A NEW FORMULATION TO THE SHORTEST PATH PROBLEM WITH TIME WINDOWS AND CAPACITY CONSTRAINTS

R. DONDO

INTEC (Universidad Nacional del Litoral - CONICET). Güemes 3450 – (3000) Santa Fe – República Argentina
Tel. +54 342 4559174/77 ; Fax: +54 342 4550944. E-mail: rdondo@santafe-conicet.gov.ar

***Abstract*— The Shortest-path problem with time-windows and capacity constraints (SPPTWCC) is a problem used for solving vehicle-routing and crew-scheduling applications. The SPPTWCC occurs as a sub-problem used to implicitly generate the set of all feasible routes and schedules in the column-generation formulation of the vehicle routing problem with time windows (VRPTW) and its variations. The problem is NP-hard in the strong sense. Classical solution approaches are based on a non-elementary shortest-path problem with resource constraints using dynamic-programming labeling algorithms. In this way, numerous label-setting algorithms have been developed. Contrarily to this approach and with the aim to obtain elemental and optimal solutions, we propose a new mixed integer-linear formulation to the SPPTWCC. Some valid inequalities that can be used to strengthen the linear relaxation of the SPPTWCC are also proposed. Numerical experiments on some VRPTW instances taken from Solomon's benchmark problems show that (near) optimal solutions are easily obtained in spite of the considerable problem size. Also the number of generated columns is kept at a very low level.**

***Keywords*— shortest path problem, MILP formulation, column generation, vehicle routing.**

I. INTRODUCTION

The shortest-path problem with time-windows and capacity constraints is a problem widely used for formulating vehicle-routing and crew-scheduling applications (Desaulniers *et al.*, 1998). The SPPTWCC consists of finding the shortest path from a source-node to some nodes of a network (while fulfilling timing and capacity constraints) that ends in a sink node. The term "shortest path" should be carefully interpreted: given costs associated to arcs and prices associated to the nodes, the aim is to find the least-cost path from the source node to the sink node. The SPPTWCC occurs as a sub-problem used to implicitly generate the set of all feasible routes in the column-generation formulation of the vehicle routing problem with time windows (VRPTW) and its variants (Cordeau *et al.*, 2002). It is NP-hard in the strong sense. In the VRPTW, the source and sink nodes are usually located in the same place. This place is commonly named "the depot". For *n*-depot routing problems, *n* source/sink pairs placed in the same location are usually defined. We may relax the problem to consider variants with source and sink nodes placed on different locations. This work deals just with the single-depot case but the more general cases are straight forward.

The SPPTWCC is also a problem with an economic meaning. I.e. given a set of profits associated to the nodes we aim to choose, at a non-zero cost, a subset of nodes that maximizes our net profit.

Classical solution approaches are based on the non-elementary shortest-path problem with resource constraints, using pseudo-polynomial dynamic programming labeling algorithms. Very refined and complex algorithms of this type have been developed. (See e.g. Houck *et al.,* 1980; Irnich and Desaulniers, 2005 and Irnich and Villenueve, 2006). These algorithms are very effective in generating, in addition to the best route, many solutions per iteration. On the contrary, our purpose is just to obtain the optimal solution to the SPPTWCC. Consequently, we propose a new mixed integer-linear formulation (MILP) to the problem.

This work is organized as follows: Section 2 describes the problem and presents its conventional MILP formulation. Section 3 presents a novel MILP formulation based on global precedence relationships. Advantages and weaknesses of this model are also discussed in this section. In Section 4 several preprocessing and polyhedral techniques are applied to the new formulation in order to improve its numerical resolubility. Numerical examples that arise from the well known Solomon benchmark collection are presented and discussed in Section 5. Finally, the conclusions are outlined in section 6.

II. PROBLEM DEFINITION AND ITS USUAL MATHEMATICAL MODEL

Consider a route-network represented by an undirected graph $G\{I \cup p, A\}$ with $I = \{i_1, i_2, ..., i_n\}$ denoting the set of nodes or customers and p representing a source /sink node called "the depot". Nodes and the depot are connected by a set of arcs $A = \{(i, j) / i,j \in I \cup p\}$. Known load and price vectors $L = [l_1 , l_2, \ldots, l_n]$ and $\Pi = [\pi_1, \pi_2, \ldots \pi_n]$ are associated to the customer set I. Loads l_i must be collected within a time window $[a_i, b_i]$ on each node $i \in I$. The parameters a_i stands for the earliest possible start-time of the service and the parameters b_i states the latest possible start-time of the service at the node. In addition, travel-costs $C = \{c_{ij}\}$ and travel times $\Gamma = \{t_{ij}\}$ are given data for any route segment $(i,j) \in A$. Moreover, the service time on node i is denoted st_i. For

each cargo l_i collected node $i \in I$, an associated price π_i is accumulated. It is assumed that the triangle inequality is satisfied by the travel costs and travel times, i.e. $c_{ik} + c_{kj} \geq c_{ij}$ and $t_{ik} + t_{kj} \geq t_{ij}$. The solution to the SPPTWCC problem must: (1) Maximize the net profit collected from the selected subset of nodes $I^{opt} \subseteq I$. This profit is defined as the sum of collected prices minus the cumulated cost incurred by traveling arcs to pick them. (2) The resulting route must start and end on the depot p. (3) The selected nodes must be visited once, so an *elemental path* is designed. (4) The total collected load must never exceed a given capacity q. (5)The time-length used to collect loads and premiums must be shorter than the maximum allowed working time t^{max}. (6)The service at every customer site i must start within the specified time window $[a_i, b_i]$. This problem is usually formulated as follows:

$$Min \qquad \sum_{i\in I}\left[-\pi_i + \sum_{j\in I\cup p} c_{ij}\right]x_{ij} = \sum_{i\in I\cup p}\sum_{j\in I\cup p}\gamma_{ij}x_{ij} \tag{1}$$

Subject to

$$\sum_{i\in I} l_i \sum_{j\in I} x_{ij} \leq q \qquad \forall k \in I \tag{2}$$

$$\sum_{i\in I} x_{pi} = 1 \tag{3}$$

$$\sum_{i\in I} x_{ik} - \sum_{j\in I} x_{kj} = 0 \qquad \forall i, j \in I : i \neq j \tag{4}$$

$$\sum_{i\in I} x_{ip} = 1 \tag{5}$$

$$T_i + st_i + t_{ij} - M_T(1 - x_{ij}) \leq T_j \qquad \forall i \in I \tag{6}$$

$$\begin{Bmatrix} a_i \leq T_i \\ T_i \leq b_i \end{Bmatrix} \qquad \forall i \in I \tag{7}$$

$$T_i + st_i + t_{ip} \leq t^{\max} \qquad \forall i \in I \tag{8}$$

$$x \in \{0,1\}$$

while Eq. (1) states the objective function above mentioned, Eq. (2) states a capacity constraint. Constraints (3), (4) and (5) are flow constraints resulting in a path from the depot p to the subset of chosen nodes and back to the depot. Constraints (6) and (7) are timing constraints and constraint (8) limits the routing time to a maximum value t^{max}. The binary variable x_{ij} indicates if arc $(i,j) \in A$ is used ($x_{ij} = 1$) or not ($x_{ij} = 0$). The parameter γ_{ij} is the reduced cost (c_{ij} - π_i) of using the arc (i,j). While c_{ij} is a non-negative number, γ_{ij} can be any real value. As the SPPTWCC is NP-hard, no algorithm with a worst- case running-time bounded by a polynomial in the size of the problem is known. To optimally solve this problem, we have to use an enumerative algorithm such as branch and bound with a worst-case solution-time that is exponential in the size of the input. Note that the formulation given by Eqs. (1) to (8) is, in practice, unsolvable because of the high number of binary variables x_{ij}. Furthermore, the weak linear relaxations of constraints (6) lead to enormous search trees. Therefore, in the context of column generation algorithms, the most common solution approach to the SPPTWCC is a dynamic-programming label-setting algorithm (See Desrochers *et al.*, 1992). While searching for the optimal route, these procedures generate many non-optimal but useful routes. Very refined and complex algorithms of this type have been developed (Houck *et al.*, 1980; Irnich and Desaulniers, 2005; and Irnich and Villenueve 2006). These methods often lead to problems with thousands of columns that are very hard to solve. Ropke and Cordeau (2009) claim that although the SPPTWCC can initially be solved with heuristic algorithms, whenever the heuristic can no longer produce columns with negative reduced costs, it is necessary to switch to exact methods and always the last pricing problem must be solved to guaranteed optimality in order to prove that the lowest bound is valid. Consequently, optimizing formulations for the SPPTWCC are of crucial importance to prove the optimality of a given solution. This constitutes the main motivation for developing the formulation to the SPPTWCC to be next presented.

III. A REFORMULATION TO THE SPPTWCC

The computational hardness of most combinatorial problems has inspired researchers to develop good formulations that are expected to reduce the size of the enumeration tree and the computation times of these problems. In such a way, we can reformulate the SPPTWCC as follows:

$$Min \quad \left[CV - \sum_{i\in I}\pi_i Y_i\right] \tag{9}$$

subject to:

$$\sum_{i\in I} w_i Y_i \leq q \tag{10}$$

$$\begin{Bmatrix} C_i \geq c_{ip} \\ T_i \geq t_{ip} \end{Bmatrix} \qquad \forall i \in I \tag{11}$$

$$\begin{Bmatrix} C_j \geq C_i + c_{ij} - M_c(1 - S_{ij}) - M_c(2 - Y_i - Y_j) \\ T_j \geq T_i + st_i + t_{ij} - M_T(1 - S_{ij}) - M_c(2 - Y_i - Y_j) \\ \\ C_i \geq C_j + c_{ij} - M_c S_{ij} - M_c(2 - Y_i - Y_j) \\ T_i \geq T_j + st_j + t_{ij} - M_T S_{ij} - M_c(2 - Y_i - Y_j) \end{Bmatrix} \qquad \forall i, j \in I : i < j \tag{12}$$

$$\begin{Bmatrix} CV \geq C_i + c_{ij} - M_c(1 - Y_i) \\ TV \geq T_i + st_i + t_{ij} - M_c(1 - Y_i) \end{Bmatrix} \qquad \forall i \in I \tag{13}$$

$$\begin{Bmatrix} a_i \leq T_i \\ T_i \leq b_i \end{Bmatrix} \qquad \forall i \in I \tag{14}$$

$$TV \leq t^{\max} \tag{15}$$

$$Y_i, S_{ij} \in \{0,1\}$$

The objective function (9) is expressed as minimization of the difference between the overall travelled distance (CV) and the total quantity of collected prices ($\Sigma_{i\in I} \pi_i Y_i$). Eq. (10) is a capacity constraint equivalent to eq. (2). Eq. (11) is like eq. (3) but states flow constraints using continuous variables. It computes the least travelling costs and times (C_i and T_i) from the depot p to a given node i. Eq. (12) combines and reformulates the information from constraints (4) and (6) in order to sequence nodes. In this way, let us assume that nodes i and j are both in the optimal path ($Y_i = Y_j = 1$). Then, the relative ordering of nodes i and j becomes determined by the sequencing variable S_{ij}. In such a case, node j can be a direct/indirect predecessor of node i or viceversa. If node i is visited before j ($S_{ij} = 1$), the travel cost from node i to

node j (C_j) must always be larger than C_i by at least c_{ij}. Furthermore, the arrival time at node j (T_j) should be larger than T_i by at least the sum of the traveling time t_{ij} and the service time (st_i) at the node i. In case node j is visited earlier ($S_{ij} = 0$) , the reverse statements hold-on. Eqs. (13) states that the overall traveling cost (CV) must always be larger than the traveling expenses from the depot to any node i (C_i) along the tour by at least the amount c_{ip}. Also, the total time (TV) required to complete the tour is found by adding the sum of both the service time st_i at node i and the travel time t_{ip} along the edge (i,p) to the initial service time at the node last visited i. Since the node last visited is not known beforehand, the eq. (13) must be written for every node $i \in I$. Eqs. (14) and (15) are time-windows and maximum routing time constraints. This formulation differs from the usual formulation to the SPPTWCC, given by Eqs. (1)-(8) in the following main issues:

- It uses a continuous representation for both the time domain (variables TV and T_i for all $i \in I$) and the cost domain (variables CV and C_i for all $i \in I$).
- The new formulation handles node-visit and node-sequencing decisions through different sets of binary variables. The variable Y_i indicates if node $i \in I$ belongs to the optimal path ($Y_i = 1$) or not ($Y_i = 0$) while the variable S_{ij} set the generalized precedence relationship between nodes $(i, j) \in I$: $i < j$ if both are in the optimal path ($Y_i + Y_j = 1$). Constraints (12) become redundant whenever nodes i and/or j are not in the optimal path ($Y_i + Y_j \leq 1$). In such a case, the constraint will state that C_i, T_i, C_j and T_j are all larger than a large negative number.
- The reformulation uses the concept of generalized predecessors for sequencing variables. Variable S_{ij} indicates that node i is visited before ($S_{ij} = 1$) or after ($S_{ij} = 0$) node j just in case both nodes belong to the optimal tour. This variable indicates both direct and indirect precedence relationships. I.e. if i is a direct predecessor of j, the term ($c_i \geq c_i + c_{ij}$) will be satisfied as equality. If i is a indirect predecessor of j, the inequality sign ">" will become active.

Our formulation is aimed at limiting one flaw of the original one; the high number of 0-1 variables. More precisely we have

$O[|I| (|I|-1)]$ binary variables;
$O[|I|]$ continuous variables;
$O[3 + 3|I| + |I| (|I|-1)]$ constraints

on the conventional formulation and

$O[|I| + ½|I| (|I|-1)]$ binary variables;
$O[2|I| + 2]$ continuous variables;
$O[6|I| + 4|I| (|I|-1) + 2]$ constraints

on the new formulation. So, a considerable saving of 0-1 variables is achieved. These can be reduced to almost a half (if $|I|$ is large) respect to the conventional formulation, at the cost of increasing continuous variables and constraints. Consequently, we have a formulation with fewer binary variables and more constraints. This should be favourable from a resolution point of view and should lead to smaller branch and bound trees. Although the new model uses fewer binary variables, greatly exploits "big-M" type constraints. Thus, it is expected to have a loose continuous relaxation. It is well known that big-M constraints can be difficult for branch and bound solvers since they generate un-tight lower bounds that are crucial for the efficiency of the solver. In general, the larger M in the big-M constraint, the less tight the formulation. Then, the big-M value should be substituted by an upper bound in the terms that activates. I.e, in the equation:

$$TV \geq T_i + st_i + t_{ij} - M_T(1 - Y_i)$$

the parameter M_T should be replaced by an upper bound on the term ($T_i + st_i + t_{ij}$). In this case the bound is ($b_i + st_i + t_{ij}$). In this way, "customized" and as small as possible big-M parameters should be introduced into each constraint. This should lead to a higher tightness of constraints and therefore, the problem might be faster solved. Nevertheless, even with this "strongest" version of inequalities, they are very unfavourable for linear relaxations. Consequently, we have achieved one objective (to reduce the number of 0-1 variables) but we are still unable to provide tight lower bounds to linear relaxations. Since the number of constraints with big-M structure are higher than in the new formulation, the original formulation (Eqs. 1-8) presents even worse bounds. Ways to improve the linear relaxations on the new formulation are presented in the next section.

IV. PRE-PROCESSING AND POLYHEDRAL TECHNIQUES IN THE NEW MODEL

The size and complexity of solved integer problems can be increased when the *polyhedral theory* is applied. The underlying idea of polyhedral *combinatronics* is to replace the constraints-set of the integer-programming problem by an alternative *convexification* of the feasible points and rays of the convex hull *conv(S)* of the problem (von Hoesel and Aardal, 1996). Then, if we can list the whole set of linear inequalities that defines *conv(S)* we can solve the integer problem by linear programming. Nevertheless, for most integer problems the minimal number of inequalities necessary to describe the polyhedron is exponential in the number of variables. Thus, it is unrealistic to search for the complete set of inequalities. In addition, if generated within a branch-and-bound tree they can be not valid throughout the entire tree since cuts are generated assuming that certain values are fixed. Therefore, we are interested in a set of constraints *conv(S)* as small as possible and with no a-priory value-assumptions. Then, one should consider inequalities that define a *facet* of *conv(S)*. They are the "best possible" in the sense that they cannot be "stronger" without losing some feasible mixed-integer solutions of the original problem. Frequently only a partial set of valid inequalities "located" in the neighbourhood of the optimal solution is useful to reduce solution times (Hoffmann, 2000). It is worth noting that, to make polyhedral methods work well, one important issue is pre-processing. Important elements of pre-processing are to reduce the size of the initial formulation and to

reduce the range of constraint coefficients to make instances numerically more efficient.

A. Pre-processing: In pre-processing, the formulation is tightened before the actual optimization. This is done by fixing some variables or by reducing the interval of values a variable can take. This leads to a more compact solution space and consequently to shorter solution times. In this way, to pre-set some sequencing constraints, we define the following useful sets:

Nodes-compatibility relationship sets:

Set of nodes compatible with $i \in I$: A node j is said to be compatible with a reference node i if can be visited either before or after i. This compatibility condition is stated by the following set:

$$Com(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) \le b_j \wedge \left(a_j + st_j + t_{ij}\right) \le b_i \right\} \quad (16)$$
$$\forall i \in I$$

Set of predecessors of node $i \in I$: A pair of nodes (i,j) is said to be pre-ordered if they must be visited in a certain pre-determined order when time-window constraints are satisfied. For instance, node j is said to be a predecessor of node i if j must be visited before node i. This condition is defined by the following set:

$$\Pr e(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) > b_j \wedge \left(a_j + st_j + t_{ij}\right) \le b_i \right\} \quad (17)$$
$$\forall i \in I$$

Set of successors of node $i \in I$: Node j is said to be a successor of node i if j must be visited after node i. Successors of node i are specified by the following set:

$$Suc(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) \le b_j \wedge \left(a_j + st_j + t_{ij}\right) > b_i \right\} \quad (18)$$
$$\forall i \in I$$

Set of nodes incompatibles with $i \in I$: Nodes (i,j) that cannot be assigned to the same path are called incompatible. The incompatibility condition for nodes $j \neq i$ is stated by the following set:

$$Inc(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) > b_j \wedge \left(a_j + st_j + t_{ij}\right) > b_i \right\} \quad (19)$$
$$\forall i \in I$$

Stronger compatibility relationships can be defined if we use the concept of "immediacy" for defining the following more restrictive sets:

Set of immediate predecessors of node $i \in I$: A node j is said to be an immediate predecessor of node i if it is a predecessor of such a node and in addition, no other node $k \in I$: $k \neq j$ can be inserted in a path from i toward j. In such a case, the only feasible path connecting i and j is the arc $(i, j) \in N$. Immediate precedence relationships are stated by the following set:

$$I\Pr e(i) = \{ j \in \Pr e(i) : a_j + st_j + t_{ij} \ge b_i - \min_{k \in I}(st_k + t_{ik}) \} \quad (20)$$
$$\forall i \in I$$

Immediate successors are defined in the same way:

$$ISuc(i) = \{ j \in Suc(i) : a_i + st_i + t_{ij} \ge b_j - \min_{k \in I}(st_k + t_{ij}) \} \quad (21)$$
$$\forall i \in I$$

Immediately compatible nodes are nodes i and $j \in I$ that are compatible and, in addition, no a third node k can be inserted between them, no matter its relative ordering. Immediate compatibility is defined by the following set:

$$ICom(i) = \{ j \in Com(i) : a_j + st_j + t_{ij} \ge b_i - \min_{k \in I}(st_k + t_{ik}) \wedge a_i + st_i + t_{ij} \ge b_j - \min_{k \in I}(st_k + t_{ij}) \} \quad (22)$$
$$\forall i \in I$$

B. Domain reduction: The narrowing of the range of a given variable is known as domain-reduction. The reduction of the domain of a continuous variable refines the information known about this variable. In the context of the VRPTW and the SPPTWCC, as the triangle inequality holds, the earliest arrival time to a customer can be strengthened by the time used to arrive straight from the depot and the latest arrival time can be strengthened by driving the fastest way to the depot. So, the customer i time-window can initially be strengthened from $[a_i, b_i]$ to [max $(a_p + t_{pi,} a_i)$, min $(b_p - st_i - t_{ip}, b_i)$]. A further reduction can be achieved, in the context of the formulation (9)-(15), by using two of the Desrochers rules (Desrochers *et al.*, 1992). Considering that node i is assigned to the optimal tour, the beginning of the time windows for all successor $j \in Suc(i)$ of customer i, and the closing of time windows for all predecessors $j \in Pre(i)$ of node i, can be strengthened as follows:

$$\forall i \in I$$
$$\forall j \in Suc(i) : a_j < a_i + t_{ij} + st_i \quad a_j \leftarrow a_i + t_{ij} + st_i \quad (23.a)$$
$$\forall j \in \Pr e(i) : b_j > b_i - t_{ij} - st_i \quad b_j \leftarrow b_i - t_{ij} - st_i \quad (23.b)$$

These rules are useful in case a node is assigned to a partial tour before the actual optimisation as i.e. in not-root nodes of a branch and bound tree for the VRPTW. Rules (23) not only narrow time windows. They also allow to move some nodes from sets $Com(i)$ to $Pre(i) \cup Suc(i)$ and from $Pre(i) \cup Suc(i)$ to $Inc(i)$. So, the re-use of narrowing rules and the re-definition of compatibility sets until no further changes are achieved lead to easier problems. In such a way, the sequencing constraints (12) now can be split as follows:

$$Y_i + Y_j \le 1 \qquad \forall i, j \in I : i < j \wedge j \in Inc(i) \quad (12.b)$$

$$\left\{ \begin{array}{l} C_j \ge C_i + c_{ij} - M_c\left(2 - Y_i - Y_j\right) \\ T_j \ge T_i + st_i + t_{ij} - M_T\left(2 - Y_i - Y_j\right) \end{array} \right\} \quad \forall i, j \in I : i < j \wedge j \in Suc(i) \quad (12.c)$$

$$\left\{ \begin{array}{l} C_i \ge C_j + c_{ij} - M_c\left(2 - Y_i - Y_j\right) \\ T_i \ge T_j + st_j + t_{ij} - M_T\left(2 - Y_i - Y_j\right) \end{array} \right\} \quad \forall i, j \in I : i < j \wedge j \in \Pr e(i) \quad (12.d)$$

$$\left\{ \begin{array}{l} C_j \ge C_i + c_{ij} - M_c\left(1 - S_{ij}\right) - M_c\left(2 - Y_i - Y_j\right) \\ T_j \ge T_i + st_i + t_{ij} - M_T\left(1 - S_{ij}\right) - M_c\left(2 - Y_i - Y_j\right) \\ \\ C_i \ge C_j + c_{ij} - M_c S_{ij} - M_c\left(2 - Y_i - Y_j\right) \\ T_i \ge T_j + st_j + t_{ij} - M_T S_{ij} - M_c\left(2 - Y_i - Y_j\right) \end{array} \right\} \quad \forall i, j \in I : i < j \wedge j \in Com(i) \quad (12.e)$$

So, the use of narrowing rules and precedence relationships can lead to a considerable saving of sequencing variables and sequencing constraints. Furthermore, the remaining sequencing constraints can be greatly simplified.

C. Valid inequalities: The use of information about the structure of the convex hull of feasible solutions has been so far one of the most useful approaches for solving combinatorial problems (Hoffmann, 2000) and will be fully exploited to strength the new formulation in order to tight the lower bounds of a branch and bound procedure. Consequently, we aim to develop specific inequalities necessary in the description of the convex hull of the solution space of the SPPTWCC formulated according Eqs. (9) to (15). If a problem is NP-hard, we cannot expect to have a concise description of the convex hull *conv(S)* of the feasible solutions. This does not have necessarily a negative impact since what we need is just a good description of the "area around" the optimal solution (Hoffmann, 2000). As the number of valid inequalities grows exponentially on the number of nodes, we should focus only in the most attractive inequalities. Those are the ones that eliminate the greatest possible number of suboptimal configurations within a given path and are easy to compute. Sometimes simple exact rules are enough to provide a number of valid inequalities and we identified the following types:

Valid inequalities based on two-vertexes: These are constraints aimed at exploit information from every pair of nodes $i,j \in I$: $i < j$. Therefore its computation burden is in the order $O(n^2)$. For a given couple of nodes $i,j \in I$: $i < j$, first let us assume that node j is a successor of node i. Then, the only feasible path involving both nodes *to* the depot is $(i \rightarrow j \rightarrow p)$. In addition, if the earliest possible returning time to the depot is larger than $t' = t_v^{max} - min_{j\, k \in I:\, j \neq k}\, (st_k + t_{jk})$, no visit to another node $k \in I$: $k \neq j \neq i$ is possible. In such a case, if the cost of returning via j $(c_{ij} - \pi_j + c_{jp})$ is higher than the cost of returning directly to the depot (c_{ip}), the partial path $(i \rightarrow j \rightarrow p)$ is sub-optimal and does not belong to the optimal path. Consequently, the inequality $(Y_i + Y_j) \leq 1$ becomes valid. A "mirror inequality" is valid if the only feasible path involving both nodes *to* the depot is $(j \rightarrow i \rightarrow p)$ and if the cost of returning via i (i.e. $c_{ji} - \pi_i + c_{ip}$) is higher than the cost of returning directly to the depot (c_{ip}). Finally, if $i,j \in I$: $i < j$ are compatible nodes, two partial paths would be feasible; $(i \rightarrow j \rightarrow p)$ and $(j \rightarrow i \rightarrow p)$. Consequently, both the direct way and the indirect way to the depot are feasible. If both indirect ways are more expensive than the direct ones, the inequality $(Y_i + Y_j) \leq 1$ becomes valid. These above assertions are mathematically expressed by Eq. (26).

$$\left\{\begin{array}{l} \forall i,j \in I : i < j,\ j \in Suc(i) \wedge \\ \left(a_i + st_i + t_{ij} + st_j + t_{jp}\right) \geq t' \wedge \left(c_{ij} - \pi_j + c_{jp}\right) \geq c_{ip} \\ \\ \forall i,j \in I : i < j,\ j \in \Pr e(i) \wedge \\ \left(a_j + st_j + t_{ji} + st_i + t_{ip}\right) \geq t' \wedge \left(c_{ji} - \pi_i + c_{ip}\right) \geq c_{jp} \\ \\ \forall i,j \in I : i < j,\ j \in Com(i) \wedge \\ \left(a_i + st_i + t_{ij} + st_j + t_{jp}\right) \geq t' \wedge \left(c_{ij} - \pi_j + c_{jp}\right) \geq c_{ip} \\ \wedge \left(a_j + st_j + t_{ji} + st_i + t_{ip}\right) \geq t' \wedge \left(c_{ji} - \pi_i + c_{ip}\right) \geq c_{jp} \end{array}\right\} \quad (24)$$

$$Y_i + Y_j \leq 1$$

where $b_k' = (b_k - st_k - min_{\,ij:\, i \neq j}\, t_{ij})$. Similar inequalities can be designed considering the depot as the start node for the partial paths involving nodes i and j. This lead to the following additional valid inequalities:

$$\left\{\begin{array}{l} \forall i,j \in I : i < j,\ j \in Suc(i) \wedge \\ \left(t_{pi} + st_i + t_{ij}\right) \geq b_j' \wedge \left(c_{pi} - \pi_i + c_{ij}\right) \geq c_{pj} \\ \\ \forall i,j \in I : i < j,\ j \in \Pr e(i) \wedge \\ \left(t_{pj} + st_j + t_{ji}\right) \geq b_i' \wedge \left(c_{pj} - \pi_j + c_{ji}\right) \geq c_{pi} \\ \\ \forall i,j \in I : i < j,\ j \in Com(i) \wedge \\ \left(t_{pi} + st_i + t_{ij}\right) \geq b_j' \wedge \left(c_{pi} - \pi_i + c_{ij}\right) \geq c_{pj} \\ \wedge \left(t_{pj} + st_j + t_{ji}\right) \geq b_i' \wedge \left(c_{pj} - \pi_j + c_{ji}\right) \geq c_{pi} \end{array}\right\} \quad (25)$$

$$Y_i + Y_j \leq 1$$

Valid inequalities based on three-vertexes: These inequalities could be derived considering every triplet $i, j, k \in I$: $i < j < k$ with a computational burden in the order $O(n^3)$. In such a case, the information about time-windows and about node-prices can be exploited.

Valid inequalities based on time-windows information: Let us assume, without loss of generality, that the only feasible partial path between nodes i, j and k is $(i \rightarrow j \rightarrow k)$. If, in addition, the minimum path time (i.e. $a_i + st_i + t_{ij} + st_j + t_{jk}$) for going to k via j is bigger than b_k, the three nodes cannot be simultaneously on the same path. Then the inequality $(Y_i + Y_j + Y_k) \leq 2$, becomes binding. This is just one case on which the nodes are pre-ordered. For the two remaining cases (i.e. $i,j,k \in I$: $i < j < k$, $j \in$ Pre*(i)*, $k \in$ Pre*(j)* and $i,j,k \in I$: $i < j < k$, $j \in$ Pre*(i)*, $k \in$ Suc*(i)*), the same inequality is binding. They are expressed as follows:

$$\left\{\begin{array}{l} i,j,k \in I : i < j < k,\ j \in Suc(i), k \in Suc(j) \wedge \\ \left(a_i + st_i + t_{ij} + st_j + t_{jk}\right) \geq b_k \\ i,j,k \in I : i < j < k,\ j \in \Pr e(i), k \in \Pr e(j) \wedge \\ \left(a_k + st_k + t_{kj} + st_j + t_{ji}\right) \geq b_i \\ i,j,k \in I : i < j < k,\ j \in \Pr e(i), k \in Suc(i) \wedge \\ \left(a_j + st_j + t_{ji} + st_i + t_{ik}\right) \geq b_k \end{array}\right\} \quad (26)$$

$$Y_i + Y_j + Y_k \leq 2$$

Furthermore, if two of the three vertexes of a given triplet are compatible, two partial paths must be verified. If both paths are suboptimal, the inequality $(Y_i + Y_j + Y_k) \leq 2$ would be binding. I.e. if nodes k and j are compatible and both are successors of node i, two minimum time paths are to be verified. If $(a_i + st_i + t_{ij} + st_j + t_{jk})$ is higher than b_k and if $(a_i + st_i + t_{ik} + st_k + t_{kj})$ is larger than b_j, then inequality $(Y_i + Y_j + Y_k) \leq 2$ becomes valid. The same inequality must be written if compatible nodes k and j are both predecessors of node i and if both three-node paths are infeasible. The whole statement is expressed by the following equation:

$$\left\{\begin{array}{l} i,j,k \in I : i<j<k, j \in Suc(i), k \in Suc(j) \wedge k \in Com(j) \\ \left(a_i + st_i + t_{ij} + st_j + t_{jk}\right) \geq b_k \wedge \\ \left(a_i + st_i + t_{ik} + st_k + t_{kj}\right) \geq b_j \\ i,j,k \in I : i<j<k, j \in \Pr e(i), k \in \Pr e(j) \wedge k \in Com(j) \\ \wedge \left(a_k + st_k + t_{kj} + st_j + t_{ji}\right) \geq b_i \wedge \\ \left(a_j + st_j + t_{jk} + st_k + t_{ki}\right) \geq b_i \end{array}\right\} \quad (27)$$
$$Y_i + Y_j + Y_k \leq 2$$

Price-based inequalities: Value of prices π_i also can be exploited for identifying suboptimal paths. I.e. let us consider that the only feasible partial path between nodes *i, j, k* ∈ *I* is ($i \rightarrow j \rightarrow k$) and that, in addition, no other node can be inserted between them, so *j* is an immediate successor of node *i* and *k* is an immediate successor of node *j*. Let us now compare the costs of both feasible paths. If the direct path from *i* to *k* is cheaper than the indirect path ($i \rightarrow j \rightarrow k$) because ($c_{ij}$ - π_j + c_{jk}) > c_{ik}, then the three nodes must not be in the optimal path. Therefore the inequality (Y_i + Y_j + Y_k) ≤ 2 becomes valid. For the two remaining cases (i.e. *i,j,k* ∈ *I: i* < *j* < *k, j* ∈ *Pre(i), k* ∈ Pre*(j)* and *i,j,k* ∈ *I: i* < *j* < *k, j* ∈ *Pre(i), k* ∈ *Suc(i)*), mirror inequalities are binding. They are derived in the same way that time-window based inequalities and are stated as follows:

$$\left\{\begin{array}{l} i,j,k \in I : i<j<k, j \in Suc(i), k \in Suc(j) \wedge \\ \left(a_i + st_i + t_{ij} + st_j + t_{jk}\right) \geq b_k' \wedge \\ \left(c_{ij} - \pi_j + c_{jk}\right) \geq c_{ik} \\ \\ i,j,k \in I : i<j<k, j \in \Pr e(i), k \in \Pr e(j) \wedge \\ \left(a_k + st_k + t_{kj} + st_j + t_{ji}\right) \geq b_i' \wedge \\ \left(c_{kj} - \pi_j + c_{ji}\right) \geq c_{ki} \\ \\ i,j,k \in I : i<j<k, j \in \Pr e(i), k \in Suc(i) \wedge \\ \left(a_j + st_j + t_{ji} + st_i + t_{ik}\right) \geq b_k' \wedge \\ \left(c_{ji} - \pi_i + c_{ik}\right) \geq c_{jk} \end{array}\right\} \quad (28)$$
$$Y_i + Y_j + Y_k \leq 2$$

Furthermore, if two nodes are compatible, inequalities like the ones of Eq. (27) would also be binding. They are stated by Eq. (29).

$$\left\{\begin{array}{l} i,j,k \in I : i<j<k, j \in Suc(i), k \in Suc(j) \wedge \\ k \in Com(j) \wedge \left(c_{ij} - \pi_j + c_{jk}\right) \geq c_{ik} \wedge \left(c_{ik} - \pi_k + c_{kj}\right) \geq c_{ij} \\ \left(a_i + st_i + t_{ij} + st_j + t_{jk}\right) \geq b_k \wedge \left(a_i + st_i + t_{ik} + st_k + t_{kj}\right) \geq b_j \\ \\ i,j,k \in I : i<j<k, j \in \Pr e(i), k \in \Pr e(j) \wedge \\ k \in Com(j) \wedge \left(c_{kj} - \pi_j + c_{ji}\right) \geq c_{ki} \wedge \left(c_{jk} - \pi_k + c_{ki}\right) \geq c_{ki} \\ \wedge \left(a_k + st_k + t_{kj} + st_j + t_{ji}\right) \geq b_i \wedge \left(a_j + st_j + t_{jk} + st_k + t_{ki}\right) \geq b_i \end{array}\right\} \quad (29)$$
$$Y_i + Y_j + Y_k \leq 2$$

Inequalities based on immediate compatibility relationships: Let us consider that node *i* ∈ *I* belongs to the optimal path. In such a case, just one of all nodes *j* ∈ *IPre*(*i*) may be a predecessor of vertex *i* and just one node *k* ∈ *ISuc*(*i*) may be its successor. This raises the following two valid inequalities:

$$\left|I \Pr e(i)\right| Y_i + \sum_{j \in I \Pr e(i)} Y_j \leq \left|I \Pr e(i)\right| + 1 \quad (30.a)$$

$$\left|ISuc(i)\right| Y_i + \sum_{j \in ISuc(i)} Y_j \leq \left|ISuc(i)\right| + 1 \qquad \forall i \in I \quad (30.b)$$

V. LIFTING THE INEQUALITIES

Lifting refers to extending valid inequalities from a low dimensional polyhedron to polyhedrons that are valid in higher dimensions. The concept of lifting has been introduced by Gomory (1958). Padberg (1979) developed a sequential lifting procedure for binary programming. After those pioneering works, numerous lifting techniques for a variety of constraint-structures have been proposed (See e.g. Hosel and Aardal, 1996). We will use some established results in order to lift the above presented inequalities. First, let us consider the incompatibility relationship (12.b). As the node *i* is incompatible with any node *j* ∈ *Inc*(*i*), if *i* belong to the optimal path, no just the node *j* ∈ *Inc*(*i*) can be inserted on it. Any other node *k* ∈ *Inc*(*i*): *k* # *j* incompatible with *i* will also be forbidden. Then, the following will be a stronger valid constraint:

$$\left|Inc(i)\right| Y_i + \sum_{j \in Inc(i)} Y_j \leq \left|Inc(i)\right| \qquad \forall i \in I \quad (31)$$

This constraint can replace the arrangement of incompatible-couple constraints related to node *i* that are stated by Eq. (12.b). Eq. (31) is a stronger constraint because it extends inequality (12.b) from a $\{0, 1\}^2$ space to a $\{0, 1\}^{|\text{Inc(i)}|}$ dimension. It also reduces the total number of constraints of the formulation.

The first constraint of Eq. (24) can be lifted by a similar procedure and the following inequality can be obtained:

$$Y_i \left|\Xi\right| + \sum_{j \in \Xi} Y_j \leq \left|\Xi\right| \qquad \forall i \in I \quad (32)$$

where Ξ = { *i,j* ∈ *I* : *j* ∈ Suc*(i), i* < *j*, $a_i + st_i + t_{ij} + st_j$ > *t′*, c_{ij} - π_j + c_{jp} > c_{ip}}. This means that if node *i* is in the optimal path, not just one but all nodes *j* that fulfil the condition stated by the first constraint of Eq. (24) must be excluded from the tour. Now, this stronger inequality can replace the former constraint. A mirror equation can be written for the second constraint of Eq. (24). In the same way, the last restriction can be lifted to obtain its following enforced replacement:

$$Y_i \left|\Psi\right| + \sum_{j \in \Psi} Y_j \leq \left|\Psi\right| \qquad \forall i \in I \quad (33)$$

where Ψ = { *i,j* ∈ *I* : *j* ∈ Suc*(i), i* < *j*, $a_i + st_i + t_{ij} + st_j$ > *t′*, c_{ij} - π_j + c_{jp} > c_{ip}, $a_j + st_j + t_{ij} + st_i$ > *t′*, c_{ji} - π_i + c_{ip} > c_{jp} }. In addition, inequalities that consider the depot as a tour start-point are lifted in the same way and enforced versions of eqs. (25) can be straightforward derived and written. Now, considering the first constraint of eq. (26), we can see that if both nodes *i* and *j* are in the optimal path, the least arrival time to node *j* moves from a_j to *max*[a_j ; $a_i + st_i + t_{ij}$]. Consequently some nodes *k* that are successors of of node *j* may become incompatible to the couple (*i*, *j*) and the following will be a facet-defining-constraint:

$$\frac{|\Omega|}{2}(Y_i + Y_j) + \sum_{k\in\Omega} Y_k \le \Omega \qquad (34)$$

$$\forall i, j \in I : i < j \wedge j \in Suc(i)$$

where $\Omega = \{ i,j,k \in I: k \in \text{Suc}(i), a_i + st_i + t_{ij} + st_j + t_{kj} > b_k\}$. This inequality states that successors of node j that cannot be visited later than $(a_i + st_i + t_{ij} + st_j + t_k)$ are to be excluded from the optimal path just in case nodes i and j belong to it. Conversely, if j is predecessor of node i, the following mirror inequality would become valid:

$$\frac{|\Omega'|}{2}(Y_i + Y_j) + \sum_{k\in\Omega'} Y_k \le |\Omega'| \qquad (35)$$

$$\forall i, j \in I : j \in \Pr e(i), i < j$$

where $\Omega' = \{k \in I: k \in \text{Suc}(i): a_j + st_j + t_{ji} + st_i + t_{kj} > b_k\}$.

VI. SOME NUMERICAL EXAMPLES AND DISCUSSION

One of the most prominent vehicle routing problems with side constraints is the VRPTW. This section illustrates the use of the new MILP formulation on a simple branch-and-price algorithm aimed at finding optimal solutions to VRPTW instances. Solomon (1987) benchmark VRPTW instances have attracted numerous researchers to develop exact and heuristic solution procedures and are commonly used as a test-bed for these procedures. The collection of Solomon's 56 problems has been grouped into three categories: C, R and RC. C-class problems feature clustered customers whose time windows have been generated based on known solutions. Locations in R-class problems were randomly generated over a square while RC-class problems comprise a combination of clustered and randomly generated customers. The data set for every category comprises 100 nodes, a central depot, similar vehicle capacities but different time-window distributions. Euclidean distances among customers and traveling times are numerically identical. Furthermore, time windows are hard constraints, service times are independent of customer requirements and the tour duration cannot exceed a maximum value t^{max}. The objective is the minimization of the total distance. Smaller problems can be generated by selecting the first 25 or 50 nodes of each instance. Benchmark problems of each class are further classified into types "1" and "2", like C1 and C2. Type-1 problems have narrow time windows and small vehicle capacities while type-2 problems feature wider time windows and larger vehicle capacities. In order to evaluate the performance of our SPPTWCC formulation we first solved all R1-type instances with just the first 25 nodes. We selected this group because the different time-windows lead to solutions involving a wide span of solution-shapes. I.e. problem R101 have a solution with numerous trips involving a few nodes per trip while problems R104, R108 and R112 have solutions with fewer tours and many nodes per tour.

In order to "translate" these benchmark problems to the SPPTWCC, we included into the data a price vector $\Pi = [\pi_1, \pi_2, \ldots \pi_{25}]$.

The vector was obtained by generating columns in a column generation procedure until reaching the optimal lower bound to the problem. Then, the price vector was "frozen" into values obtained in such a way. Also inter-node distances were rounded to the nearest first decimal value. Price-values are reported in Table 1. To test the proposed SPPTWCC model, different configurations of enforced formulations were coded using ILOG OPL Studio 3.7 and all R1-problems with 25 nodes were solved in a 2.8 GHz 1.0 GB RAM Pentium IV PC.

We first compared three basic configurations. Configuration 1 uses the new model after applying preprocessing and domain reduction but no valid inequalities at all. Configuration 2 adds the valid inequalities and Configuration 3 replaces them by the "lifted" equations presented in section V. Table 2 summarizes objective function values of relaxed SPPTWCC. These relaxed problems were solved considering variables Y_i and S_{ij} as continuous variables bounded by the interval [0, 1]. As prices reported on Table 1 were computed considering variables Y_i and S_{ij} as binary variables, the re laxation of their integrality leaded to negative objective

Table 1: Optimal-price values on each Solomon R1-instances comprising the first 25 nodes

Node	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
1	5.3	5.3	6.6	11.6	8.7	1.0	14.7	7.3	5.3	1.0	8.1	5.3
2	29.7	9.5	8.6	7.9	31.5	7.7	9.7	10.4	10.4	10.4	10.3	10.8
3	34.8	9.2	19.7	24.4	38.8	16.7	21.8	19.2	39.5	37.2	25.8	15.9
4	20.5	21.6	35.7	24.9	20.5	15.2	20.4	34.5	17.0	8.9	16.6	16.0
5	16.0	8.5	12.6	8.0	18.6	2.6	6.8	12.3	19.4	11.7	4.1	15.1
6	6.6	6.6	20.9	0.6	14.1	16.1	17.6	3.6	13.8	6.6	21.3	8.3
7	41.8	7.2	13.2	11.9	15.6	19.7	10.1	11.8	15.6	15.6	9.0	6.6
8	17.8	51.5	38.2	31.0	34.1	29.3	33.3	32.2	21.6	18.1	22.2	22.1
9	34.6	33.1	35.2	34.4	29.3	41.5	22.4	22.3	18.8	27.7	22.6	33.1
10	19.0	26.2	29.6	14.9	25.5	3.2	20.6	14.9	29.6	9.8	18.1	14.9
11	49.9	39.5	27.0	28.4	37.2	34.5	21.9	16.1	27.6	26.1	36.2	26.1
12	8.5	2.7	2.70	2.7	8.5	3.8	4.1	7.8	3.8	8.3	8.9	14.6
13	7.1	7.1	8.5	13.2	7.1	13.4	11.9	9.3	7.1	7.1	7.1	7.1
14	27.4	19.9	31.5	22.8	36.8	25.1	30.6	27.4	27.3	20.2	32.2	22.4
15	44.5	43.2	30.6	35.3	23.6	30.2	21.1	22.6	17.6	20.7	23.4	19.6
16	38.4	45.9	4.4	9.0	11.5	24.8	12.4	9.6	15.0	24.2	9.6	12.4
17	18.1	10.7	12.1	16.8	18.1	18.6	19.2	12.2	28.4	15.6	19.4	11.7
18	31.6	31.6	6.4	6.4	18.3	20.0	4.6	6.3	12.4	25.2	13.7	14.1
19	12.2	10.9	5.1	12.1	9.7	18.7	14.2	16.1	3.3	10.6	6.0	9.5

20	35.0	32.7	16.7	9.9	16.7	21.0	15.4	27.4	16.7	35.0	15.4	15.1
21	13.2	4.3	3.7	5.7	0.0	5.6	10.9	8.2	9.2	15.3	13.8	13.5
22	10.3	10.3	10.3	15.0	10.3	15.6	16.4	9.8	37.5	23.0	16.4	17.9
23	55.6	48.9	35.9	30.9	55.6	27.7	28.5	20.3	15.4	26.1	28.3	21.1
24	21.7	41.8	21.0	19.0	23.0	29.0	17.9	16.6	18.3	21.7	21.1	17.9
25	18.5	19.8	19.2	21.2	18.5	40.7	22.3	19.7	11.7	14.1	20.0	20.4
Total	617.1	547.1	454.7	417.0	530.5	465.4	424.4	397.3	441.7	452.2	429.7	395.1

function values. It can be observed here the effect of valid inequalities in the relaxation of the problems. The shrinkage of the objective function value shows that these additional constraints might lead to smaller branch and price trees and therefore to shorter CPU times. Nevertheless, the relative reduction of the (negative) objective function value seems to depend on the problem morphology. These claims were tested by solving the R1-type VRPTW instances using the three different configurations of the SPPTWCC as slave problems within a basic branch and bound procedure (Barnhart *et al.*, 2000). Results are presented in Table 3. We can note the following patterns:

- The effect of valid inequalities in problems constrained by tight time-windows (i.e. R101, R105 and R109) is null or slightly negative.
- The positive effect of valid inequalities is mostly seen in problems constrained by weak time-windows and can be considerable. See solution times for the R104, R108 and R-112 problems.
- The global effect of valid inequalities is positive because tightly constrained problems (where the effect of inequalities is slightly negative) are much easier to solve than almost no-constrained problems on which the accelerating effect of valid inequalities can be substantially positive.
- The accelerating effect of lifted inequalities respect to un-lifted valid inequalities is minor.

The Table 4 shows the number of columns generated while using the MILP formulations of the SPPTWCC. Interestingly, this number remains more or less constant for the whole problem series and is dramatically smaller than in algorithms that use conventional label setting procedures as routes-generators (See Larsen, 2000). In spite of this, CPU times are larger in our algorithms. This is because label-setting algorithms generate many columns per iteration. In spite of this initial disadvantage, our procedure shows the following properties:

- It is very flexible since it can handle almost any structure of constraints while providing good performances on known benchmark problems. Then, this MILP model can be a cornerstone to model more complex problems, like i.e. the pick-up and delivery problem and its realistic variations as multi-commodity vehicle routing problems, routing problems with multiple time-windows, etc.
- It should lead to smaller branch-and-price trees, thus erasing its disadvantages respect to label setting algorithms in larger problems.

To test the last claim we solved the R1 series involving the first 50 customers. The Table 5 shows resolution data for this problem set. Interestingly, if we compare this data with data from problems involving just 25 nodes, we can't detect an explosion of CPU times and of generated columns.

The Table 6 shows resolution data en the R1-series involving 100 nodes. Although the number of columns grow with the problem size, this growing seems to be more or less bounded. I.e. the average number of columns is around 200 for the 25 node-series, 400 for the 50 nodes series and around 1000 for the 100 nodes series.

Table 2: Objective function values for different SPPTWCC formulations with relaxed assignment (Y_i) and sequencing (S_{ij}) desicion-variables

Config	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
1	-324.77	-305.47	-277.18	-266.30	-306.53	-289.07	-259.20	-263.08	-287.41	-319.48	-270.64	-272.35
2	-296.18	-268.27	-241.07	-223.99	-281.19	-265.04	-215.26	-211.36	-235.79	-243.80	-221.80	-206.78
3	-314.55	-279.01	-237.46	-213.70	-260.29	-251.99	-212.48	-203.81	-217.20	-235.45	-218.59	-206.78

Table 3: Resolution times (in seconds) and objective function values for the Solomon R1-instances comprising the first 25 nodes.

Config	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
						CPU time						
1	3.81	6.94	24.68	72.81	7.93	13.88	40.50	104.12	24.63	119.31	66.41	454.39
2	8.93	7.87	22.22	41.63	9.16	13.69	37.39	74.04	22.95	66.35	31.62	185.10
3	4.41	7.95	18.59	34.42	6.28	11.39	24.65	73.31	22.07	73.76	33.36	180.77
						Integer objective function						
	617.1	547.1	454.7	417.0	530.5	465.4	424.4	397.3	441.7	452.2	429.7	395.1

Table 4: Columns generated while solving R1-instances (25 nodes) with the three configurations.

Config	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
1	70	84	115	131	89	113	122	133	116	109	135	118
2	70	84	116	130	89	110	124	165	124	110	123	122
3	67	85	110	132	86	110	111	165	136	120	128	122

Table 5: Overview of solution parameters for the R1-Solomon examples with 50 nodes

	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
Columns	216	292	393	488	370	353	431	556	365	381	415	506
CPU (*s*)	19.5	56.9	238.6	423.5	61.5	107.2	378.9	935.0	221.1	359.4	419.7	809.2
Integer Obj. Func.	1044.7	909.0	778.1	625.5	902.9	793.1	716.9	623.0	801.5	697.0	715.0	634.3
Linear Obj. Func	1044.7	909.0	776.1	625.5	901.0	793.1	716.9	618.1	777.9	695.2	699.2	619.0
Proven Opt	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No

Table 6: Overview of solution parameters for the R1-Solomon examples with 100 nodes

	R101	R102	R103	R104	R105	R106	R107	R108	R109	R110	R111	R112
Columns	668	787	920	1035	863	964	996	1134	960	951	1051	1008
CPU (*s*)	326.6	586.1	1820	4018	1454	4270	2362	4435	6971	2928	2716	3934
Integer Obj. Func.	1644.4	1468.1	1231.2	1033.5	1373.7	1261.0	1108.6	1008.2	1191.1	1090.6	1071.0	1026.4
Linear Obj. Func	1631.5	1468.1	1209.7	976.7	1348.3	1232.7	1061.9	939.1	1143.4	1064.1	1037.9	977.1
Proven Opt	No	Yes	No	No	No	No	No	No	No	No	No	No

VII. CONCLUSIONS

In this work, we developed a new MILP formulation for the SPPTWCC that is useful in the context of column generation methods designed to solve vehicle routing problems. It is flexible since it can handle almost any structure of constraints while providing good performances on known benchmark problems. Thus, this MILP model can be used as a cornerstone to model more complex and more realistic problems. We have also developed new valid inequalities tailored for such a model and introduced its "lifted" equivalences. The numerical research demonstrates that these inequalities are useful in substantially reducing CPU times.

We think that those developments constitute a starting point for the development of slave route-generator problems that can model a wide arrange of realistic and complex problems. We can see a remarkably constant level of generated columns on problems of a given size but with different solution-morphologies. This obviously means that columns of more complex problems demand higher generation times. Nevertheless, CPU times used to solve small problems with our formulation are higher than CPU times reported in the literature for algorithms based on label-setting procedures. This is because these procedures are able to generate many columns per iteration. Nevertheless, optimality for these columns maybe hard to prove. It is important to highlight that the objective of the proposal is not to compete with algorithms based on label setting procedures but to complement them with this alternative model. It seems that a column generation algorithm involving calling to both types of "routes generators" seems to be the most efficient option. We should first use the label-setting algorithm and then, whenever the branching mechanism demands a few but hard to find columns we should switch to the MILP formulation. This, of course, is an issue for future research. Another area of continuing research should include the design of branching methods tailored to exploit the structure of the new formulation. It seems that branching on assignment variables Y_i is an effective method to keep the search tree bounded. The continuing development of valid inequalities also deserves additional work.

REFERENCES

Barnhart, C., E. Johnson, G. Nemhauser, M. Savelsbergh and P. Vance, "Branch and Price. Column Generation for Solving Huge Integer Programs," *Op. Res.*, **48**, 316-329 (2000).

Cordeau, J.., G. Desaulniers, J. Desrosiers, M. Solomon, and F. Soumis, "VRP with time windows," *The Vehicle Routing Problem,* P. Toth, D. Vigo, eds. SIAM, Philadelphia, PA., **7**, 155-194 (2002).

Desaulniers, G., J. Desrosiers, I. Ioachim, M. Solomon, F. Soumis and D. Villeneuve, "A unified framework for deterministic time constrained vehicle routing and crew scheduling problems," *Fleet Management and Logistics,* T. Crainic, G. Laporte, eds. Kluwer Academic Publisher, Boston, MA., **3**, 57-93 (1998).

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

Gomory, R., "Outline of an algorithm for integer solutions to linear programs," *Bull. AMS*, **64**, 275–278 (1958).

Hoffmann, K., "Combinatorial Optimization: History and Future Challenges," *J of Appl. and Comp. Math.*, **124**, 341-360 (2000).

Houck, D, J. Picard, M. Queyranne and R. Vemuganti, "The travelling salesman problem as a constrained shortest path problem: theory and computational experience," *Op. Res*., **17**, 93-109 (1980).

Irnich, S and G. Desaulniers, "Shortest path problems with resource constraints," *Column Generation*, In G. Desaulniers, J. Desrosiers and M. Solomon, editors. Chapter 2, 33-65 (2005).

Irnich, S and D. Villenueve, "The Shortest-Path Problem with Resource Constraints and k-Cycle Elimination for k > 3. Improving a branch and price algorithm for the VRPTW. INFORMS," *J. of Computing,* **18**, 391-406 (2006).

Larsen, J., *The Dynamic Vehicle Routing Problem*, Ph.D. Thesis. Technical University of Denmark (2001).

Padberg, M., "A note on 0-1 programming," *Operations. Research*, **23**, 833-877 (1979).

Ropke, S and J. Cordeau, "Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows," *Transp. Sci*., **43**, 267-286 (2009).

Solomon, M., "Algorithms for the vehicle routing and scheduling with time window constraints," *Op. Res*., **15**, 254–265 (1987).

van Hoesel, C.P.M. and K.I. Aardal, "Polyhedral techniques in combinatorial optimization I: theory," *Statistica neerlandica: Orgaan van de Vereniging voor Statistiek*, **50**, 3-26 (1996).

Received: June 27, 2011
Accepted: October 24, 2011
Recommended by subject editor: Pedro de Alcântara Pessôa