On Alternative Formulations to the Shortest Path Problem with Time Windows and Capacity Constraints

Ignacio Vitale[1] and Rodolfo Dondo[2]

[1] Facultad de Ingeniería Química, U.N.L, Santiago del Estero 2829, 3000 Santa Fe, Argentina
vitalenacho@gmail.com
[2]Instituto de Desarrollo Tecnológico para la Industria Química (U.N.L. - Conicet), Güemes 3450, 3000 Santa Fe, Argentina
rdondo@santafe-conicet.gov.ar

Abstract. The elementary shortest-path problem with time-windows and capacity constraints is a problem used for solving vehicle-routing and crew-scheduling applications. It 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 and its variations. In the problem there is a directed graph with a source node and a destination node, and each arc has a cost and a vector of weights specifying its requirements of a resource with a finite capacity. A minimum cost source–destination directed path is sought such that the total consumption of the resource does not exceed the capacity. The problem ins NP-hard in the strong sense. We review integer-linear formulation to the problem and compare them in order to study their computational efficiency.

Keywords: shortest path problem, alternative formulations, column generation.

1 Introduction

The elementary shortest-path problem with time-windows and capacity constraints (SPPTWCC) is a problem used for solving vehicle-routing and crew-scheduling applications. It 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 and its variations. The SPPTWCC has been shown to be NP-hard in the strong sense for graphs containing negative cost cycles by Dror [1]. However, the problem remains NP-hard even if the graph is acyclic. The problem is a special case of the resources constrained shortest path problem (RCSPP) and several types of methods have been proposed to solve it. See e.g., [2, 3]. Several solution approaches have been developed for solving the SPPTWCC up to optimality. The main kinds are: (1) label setting algorithms [4, 5, 6]; (2) label correcting algorithms [7, 8, 9]; (3) constraints programming [10]; and (4) methods based on branch-and-bound [11, 12, 13, 14]. Solution approaches proposed in the literature for solving the RCSPP

to optimality are characterized by three main steps: (1) a preprocessing phase, where the dimension of the original network is reduced by eliminating nodes and arcs that cannot be part of any feasible solution; (2) computation of lower and upper bounds; and (3) gap closing step in which the optimal solution is found.

This work compares two already proposed integer-linear formulations of the SPPTWCC with a new one; all of them are solved by branch-and-bound. The purpose is to select the most efficient one for embedding it into column generation algorithms tailored to solve routing problem variations. Numerical examples for testing alternative formulations are solved with such aims.

2 Problem Statement

Consider a route-network represented by an directed 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 "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 $W = [w_1, w_2, \ldots, w_n]$ 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 stand for the earliest possible start-time of the service and parameters b_i state the latest possible start-time of the service at any node. 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 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 prices 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 $[a_i, b_i]$.

3 Formulations

The computational hardness of this problem has inspired researchers to develop creative formulations that are expected to reduce the size of the enumeration branch-and-bound tree and the computation times used to solve the problem. The classical formulation to this problem dating back to [15] here cited as *formulation 1* is written as follows:

Min

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

subject to

$$\sum_{i \in I} w_i \sum_{j \in I \cup p : i \neq j} x_{ij} \leq q \tag{2}$$

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

$$\sum_{k \in I \cup p : k \neq i} x_{ki} = \sum_{k \in I \cup p : k \neq i} x_{ik} \qquad \forall i \in I \tag{4}$$

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

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

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

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

$$x_{ij} \in \{0,1\}$$

$$T_i \geq 0$$

While eq. (1) states the objective function, eq. (2) states the capacity constraint. Constraints (3), (4) and (5) are flow constraints resulting in a path from the depot p to the subset of visited 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 whether arc $(i,j) \in A$ belongs to the optimal path ($x_{ij} = 1$) or not ($x_{ij} = 0$).

In *formulation 2* [14] the problem was re-modelled in order to reduce the number of binary variables. Although the number of binary variables was halved with respect to *formulation 1*, the *formulation 2* extensively uses big-M type constraints leading to poor linear relaxations within the branch-and-bound tree generated to solve it. The problem was remodelled as follows:

Min

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

subject to

$$\sum_{i \in I} w_i Y_i \leq q \qquad \forall i \in I \qquad (10)$$

$$\left\{ \begin{array}{l} C_i \geq c_{ip} \\ T_i \geq t_{ip} \end{array} \right\} \qquad \forall i, j \in I : i < j \qquad (11)$$

$$\left\{ \begin{array}{c} C_j \geq C_i + c_{ij} - M_c \left(1 - S_{ij}\right) - M_c \left(2 - Y_i - Y_j\right) \\ T_j \geq T_i + st_i + t_{ij} - M_T \left(1 - S_{ij}\right) - M_c \left(2 - Y_i - Y_j\right) \\ \\ C_i \geq C_j + c_{ij} - M_c S_{ij} - M_c \left(2 - Y_i - Y_j\right) \\ T_i \geq T_j + st_j + t_{ij} - M_T S_{ij} - M_c \left(2 - Y_i - Y_j\right) \end{array} \right\} \qquad \forall i \in I \qquad (12)$$

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

$$\left\{ \begin{array}{l} a_i \leq T_i \\ T_i \leq b_i \end{array} \right\} \qquad (14)$$

$$TV \leq t^{\max} \qquad (15)$$

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

$$C_i, T_i, CV, TV \geq 0$$

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 time and cost 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) state 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.

In this work, *formulation 3* is developed in a way opposite to formulation *2*; i.e. using more binary variables but aiming to achieve tight linear relaxations. The problem remodelled according to *formulation 3* is written as follows:

Min

$$\sum_{p\in P}\sum_{i\in I}\sum_{l=1} x_{pi}^{l} c_{pi} + \sum_{i\in I}\sum_{j\in I}\sum_{l\in L} x_{ij}^{l}(c_{ij}-\pi_i) + \sum_{p\in P}\sum_{i\in I}\sum_{l=1} x_{ip}^{l}(c_{ip}-\pi_i) \tag{16}$$

subject to

$$\sum_{i\in I} w_i \sum_{j\in I\cup P}\sum_{l\in L} x_{ij}^{l} \le q \tag{17}$$

$$\sum_{p\in P}\sum_{l=1} x_{pi}^{l} \le 1 \qquad \forall i \in I \tag{18}$$

$$\sum_{j\in I}\sum_{l'=l-1} x_{ji}^{l} = \sum_{j\in I}\sum_{l'=l-1} x_{ij}^{l} + \sum_{p\in P} x_{ip}^{l} \qquad \forall i \in I, l>1 \in L \tag{19}$$

$$\sum_{i\in I}\sum_{l=1} x_{pi}^{l} = \sum_{i\in I}\sum_{l\in L} x_{ip}^{l} \qquad \forall p \in P \tag{20}$$

$$\sum_{j\in I}\sum_{l\in L} x_{ij}^{l} + \sum_{p\in P}\sum_{l\in L} x_{ip}^{l} \le 1 \qquad \forall i \in I \tag{21}$$

$$T_l \ge \sum_{p\in P}\sum_{i\in I} x_{pi}^{l} t_{pi} \qquad l=1 \tag{22}$$

$$\left.\begin{array}{l} T_l \ge \sum_{p\in P}\sum_{i\in I}\sum_{l=1} x_{pi}^{l} t_{pi} + \sum_{i\in I}\sum_{j\in I}\sum_{l'=l-1} x_{ij}^{l'}(st_i + t_{ij}) \\ T_l \ge T_j + \sum_{i\in I} x_{ji}^{l'}(st_i + t_{ji}) - M_T\left(1-\sum_{i\in I} x_{ji}^{l'}\right) \end{array}\right\} \qquad l>1\in L \tag{23.a, 23.b}$$

$$T_i \ge T_l - M_T\left(1-\sum_{j\in I} x_{ij}^{l} - \sum_{p\in P} x_{ip}^{l}\right) \qquad \forall i \in I, l \in L \tag{24}$$

$$\left.\begin{array}{l} a_i \le T_i \\ T_i \le b_i \end{array}\right\} \qquad \forall i \in I \tag{25}$$

$$T_i + \sum_{p\in P}\sum_{l\in L} x_{ip}^{l}(st_i + t_{ip}) \le t^{\max} \qquad \forall i \in I \tag{26}$$

$$x_{ij}^{l} \in \{0,1\}$$

$$T_i \ge 0$$

Since set L is used to order visited nodes along the computed path, objective function (16) is defined like in eq. (1) but considering $l \in L$ as a position indicator for the visited node along the optimal path. Eq. (17) is the capacity constraint and eqs. (18), (19) and (20) are flow constraints just like eqs. (3), (4) and (5) but taking into account

the position indicator $l \in L$. Eq. (21) is a returning constraint to the depot. Eq. (22) set the minimum time to reach the first visited position along the path. Eq. (23) computes successive positions along the shortest path. While eq. (23.a) allows a waiting time before a time windows is open, eq. (23.b) doesn't consider such an option and is more useful for instances no constrained by time-windows. Eq. (24) determines the node allocated to each position $l \in L$ along the path. Eqs (25) and (26) are respectively time-windows and maximum routing time constraints.

4 Variable Fixing

Variable fixing is used for reducing the size of a problem. It is a pre-processing technique for tightening the formulation before the actual optimization. It consists on fixing some variables or/and 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-fix some sequencing constraints [14] the following sets are defined:

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

$$Com(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) \le b_j \wedge \left(a_j + st_j + t_{ij}\right) \le b_i \right\} \qquad \forall i \in I \qquad (27)$$

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:

$$\mathrm{Pre}(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) > b_j \wedge \left(a_j + st_j + t_{ij}\right) \le b_i \right\} \qquad \forall i \in I \qquad (28)$$

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:

$$Suc(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) \le b_j \wedge \left(a_j + st_j + t_{ij}\right) > b_i \right\} \qquad \forall i \in I \qquad (29)$$

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:

$$Inc(i) = \left\{ j \in I : \left(a_i + st_i + t_{ij}\right) > b_j \wedge \left(a_j + st_j + t_{ij}\right) > b_i \right\} \qquad \forall i \in I \qquad (30)$$

The use of the above sets allows fixing some variables of the formulations before actually solving it. The following table summarized the pre-fixing decisions that can be made a priori on each formulation by using information provided by the above sets.

Table 1. Variable pre-fixing rules

	Formulation 1	*Formulation 2*	*Formulation 3*
$j \in \Pr e(i)$	$x_{ij} = 0; x_{ji} = 1$	$S_{ij} = 1$	$x^l_{ij} = 0; x^l_{ji} = 1$
$j \in Suc(i)$	$x_{ij} = 1; x_{ji} = 0$	$S_{ij} = 0$	$x^l_{ij} = 1; x^l_{ji} = 0$
$j \in Inc(i)$	$x_{ij} = 0; x_{ji} = 0$	$Y_i + Y_j \leq 1$	$x^l_{ij} = 0; x^l_{ji} = 0$

5 Results and Discussion

The SPPTWCC occurs as a sub-problem used to implicitly generate the set of all feasible routes and schedules in the column-generation (CG) formulation of the vehicle routing problem with time windows (VRPTW) and its realistic variations. This section compares the above formulations first by solving some SPPTWCC instances and later by solving some benchmark VRPTW instances when the formulations are embedded into a simple CG algorithm. The Solomon's 56 benchmark problems [16] has been grouped into C, R and RC categories. C-class problems feature clustered customers. Locations in R-class problems were randomly generated while RC-class problems comprise clustered and randomly located customers. The data set for every category comprises 100 nodes, a depot, similar vehicle capacities but different time-window distributions. Euclidean distances among customers and traveling times are numerically identical. 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 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 $\Pi = [\pi_1, \pi_2, \ldots \pi_{25}]$ reported in [14]. The vector was obtained by generating columns in a CG procedure until reaching the optimal lower bound to the problem. Afterwards, we solved the SPPTWCC using the three above formulation both without (Configuration 1) and with (Configuration 2) prefixed variables accord-

ing rules presented on Table 1. The results are summarized in Tables 2 to 4. In these tables we reported the optimal integer solution (IS), the linear relaxation (LS) of the formulation and the CPU time used to solve the instances both without (Configuration 1) and with (Configuration 2) prefixed variables.

Table 2. Objective function values and CPU times for the resolution by *formulation 1* of the SPPTWCC 25-nodes instances generated from R1 Solomon problems.

Instance	IS	*Configuration 1*		*Configuration 2*	
		CPU (s)	LS	CPU (s)	LS
R101	-0.09	0.12	-257.15	0.16	-0.09
R102	-12.48	2.48	-239.92	2.34	-138.70
R103	-6.65	13.06	-144.25	12.93	-119.44
R104	-1.96	29.53	-115.59	29.13	-111.92
R105	-26.26	0.30	-171.12	0.20	-94.76
R106	-32.83	0.62	-177.84	0.55	-140.65
R107	-5.38	7.36	-106.44	7.35	-86.07
R108	-17.40	4.40	-114.64	4.24	-106.52
R109	-8.09	0.50	-128.00	0.52	-86.00
R110	-3.40	11.34	-140.93	11.45	-121.89
R111	-10.19	4.96	-107.01	5.01	-96.56
R112	-4.23	15.80	-80.29	16.08	-80.29
Average		7.54		7.79	

Table 3. Objective function values and CPU times for the resolution by *formulation 2* of the SPPTWCC 25-nodes instances generated from R1 Solomon problems.

Instance	IS	*Configuration 1*		*Configuration 2*	
		CPU (s)	LS	CPU (s)	LS
R101	-0.09	1.09	-336.67	0.34	-269.09
R102	-12.48	1.51	-313.90	0.81	-249.29
R103	-6.65	14.21	-222.04	9.20	-223.84
R104	-1.96	39.91	-228.23	38.13	-201.45
R105	-26.26	1.90	-258.79	0.58	-245.84
R106	-32.83	4.51	-275.74	1.15	-242.83
R107	-5.38	37.99	-222.39	28.91	-208.13
R108	-17.40	53.57	-210.49	35.35	-204.97
R109	-8.09	8.38	-244.23	2.00	-223.26
R110	-3.40	24.74	-237.90	26.97	-237.90
R111	-10.19	27.44	-231.90	13.21	-215.58
R112	-4.23	1200*	-192.61	1200*	-192.61
Average		117.94		113.05	

From the tables we can conclude that: (1) formulation 3 provides the tightest lower bound both with and without pre-fixing rules; (2) tight bounds translate, in average, in shorter CPU times. Consequently, achieving good bounds seems more important than lowering the number of binary variables used to model the problem. (3) Pre-fixing rules have a sizable effect in reducing CPU times in all formulations.

Table 4. Objective function values and CPU times for the resolution by *formulation 3* of the SPPTWCC 25-nodes instances generated from R1 Solomon problems.

Instance	IS	*Configuration 1*		*Configuration 2*	
		CPU (s)	LS	CPU (s)	LS
R101	-0.09	0.91	-167.46	0.14	-0.09
R102	-12.48	2.62	-110.39	1.95	-65.52
R103	-6.65	2.36	-64.56	2.01	-46.95
R104	-1.96	12.81	-54.28	11.83	-49.86
R105	-26.26	1.17	-83.32	0.59	-55.90
R106	-32.83	2.20	-87.21	1.67	-69.82
R107	-5.38	17.75	-51.03	19.33	-44.09
R108	-17.40	2.76	-55.68	2.72	-47.06
R109	-8.09	1.36	-80.24	0.59	-60.44
R110	-3.40	61.67	-82.19	39.41	-78.76
R111	-10.19	15.81	-57.99	4.74	-55.04
R112	-4.23	2.37	-37.48	2.59	-37.48
		10.31		7.29	

Afterwards, we inserted the above formulations of the SPPTWCC into a simple column generation procedure written in GAMS [17] in order to solve Solomon's R1 instances with 25 nodes (See Figure 1). Since feasible columns may run into billions and it is not possible to realistically generate all columns, the column generation approach handles this by implicitly considering all columns trough the solution of the linear relaxation of the SPP. A portion of all possible routes is enumerated and the resulting linear relaxation with this partial set is solved. The solution to this linear problem is used to determine if there are any route not included that can reduce the objective function value. Using the value of the optimal dual variables with respect to the partial routes set, new routes are generated and incorporated and the linear relaxation is solved again. This continues until one can show that an optimal solution to the linear problem cannot be improved with the addition of another route. The logic of this algorithm is illustrated on Figure 1. We also collected al solutions with negative reduced costs generated per iteration via the *Solnpool* CPLEX procedure. Al examples were solved in a 2.0 GHz 16 GRAM PC. The purpose was to compare times consumed to reach optimality. Results are summarized in Table 5. The table, for all used formulations, reports the best found integer solution and the corresponding linear solution on the pool of generated columns. It also reports the size of the columns pool and the CPU time consume by the CG algorithm to solve each instance. From the information summarized in Table 4 it can be concluded that faster solutions to the slave SPPTWCC subproblem doesn't automatically translate as faster resolution times via CG for the routing problem. In average, formulation 3 performed slightly better than the other ones and formulation 2 is, by a little, the worst one. On the other hand, as formulation 2 collects more generated routes per iteration, it compensates its slow convergence speed to prove optimality.

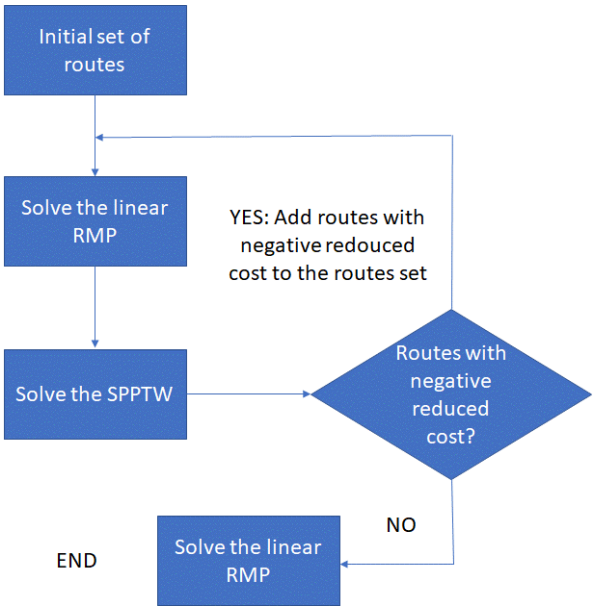

Figure 1: The incomplete optimization algorithm.

Table 5. Solution data for R1 Solomon's instances with 25 nodes.

Instance	Integer solution	Linear solution	Columns	CPU times (s)
		Formulation 1		
R101	618.4	618.4	303	8.9
R102	549.7	549.7	356	40.4
R103	455.8	455.8	457	223.3
R104	418.1	418.1	453	540.6
R105	531.7	531.7	363	16.2
R106	466.6	462.0	401	50.3
R107	435.4	427.6	436	163.3
R108	404.4	403.6	442	311.9
R109	442.8	442.8	709	43.4
R110	448.4	441.2	466	185.4
R111	446.2	429.0	501	127.7
R112	409.6	394.3	429	312.2
Average			443	168.6
		Formulation 2		
R101	618.4	618.4	165	5.1
R102	549.0	547.5	218	10.5
R103	455.8	455.8	327	86.2
R104	418.1	418.1	462	424.9
R105	531.9	531.9	234	43.6
R106	466.6	466.2	310	36.8
R107	445.4	427.7	544	351.1
R108	424.2	403.7	548	553.1
R109	442.7	442.7	536	79.8
R110	449.3	441.1	321	336.6
R111	451.3	428.9	389	180.4
R112	413.4	397.2	542	461.2
Average			383	255.8

		Formulation 3		
R101	618.4	618.4	163	7.9
R102	548.2	547.5	225	87.5
R103	455.8	455.8	254	225.8
R104	418.1	418.1	383	234.5
R105	531.6	531.6	201	15.4
R106	471.2	465.6	295	34.1
R107	429.3	427.6	348	215.4
R108	404.9	403.6	303	225.3
R109	442.7	442.7	437	104.2
R110	445.3	445.3	289	252.9
R111	429.8	429.8	412	231.6
R112	410.5	394.2	267	201.5
Average			298	153.1

Also, it seems that formulation 3 performs better in loosely time-windows constrained problems while formulation 2 performs better in tightly time-windows constrained instances. Formulation 1 performs quite well in the widest spam of instances-topologies.

5 Conclusions

In this work, we developed a new MILP formulation for the SPPTWCC and performed some numerical studies to compare its computational efficiency with respect of two previously presented formulations. The problem is useful in the context of CG methods designed to solve vehicle routing problems and its realistic variations. It is important to highlight that the MILP formulations of the slave subproblem usually don't compete with algorithms based on label setting procedures but to complement them in a CG algorithm calling both types of "routes generators". 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 best MILP formulation.

We can conclude that there is not "a best" MILP formulation and the performance of alternative models depends on tightness and number of time windows. I.e. in average formulation 3 performed slightly better than the other ones and formulation 2 is, by a little, the worst one. Formulation 1 collects more generated routes per iteration. Formulation 3 performs better in loosely time-windows constrained problems while formulation 2 performs better in tightly time-windows constrained instances. Formulation 1 performs quite well in the widest spam of instances-topologies.

References

1. Dror M. Note on the complexity of the shortest path models for column generation in VRPTW. Oper. Res. (1994);42:977–8.

2. Irnich S, and Desaulniers G. Shortest path problems with resource constraints. In: Desaulniers G,Desrosiers J,Solomon M, editors. Columngeneration. US: Springer (2005).
3. Pugliese L, snd GuerrieroF. A survey of resource constrained shortest path problems: exact solution approaches.Networks (2013) ;62:183–200.
4. Desrochers M. and Soumis, F. A generalized permanent labelling algorithm for the shortest path problems with time windows, INFOR 26 (1988), 193–214.
5. Gallo G. and Pallottino S. Shortest path algorithms, Ann Oper Res 13 (1988), 1–79.
6. Denardo E. and Fox B. Shortest-route methods: reaching, pruning and buckets, Oper Res 27 (1979), 161–186.
7. Di Puglia Pugliese L. and Guerriero F. A computational study of solution approaches for the resource constrained elementary shortest path problem, Ann Oper Res (2012) 201, 131–157.
8. Powell W. and Chen Z. A generalized threshold algorithm for the shortest path problem with time windows. Network design: Connectivity and facilities, P.M. Panalos and D. Du (Editors), American Mathematical Soc., Providence, RI, (1998) 303–318.
9. Glover F., Glover R., and Klingman D. The threshold shortest path algorithm, Networks 14 (1984), 25–36.
10. Rousseau L., Gendreau M. and Pesant G. Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of heuristics. (2002) 8 (1), 43-58.
11. Beasley J. and Christofides N. An algorithm for the resource constrained shortest path problem, Networks 19 (1989), 379–394.
12. Carlyle W., Royset J., and Wood R. Lagrangian relaxation and enumeration for solving constrained shortest path problems, Networks 52 (2008), 256–270.
13. Muhandiramge R. and Boland N.. Simultaneous solution of Lagrangean dual problems interleaved with preprocessing for the weight constrained shortest path problem, Networks 53 (2009), 358–381.
14. Dondo, R. A New MILP Formulation to the Shortest Path Problem with Time Windows and Capacity Constraints. Latin American Applied Research, (2012) 42, 257-265.
15. Desrochers M.; Desrosiers J. and Solomon M. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research, (1992) 40 (2). 342-354.
16. Solomon, M.. Algorithms for the Vehicle Routing and Scheduling Problem With Time Window Constraints. Opns. Res. (1987) 35, 254-265.
17. Kalvelagen, E. Columns generation with GAMS (2011). Downloaded from http://amsterdamoptimization.com/pdf/colgen.pdf