Integer programming formulations for the time-dependent elementary shortest path problem with resource constraints

Gonzalo Lera-Romero [1]

Departamento de Computación
Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires
CABA, Argentina

Juan José Miranda-Bront [2]

Universidad Torcuato Di Tella
Consejo Nacional de Investigaciones Científicas y Técnicas
CABA, Argentina

Abstract

In this paper we study the Time-Dependent Elementary Shortest Path Problem with Resource Constraints (TDESPPRC). We consider two integer programming formulations which exploit the characteristics of the time-dependent travel time function. Two exact algorithms based on these formulations are developed and evaluated on benchmark instances from the literature. Preliminary experiments show that the approach has potential to be used within a Branch and Price algorithm.

Keywords: Elementary Shortest Path, Integer Programming, Time-Dependent Travel Times

1 Introduction

Congestion in large cities and populated areas is one of the major challenges in urban logistics, becoming one of the major issues in city planning and last-

[1] Email: `gleraromero@dc.uba.ar`
[2] Email: `jmiranda@utdt.edu`

mile logistics due to its economic, social and environmental impact. Most of the research devoted to the Vehicle Routing Problem (VRP) considers that travel times between any two locations are fixed along the time horizon. In the last few years, there has been a trend to enrich these models by incorporating more complex travel time functions to capture the effect of congestion, known as Time-Dependent VRPs (TDVRP, see, e.g., Gendreau et al. [2], for an updated survey). Incorporating the congestion explicitly at a tactical level is a key aspect of modern Decision Support Systems (DSS) in order to obtain distribution plans that are representative of the real-life operations.

Exact approaches for multi-vehicle variants of the classical VRP generally resort to Integer Linear Programming (ILP) and decomposition techniques, resulting in *Branch and Price* (BP) and *Branch-cut and price* (BCP) algorithms. These approaches produce the best results in general, and one of their key ingredients rely on the pricing problem within the column generation algorithm to solve the LP relaxation at each node of the enumeration tree. When tackling a TDVRP, the pricing problem can be formulated as a Time-Dependent Elementary Shortest Path Problem with Resource Constraints (TDESPPRC). In this sense, to the best of our knowledge, most of the research has been devoted to variants of the TDVRP with the presence of time windows. This is the case of Dabia et al. [1] for the capacitated TDVRP with time windows (TDVRP-TW) and Sun et al. [6], which further considers the TDESPPRC with time windows and precedence constraints. In both cases, the problem is tackled using dynamic programming (DP).

In this paper, we consider the capacitated TDVRP without the time windows constraints. From an algorithmic perspective, under this setting the traditional DP and labeling algorithms are not expected to produce good results, especially if no limiting resource is present. In this sense, we build upon the works of Taccari [7] and Jepsen et al. [4] on the ESPPRC with time-independent travel times, where they consider ILP formulations to address the ESPPRC. We propose a strengthened version of the ILP formulation proposed in Sun et al. [6], studied also by Montero et al. [5] for the Time-Dependent Traveling Salesman Problem with Time Windows (TDTSP-TW). In addition, we present a tailored formulation which reduces the number of variables and constraints in the model by exploiting the structure of the travel time function. We develop a *Branch and Cut* (BC) algorithm including the well-known *cutset inequalities*. To the best of our knowledge, we conduct the first computational experiments over a set of TDESPPRC benchmark instances without time windows, comparing both formulations and evaluating the feasibility of using them within a BP algorithm.

2 Problem definition

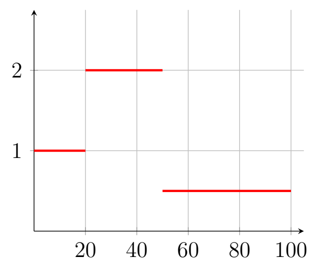
In this section we define the TDESPPRC in its general fashion. We refer the reader to Dabia et al. [1] for details regarding its incorporation within a BP framework based on a set partitioning formulation for the TDVRP.

The time-dependent network is defined as follows. Let $D = (V, A)$ be a digraph, with $V = \{v_s, v_e, 1, 2, \dots, n\}$ the set of vertices representing the customers, and A the set of arcs representing their connections. Two distinguished vertices, v_s and v_e, denote the starting and ending point of the path, respectively. Typically, these two vertices represent the depot and therefore no incoming arcs to v_s and outgoing arcs from v_e are considered. Each vertex $i \in V$ has an associated demand q_i, a service time s_i and a profit p_i which is collected when visiting customer i. Given the context, we set $p_{v_s}, q_{v_s}, s_{v_s}$ and $p_{v_e}, q_{v_e}, s_{v_e}$ to 0. We consider a single vehicle with capacity Q.

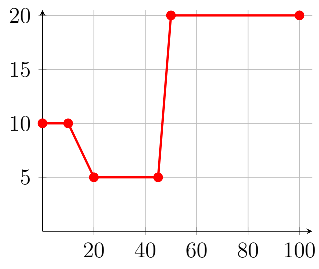
We consider the travel time model proposed in Ichoua et al. [3]. Each arc $(i, j) \in A$ has an associated travel distance L_{ij}. There is a time horizon $[0, T]$ in which operations take place, which is partitioned into H intervals $[T_h, T_{h+1}]$, $h = 0, \dots, H-1$. The average travel speed for each arc (i, j) during time interval $[T_h, T_{h+1}]$ is assumed to be known, and is denoted by v_{ijh}. This partition is usually referred as *speed profiles*. The model captures the variable travel times by combining the distance to be traveled for an arc with its different travel speeds defined for the arc depending on the starting time of the trip. Formally, we let $\tau_{ij}(t)$ denote the time-dependent travel time for arc $(i, j) \in A$ when departing at i at time $t \in [0, T]$, which is computed using the Algorithm 1 from Ichoua et al. [3]. It can be easily shown that the resulting travel time function $\tau_{ij}(t)$ is piecewise linear. Figure 1 illustrates the relation between the travel speed and the travel time function. It is important to remark that the travel time function satisfies the *First-In First-Out* property.

The definition of the travel time function $\tau_{ij}(t)$ induces a new partition of the planning horizon for each edge $(i, j) \in A$. The limits defining this partition can be computed by a simple algorithm and are usually referred as *travel time breakpoints*. For the sake of simplicity, we follow the notation introduced in Montero et al. [5]. Let $T^{ij} = \{T_1^{ij}, \dots, T_M^{ij}\}$ be the this new partition for arc $(i, j) \in A$, where $T_m^{ij} = [w_m, w_{m+1}]$. We remark that the value of M may be different among arcs. Within each of these intervals, $\tau_{ij}(t)$ remains linear.

Similarly to Dabia et al. [1] and Sun et al. [6], we focus on the path duration instead of the makespan. Consequently, the vehicle is allowed to delay its departure from v_s to reduce the duration of the route. Given a path P, let $\Delta(P)$ denote the duration. Therefore, the TDESPPRC involves finding an



(a) Travel speed function for (i, j).



(b) Travel time function $\tau_{ij}(t)$.

Fig. 1. Time-dependent travel time model ($L_{ij} = 10$).

elementary path $P = (v_s, v_1, \ldots, v_k, v_s)$ minimizing $\Delta(P) - \sum_{v \in P} p_v$, starting and ending within the planning horizon and not exceeding the capacity.

Finally, we introduce some further notation used throughout the paper. We let $\delta^-(j) = \{(i,j) : (i,j) \in A\}$ and $\delta^+(i) = \{(i,j) : (i,j) \in A\}$. In addition, we denote with $\tau_{ij}(t) = \theta^m_{ij} t + \eta^m_{ij}$, for $t \in T^{ij}_m$, to the linear travel time function in time period $m \in T^{ij}$.

3 ILP formulations

We next present a formulation derived from the one presented in Sun et al. [6] and considered also in Montero et al. [5]. Binary variables x^m_{ij} indicates the path travels from i to j departing from i within travel time period $m \in T^{ij}$. We implicitly define traditional binary variables x_{ij} in terms of x^m_{ij}. Binary variable y_i takes value 1 iff vertex $i \in V$ is visited. In addition, for each vertex a non-negative variable t_i indicates the departure time from $i \in V$, if visited by the path. The value of t_i is decomposed to account for the departure time for each specific arc and travel time period, if any is selected. Thus, we define variables $t^m_{ij} = t_i$ if $x^m_{ij} = 1$, and $t^m_{ij} = 0$ otherwise.

$$\min \quad t_{v_e} - t_{v_s} - \sum_{i \in V \setminus \{v_s, v_e\}} p_i y_i \tag{1}$$

$$\text{s.t.:} \quad \sum_{m=0}^{|T^m_{ij}|} x^m_{ij} = x_{ij} \qquad (i,j) \in E \tag{2}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} = y_j \qquad j \in V \setminus \{v_s\} \tag{3}$$

$$\sum_{(v_s,j) \in \delta^+(v_s)} x_{v_s j} = \sum_{i \in \delta^-(v_e)} x_{i v_e} = 1 \tag{4}$$

$$\sum_{(i,k)\in\delta^-(k)} x_{ik} - \sum_{(k,j)\in\delta^+(k)} x_{kj} = 0 \qquad k \in V\backslash\{v_s, v_e\} \qquad (5)$$

$$\sum_{(i,j)\in\delta^-(j)} \sum_{m=0}^{|T_{ij}^m|} (1+\theta_{ij}^m)t_{ij}^m + \eta_{ij}^m x_{ij}^m + s_j x_{ij}^m = t_j \qquad j \in V\backslash\{v_s\} \qquad (6)$$

$$\sum_{(i,j)\in\delta^+(i)} \sum_{m=0}^{|T_{ij}^m|} t_{ij}^m = t_i \qquad i \in V\backslash\{v_e\} \qquad (7)$$

$$w_{ij}^m x_{ij}^m \leq t_{ij}^m \leq w_{ij}^{m+1} x_{ij}^m \qquad (i,j) \in E, m \in T_{ij}^m \qquad (8)$$

$$\sum_{i\in V\backslash\{v_s,v_e\}} q_i y_i \leq Q \qquad (9)$$

$$t_{ij}^m, t_i \geq 0 \qquad (i,j) \in E, m \in T_{ij}^m \qquad (10)$$

$$y_i, x_{ij}, x_{ij}^m \in \{0,1\} \qquad (i,j) \in E, m \in T_{ij}^m \qquad (11)$$

The objective function (1) minimizes the duration while maximizing the profits collected. Constraints (2) - (5) define the x_{ij} and y_i variables in terms of x_{ij}^m and establish that the vehicle must depart from v_s, finish at v_e, and impose the classical flow conservation constraints. Constraints (6) compute the ready-time for vertex j, if visited. We remark that this constraint is a strengthened version of the ones considered in [6,5]. Constraints (7) express variables t_i in terms of t_{ij}^m, and constraints (8) impose the relation between variables t_{ij}^m and x_{ij}^m. Finally, constraints (10) and (11) establish the domain for all variables.

The absence of time windows and precedence constraints limits, compared to other variants, the possibility of applying effective preprocessing techniques to reduce the size of the ILP formulation. This increment in the size of the model has been observed in preliminary experiments and is aligned with the results reported in Montero et al. [5] where the preprocessing phase is one of the key ingredients of the algorithm.

Depending on the definition of the speed profile and the length of the arc, L_{ij}, we noted that some of the travel time periods correspond a trips starting and ending within the same travel speed interval T_h, meaning that there is no boundary crossing between consecutive travel speed intervals. As a result, in these cases the travel time function for some of the travel time intervals $T_m^{ij} \in T^{ij}$ becomes constant, and therefore has $\theta_{ij}^m = 0$. This is the case for the first, third and fifth time intervals in Figure 1b.

For each $(i,j) \in A$, let $\hat{T}^{ij} = \{m \in T^{ij} : \theta_{ij}^m = 0\}$ be the subset of intervals where the travel time function is constant. We further define a new set of variables $\hat{t}_{ij} = t_i$ if arc (i,j) is traversed departing from i within one of the time intervals $m \in \hat{T}^{ij}$, and $\hat{t}_{ij} = 0$ otherwise. Formally, we can rewrite

$\hat{t}_{ij} = \sum_{m \in \hat{T}^{ij}} t_{ij}^m$.

We first rewrite constraints (6) by including the definition of $\hat{t}_{ij}$. Then, the constraints can be rewritten as

$$\sum_{i \in \delta^-(j)} \left(\hat{t}_{ij} + \sum_{m \in T^{ij} \setminus \hat{T}^{ij}} (1 + \theta_{ij}^m) t_{ij}^m + \eta_{ij}^m x_{ij}^m + s_j x_{ij}^m \right) = t_j, \quad j \in V \setminus \{v_s\}. \quad (12)$$

Similarly, we rewrite the decomposition of variables t_i as

$$\sum_{j \in \delta^+(i)} \left(\hat{t}_{ij} + \sum_{m \in T^{ij} \setminus \hat{T}^{ij}} t_{ij}^m \right) = t_i, \quad i \in V \setminus \{v_e\}. \quad (13)$$

Finally, constraints (8) can be adapted as well to incorporate the new definitions. First, we remark that for $m \in T^{ij} \setminus \hat{T}^{ij}$ constraints (8) should be maintained as part of the formulation. However, for $m \in \hat{T}^{ij}$, we can replace them by the following aggregated constraints

$$\sum_{m \in \hat{T}^{ij}} w_{ij}^m x_{ij}^m \leq \hat{t}_{ij} \leq \sum_{m \in \hat{T}^{ij}} w_{ij}^{m+1} x_{ij}^m, \quad (i,j) \in E. \quad (14)$$

Recall that at most one variable x_{ij}^m, $m \in \hat{T}^{ij}$, can take value 1 and therefore $\hat{t}_{ij}$ captures the correct departure time.

These definitions allow to reduce the size of the resulting formulation by replacing all variables t_{ij}^m for $m \in \hat{T}^{ij}$ by only one variable $\hat{t}_{ij}$. A similar reduction is obtained in terms of the number of restrictions by considering constraints (14). It must be noted that the impact of the reduction depends on the granularity of the speed profiles and the distances to be travelled. This is the case on the benchmark instances available in the related literature, which are defined aiming to capture the standard rush hour congestion in large cities. In addition, we emphasize that this also appears as a reasonable setting for tactical planning, which motivates and justifies our approach.

4 Computational results

Computational experiments have been conducted to evaluate the performance of both formulations presented in this paper. The algorithms are coded in C++, using g++ 4.8.4 and an Ubuntu Linux 16.04 LTS, in addition to CPLEX 12.4 Callable Library as an ILP solver. The experiments were run on a Workstation with an Intel Core i7-2600 3.4GHz CPU and 16Gb of RAM.

	TTBF					TTBF-Compact				
Instance	#vars	#cons	Opt	Time	Nodes	#vars	#cons	Opt	Time	Nodes
C101_25	12354	12351	5	121.30	14948.80	7804	9751	5	84.59	15806.80
C201_25	12353	12351	4	1688.71	268141.2	7803	9751	5	941.32	308258.80
R101_25	11862	11859	5	202.15	29755.40	9732	10503	5	107.57	27912.20
R201_25	12345	12343	4	11.84	3935.25	8027	9859	5	638.69	172679.20
RC101_25	11384	11383	5	251.21	26668.40	9332	10096	5	144.13	22636.20
RC201_25	12353	12351	5	347.58	52348.40	8243	9971	5	48.50	26535.20
Average over solved instances				402.91	60961				146.68	47474

Table 1
Average aggregated results for instances with 25 customers.

Regarding the methods, we consider a BC algorithm for each of the ILP formulation presented in the previous section. The standard formulation (1) - (10) is named as TTBF while the new formulation obtained by considering constraints (12) - (14) is named as TTBF-Compact. For both algorithms, a cutting plane algorithm at the root node including the *cutset inequalities* (see, e.g., Taccari [7]) and an initial heuristic are considered. Thus, the algorithms can be seen as a *Cut and Branch*. The algorithms are evaluated over a subset of the benchmark instances with 25 customers proposed in Dabia et al. [1] for the TDVRPTW, constructed by incorporating the time-dependency to the well-known Solomon instances, but discarding the time windows. We report the results over 30 instances derived from the following instances: *C101_25, C201_25, R101_25, R201_25, RC101_25, RC201_25.* For each of these, the LP relaxation of the set partitioning formulation for the TDVRP is tackled using column generation, and five instances of the column generation subproblem at different points in the enumeration of columns are generated using the corresponding dual variables (which represent the profits in the TDESPPRC). A time limit of two hours is imposed for the running times.

Table 1 presents the average results over each of the original instances. For each algorithm, we report the number of variables (#vars) and constraints (#cons) in the resulting formulation, as well as the number of instances solved to optimallity (Opt), the computational time in seconds (Time) and the number of nodes (Nodes) enumerated in the BC tree.

The main message in Table 1 is that TTBF-Compact obtains better results than TTBF on average, solving all instances within the imposed time limit. Furthermore, if restricted to the 28 instances solved by both algorithms, TTBF requires on average 402.91 seconds while TTBF-Compact only requires 146.68 seconds, approximately half of the time. As expected, we can observe a significant reduction in the number of variables and constraints in TTBF-Compact compared to TTBF.

5 Conclusions

In this paper, we consider the TDESPPRC formulation proposed by Sun et al. [6] for the case with time windows and precedence constraints and adapted it to the case where the only resource is the capacity of the vehicle. Two formulations have been evaluated on benchmark instances, and preliminary results show that the approach has potential to be included as part of a BP algorithm. As future research, it would be interesting to derive families of valid inequalities in order to improve the LP relaxation and evaluate its behaviour within a column generation algorithm.

References

[1] Dabia, S., S. Ropke, T. van Woensel and T. D. Kok, *Branch and price for the time-dependent vehicle routing problem with time windows*, Transportation Science **47** (2013), pp. 380–396.

[2] Gendreau, M., G. Ghiani and E. Guerriero, *Time-dependent routing problems: A review*, Computers & Operations Research **64** (2015), pp. 189–197.

[3] Ichoua, S., M. Gendreau and J.-Y. Potvin, *Vehicle dispatching with time-dependent travel times*, European journal of operational research **144** (2003), pp. 379–396.

[4] Jepsen, M. K., B. Petersen, S. Spoorendonk and D. Pisinger, *A branch-and-cut algorithm for the capacitated profitable tour problem*, Discrete Optimization **14** (2014), pp. 78–96.

[5] Montero, A., I. Méndez-Díaz and J. J. Miranda-Bront, *An integer programming approach for the time-dependent traveling salesman problem with time windows*, Computers & Operations Research **88** (2017), pp. 280 – 289.

[6] Sun, P., L. P. Veelenturf, S. Dabia and T. V. Woensel, *The time-dependent capacitated profitable tour problem with time windows and precedence constraints*, European Journal of Operational Research **264** (2018), pp. 1058 – 1073.

[7] Taccari, L., *Integer programming formulations for the elementary shortest path problem*, European Journal of Operational Research **252** (2016), pp. 122–130.