J. Desrosiers, A dynamic programming algorithm for the shortest path problem with time windows and linear node costs, Networks 31 (1998), 193–204.

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

[23] T. Justesen and M.S. Rasmussen, The home care crew scheduling problem, Master's thesis, Informatics and Mathematical Modeling, Technical University of Denmark, 2008.

[24] B. Kallehauge, J. Larsen, O.B.G. Madsen, and M.M. Solomon, "Vehicle routing problem with time windows," Chapter 3, Column Generation, G. Desaulniers, J. Desrosiers, and M.M. Solomon (Editors), Springer, NY, 2005, pp. 67–98.

[25] P. Kilby, P. Prosser, and P. Shaw, A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints, Constraints 5 (2000), 389–414.

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

[27] D. Lesaint, N. Azarmi, R. Laithwaite, and P. Walker, Engineering dynamic scheduler for work manager, BT Technol J 16 (1998), 16–29.

[28] Y. Li, A. Lim, and B. Rodrigues, Manpower allocation with time windows and job-teaming constraints, Nav Res Logist 52 (2005), 302–311.

[29] A. Lim, B. Rodrigues, and L. Song, Manpower allocation with time windows, J Oper Res Soc 55 (2004), 1178–1186.

[30] R. Lougee-Heimer, The common optimization interface for operations research: promoting open-source software in the operations research community, IBM J Res Dev 47 (2003), 57–66.

[31] J. Lysgaard, A.N. Letchford, and R.W. Eglese, A new branch-and-cut algorithm for the capacitated vehicle routing problem, Math Prog 100 (2004), 423–445.

[32] D. Oron, S.-N. Sze, and A.S.-F. Ng, A heuristic manpower scheduling for in-flight catering service, The 13th International Conference of Hong Kong Society for Transportation Studies, Kowloon, Hong Kong, 2008.

[33] L.-M. Rousseau, M. Gendreau, and G. Pesant, The synchronized vehicle dispatching problem, Technical Report CRT-2003-11, Centre de Recherche sur les Transports, Université de Montréal, 2003, Canada, Conference paper, Odysseus.

[34] D.M. Ryan, The solution of massive generalized set partitioning problems in aircrew rostering, J Oper Res Soc 43 (1992), 459–467.

[35] M.W.P. Savelsbergh, Local search in routing problems with time windows, Ann Oper Res 4 (1985), 285–305.

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

[37] J.M. Van den Akker, J.A. Hoogeveen, and J.W. Van Kempen, Parallel machine scheduling through column generation: Minimax objective functions, Lecture Notes in Computer Science, 14th Annual European Symposium on Algorithms, ESA 4168 (2006), 648–659.

[38] J.M. Van den Akker, C.A.J. Hurkens, and M.W.P. Savelsbergh, Time-indexed formulations for machine scheduling problems: Column generation, INFORMS J Comput 12 (2000), 111–124.