Computers and Operations Research 88 (2017) 280–289

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

An integer programming approach for the time-dependent traveling salesman problem with time windows

Agustín Montero[a,c,*], Isabel Méndez-Díaz[a,d], Juan José Miranda-Bront[a,b,c,*]

[a] Departamento de Computación, FCEyN, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, C1428EGA, CABA, Argentina
[b] Universidad Torcuato Di Tella, Av. Figueroa Alcorta 7350, C1428BCW, CABA, Argentina
[c] Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
[d] CONICET-Universidad de Buenos Aires. Instituto de Investigación en Ciencias de la Computación (ICC). Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 15 June 2016
Revised 9 May 2017
Accepted 26 June 2017
Available online 4 July 2017

Keywords:
Time-dependent TSP
Time windows
Integer linear programming
Branch-and-Cut

ABSTRACT

Congestion in large cities and populated areas is one of the major challenges in urban logistics, and should be addressed at different planning and operational levels. The Time Dependent Travelling Salesman Problem (TDTSP) is a generalization of the well known Traveling Salesman Problem (TSP) where the travel times are not assumed to be constant along the day. The motivation to consider the time dependency factor is that it enables to have better approximations to many problems arising from practice. In this paper, we consider the Time-Dependent Traveling Salesman Problem with Time Windows (TDTSP-TW), where the time dependence is captured by considering variable average travel speeds. We propose an Integer Linear Programming model for the problem and develop an exact algorithm, which is compared on benchmark instances with another approach from the related literature. The results show that the approach is able to solve instances with up to 40 customers.

1. Introduction and literature review

The use of the transportation infrastructure and the impact of congestion have become one of the major issues in city planning and urban logistics. Projections indicate that this effect is expected to worsen in the medium and long term. Therefore, the current traffic situation as well as the projected traffic scenarios are likely to have, if not addressed correctly, a negative impact from a social, economic and a environmental standpoint.

Most of the research related to the Vehicle Routing Problem (VRP) considers that the travel time between two locations are fixed along the time horizon. An updated description of variants and methods can be found in Toth and Vigo (2014). In the last few years, there has been a trend to enrich these models by incorporating more complex travel time functions, aiming to obtain solutions that are closer to real-world operations. These models are particularly useful for urban logistics, where congestion may produce significant variations in travel times during different moments of the day. For instance, last mile deliveries, which are estimated to account of an important percentage of the total delivery costs, could be significantly improved by more realistic approaches, translating into a better service and a more efficient use of the resources.

Time-Dependent Vehicle Routing Problems (TDVRPs) is the name given to a family of problems that generalize the classical VRPs by considering more complex travel time and cost functions, generally by incorporating some variability depending on the moment of the day an arc is traversed. A recent survey on TDVRP variants is available in Gendreau et al. (2015), covering exact and heuristic algorithms. Commercial applications including traffic information are, to the best of our knowledge, quite scarce in practice. Google Maps and Waze provide detailed directions including traffic information, but limited to the *quickest path* between two points.

One of the variants that received some attention in the last decade is the so-called *Time-Dependent Traveling Salesman Problem* (TDTSP), which considers only one vehicle with infinite capacity. Therefore, the problem reduces to find a Hamiltonian tour at minimum total cost, while accounting for some particular travel time function. In this context, the name TDTSP has been used to refer to problems considering different travel time functions. The simplest generalization is the variant of the TDTSP considered in Picard and Queyranne (1978), which has applications within scheduling contexts and generalizes the well-known Traveling Deliveryman Problem (see, e.g., Fischetti et al., 1993; Lucena, 1990; Méndez-Díaz et al., 2008). The improvement with respect to the traditional TSP

* Corresponding authors.
E-mail addresses: aimontero@dc.uba.ar (A. Montero), imendez@dc.uba.ar (I. Méndez-Díaz), jmiranda@dc.uba.ar, jmiranda@utdt.edu (J.J. Miranda-Bront).

http://dx.doi.org/10.1016/j.cor.2017.06.026
0305-0548/

is that it considers that the travel cost function between two cities depends not only on the distance, but also on the position of the arc in the tour. Exact approaches for this problem can be found in Gouveia and Voß (1995), Abeledo et al. (2012), Miranda-Bront et al. (2013) and Godinho et al. (2014), where instances with up to 100 customers can be solved within reasonable computing times. To the best of our knowledge, the best exact approach in the literature is the Branch-Price and Cut (BPC) proposed in Abeledo et al. (2012).

A different approach is proposed in Malandraki and Daskin (1992), where the travel time between two cities depends on the moment of the day in which the arc is traversed. For this purpose, the authors proposed partitioning the time horizon in different *time periods* and the travel time is defined as a step function over these periods. This allows the model to capture, at least partially, the effect of congestion in different moments of the day. Exact approaches for variants of this problem with minor modifications (i.e., different objective functions and operational constraints) can be found in Stecco et al. (2008), Albiach et al. (2008), Méndez-Dıaz et al. (2011), Miranda-Bront (2012) and Melgarejo et al. (2015). Regarding applications, Furini et al. (2015) formulate a TDTSP to model an aircraft sequencing problem.

One of the major objections to the above model is that the travel times do not necessarily satisfy the FIFO (*First In, First Out*) condition, which is usually a desired property for the network from a vehicle routing perspective. To overcome this difficulty, Ichoua et al. (2003) builds upon the model proposed by Hill and Benton (1992) and propose a similar setting as in Malandraki and Daskin (1992) but for the average travel speed within each period. The resulting travel times are computed based on the departure instant from the origin customer, assuming that the distance of the trip is traveled at the average speed and that when crossing the boundaries between consecutive time periods, the average speed is adjusted. This model is able to capture the time dependency while satisfying the FIFO condition on the travel times.

The model proposed by Ichoua et al. (2003) has recently caught the attention of many researchers. Cordeau et al. (2012) tackles the TDTSP with the objective to minimize the makespan. They study some of the properties of the travel time function, including the computation of a lower bound obtained by solving an auxiliary TSP with constant travel times. They show that the bound is tight depending on some parameters related to the travel speed definitions and that, under some particular settings, the solution of the auxiliary TSP is indeed optimal. They also propose a Branch and Cut (BC) algorithm and are able to solve instances with up to 40 vertices. Ghiani and Guerriero (2014) further exploit some of the properties of the travel time function, and study its generality. In a follow up paper, Arigliano et al. (2015) extend the ideas proposed in Cordeau et al. (2012) to the TDTSP with Time Windows (TDTSP-TW). However, the results obtained are not as good as for the TDTSP. A BC algorithm is evaluated on instances with up to 40 clients, obtaining mixed results.

Multi-vehicle versions of the TDVRP have also been tackled by exact algorithms that consider the model proposed in Ichoua et al. (2003). Dabia et al. (2013) consider the TDVRP with time windows with the objective of minimizing the overall duration instead of the makespan. They propose a set partitioning model and develop a Branch and Price (BP) algorithm, where the column generation subproblem is tackled by means of a tailored labeling algorithm. The authors conduct experiments on instances of different sizes, showing that the approach is able to solve consistently instances with 25 vertices and some of the ones having 100 customers. Related to this research is the work by Sun et al. (2015), where a profitable TDTSP with time windows and precedence constraints are considered. Indeed, this particular variant arises as the column generation subproblem of a TDVRP with time-windows and precedence constraints. They propose an Integer Linear Programming (ILP) model for the problem, which is not studied in detail due to its performance in standard commercial solvers, and resort to dynamic programming techniques.

In this paper we tackle the version TDTSP-TW considered also in Arigliano et al. (2015). The contribution of this paper is two fold. Firstly, we propose an alternative approach for the TDTSP-TW that builds on the ILP formulation proposed by Sun et al. (2015). This model is used to develop an exact algorithm following a Branch-and-Cut scheme (BC. We included several initial heuristics, preprocessing rules and incorporate several families of valid inequalities, which are used as cuts, in order to improve the overall computational times of the algorithm. Secondly, we evaluate our approach on benchmark instances and compare our results with two sets of instances proposed by Arigliano et al. (2015). To the best of our knowledge, this is the first comparison of two exact approaches for the TDTSP-TW, establishing a baseline for future approaches for the TDTSP-TW and related problems and opening the discussion regarding formulations, algorithms and benchmark instances.

The rest of the paper is organized as follows. In Section 2 we introduce the notation used throughout the paper and provide the detailed definition of the problem. In Section 3 we describe with more details some of the developments proposed for the TDTSP and TDTSP-TW with time-dependent travel speeds, and present a new formulation for the TDTSP-TW using the ideas proposed in Sun et al. (2015). Section 4 describes the details of the BC algorithm based on this formulation. Computational results are shown in Section 5 and finally we conclude and state some future research directions in Section 6.

2. Problem definition

In this section we present the definitions and the basic properties of the TDTSP-TW with the travel time model proposed in Ichoua et al. (2003).

For the definition of the network, consider a digraph $D = (V, A)$, with $V = \{0, 1, \ldots, n, n+1\}$ the set of vertices and A the set of arcs. Vertices 0 and $n+1$ represent the depot, for which we do not consider the incoming and outgoing arcs, respectively. We denote by $V_0 = V \backslash \{n+1\}$ and $V_{n+1} = V \backslash \{0\}$. There is a time horizon $[0, T]$ (typically a single day) in which vehicles move along the network. For each vertex $i \in V$, we denote by p_i to its processing time and $W_i = [r_i, d_i]$ the corresponding (hard) time window, where r_i and d_i are the release and deadline times, respectively. In particular, we set $W_0 = W_{n+1} = [0, T]$. We allow waiting times when arriving at a vertex before its release time r_i, but the vehicle must wait until r_i before starting to process it. In addition, each arc $(i, j) \in A$ has an associated travel distance L_{ij}. Without loss of generality, $d_i + p_i \leq T$ for all $i \in V$. In addition, to simplify the notation in the manuscript, we slightly modify the standard definition and assume that $p_i = 0$ for $i \in V$. However, the models and formulae present in this paper can be easily adapted to consider processing times.

The time dependency is modeled as follows. The planning horizon is partitioned into M intervals $[T_h, T_{h+1}]$, $h = 0, \ldots, M-1$. We assume that, for each arc $(i, j) \in A$, the average value of the travel speed during the time interval $[T_h, T_{h+1}]$, denoted by v_{ijh}, for $h = 0, \ldots, M-1$, is known. This partition with its corresponding travel speeds are referred as *speed profiles*. It is important to remark that the speed profiles may differ among arcs. Based on this definition, the main idea behind the speed model is to compute the travel times using the information of the distance to be traveled, i.e. L_{ij}, combined with the travel speeds v_{ijh} defined for the arc. However, it is not assumed that the travel speed remains fixed during the trip and it may change whenever the boundaries of an interval are crossed. We denote by $\tau_{ij}(t)$ to the time-dependent travel time value on arc $(i, j) \in A$ if departing from i at time $t \in [0, T]$, and it

can be computed following Algorithm 1 as proposed in Ichoua et al. (2003).

Algorithm 1 Computing the travel time of arc (i, j) at time t_0 (Ichoua et al., 2003).

```
t ← t_0
k ← k_0 : T_{k_0} ≤ t_0 ≤ T_{k_0+1}
d ← L_ij
t' ← t + (d/v_ijk)
while t' > T_{k+1} do
    d ← d − v_ijk × (T_{k+1} − t)
    t ← T_{k+1}
    t' ← t + (d/v_ijk+1)
    k ← k + 1
end while
return t' − t_0
```

The TDTSP-TW involves finding a tour that visits each vertex exactly once with the objective of minimizing the makespan of the route. The route starts at vertex 0 and ends at vertex $n+1$, while processing each vertex within its defined time window and computing the travel times following the speed model proposed in Ichoua et al. (2003).

Cordeau et al. (2012) propose expressing the travel speeds v_{ijh} as

$$v_{ijh} = \delta_{ijh} b_h u_{ij}, \tag{1}$$

where u_{ij} represents the maximum speed for arc $(i, j) \in A$ during the planning horizon, $b_h \in [0, 1]$ is the best congestion factor during interval $[T_h, T_{h+1}]$ and $\delta_{ijh} \in [0, 1]$ represents the heaviest degradation of the congestion factor of $(i, j) \in A$ in interval $[T_h, T_{h+1}]$ with respect to the less congested arc in $[T_h, T_{h+1}]$. From a practical standpoint, this decomposition allows the authors to formulate alternative scenarios that can be used to compute lower bounds for the problem. For instance, consider the TDTSP studied in Cordeau et al. (2012). If the travel speeds are increased by setting δ_{ijh} and b_h to one, then the problem reduces to a classical TSP with constant travel times. Therefore, computing the makespan of this solution using these increased speeds represents a lower bound for the problem. Furthermore, the authors show that this solution is optimal also for the case having general values of b_h but fixed δ_{ijh}, and that its objective function can be used to compute a lower bound on the tour duration for subpaths that are part of a feasible solution. These results are extended to the TDTSP-TW in Arigliano et al. (2015). Finally, we also remark some results regarding the generality of the speed model present in Ghiani and Guerriero (2014), where the authors show that any continuous piecewise linear travel time function can be modeled by the travel speeds defined in (1).

3. ILP formulations

In this section we present two ILP formulations for the TDTSP-TW. We begin by showing the formulation proposed in Arigliano et al. (2015) and describing some of its characteristics. We then present our formulation, which is based on the one proposed by Sun et al. (2015).

3.1. Travel-speed relaxation based model

This section presents the formulation proposed in Arigliano et al. (2015), which we name LBF. We follow most of their notation, although the formulation is presented in a slightly different way. Let $\mathcal{P}_i$ be the set of simple paths $\mathfrak{p} = (i_0, \ldots, i_k)$ in G starting at the depot and ending at i, i.e. $i_0 = 0$ and $i_k = i$. Let $LB_{\mathfrak{p}}$ denote a lower bound on the duration of any feasible solution having path $\mathfrak{p} \in \mathcal{P}_i$, $i \in V$, and that $LB_{\mathfrak{p}}$ is indeed the makespan whenever $\mathfrak{p}$ is a (feasible) Hamiltonian path from 0 to $n+1$. Let x_{ij} be the classical variable taking value one iff arc $(i, j) \in A$ is part of the solution, and z a variable that captures the makespan of the optimal solution. The LBF is shown below.

$$\min \ z \tag{2}$$

$$\text{s.t. } z \geq LB_{\mathfrak{p}} \left(1 + \sum_{(a,b)\in\mathfrak{p}} (x_{ab} - 1)\right), i \in V_{n+1}, \ \mathfrak{p} \in \mathcal{P}_i \tag{3}$$

$$\sum_{i\in V\setminus\{j,n+1\}} x_{ij} = 1, \quad j \in V_{n+1} \tag{4}$$

$$\sum_{j\in V\setminus\{i,0\}} x_{ij} = 1, \quad i \in V_0 \tag{5}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ij} \geq 1, \quad \forall S \subseteq V_0, \ |S| \geq 2 \tag{6}$$

$$\sum_{(a,b)\in\mathfrak{p}} x_{ab} \leq |\mathfrak{p}| - 1, \quad \mathfrak{p} \text{ infeasible path} \tag{7}$$

$$x_{ij} \in \{0, 1\}, \quad (i, j) \in A \tag{8}$$

The objective function (2) minimizes variable z, which accounts for the makespan of the optimal solution. Constraints (3) adjust the value of z by setting the lower bound $LB_{\mathfrak{p}}$ and the variables x_{ij} defining the solution. Constraints (4) and (5) are the outdegree and indegree constraints, respectively. Subtour Elimination Constraints (SEC) are imposed by constraints (6). Solutions that violate the time windows constraint are forbidden by means of the well known *Infeasible Path Elimination Constraints* (IPEC), proposed by Ascheuer et al. (2001), in constraints (7). Finally, the integral domain of the variables are imposed by constraints (8).

A BC algorithm is developed in Arigliano et al. (2015) using this formulation as starting point. Indeed, constraints (3) are used as cuts, as well as the SEC. The IPEC constraints (7) are replaced by the well-known *tournament constraints*. A set of valid inequalities, polynomial in the number of time intervals H are considered as well. Since H is rather small, they are directly included as a part of the formulation. In addition, as mentioned before, a lower bound is initially computed by solving an auxiliary problem, which is then also used to tighten the bound $LB_{\mathfrak{p}}$. Due to space limitations we omit the details regarding the computation of this bound and refer the reader to Cordeau et al. (2012) and Arigliano et al. (2015). However, we remark that results reported indicate that the bound is rather tight in most of the instances, both in the case with and without time windows.

3.2. Travel-time breakpoints based model

An alternative formulation for the TDTSP-TW can be obtained from the model proposed by Sun et al. (2015) for the Profitable TDTSP with Time Windows and Pickup and Delivery. This problem generalizes the TDTSP-TW since the vertices are not required to be visited and also incorporates one-to-one precedences. Sun et al. (2015) report that for this particular problem the model does not produce good results when solved by a commercial solver.

One of the interesting features of this ILP formulation is that, for each edge, it redefines the partitions of the time horizon in order to obtain a linear travel time function within each of them. The limits defining this new partition are referred as *time breakpoints* and allow to easily embed the piecewise linear time function within an ILP formulation. Formally, let $T^{ij} = \{T_1^{ij}, \ldots, T_M^{ij}\}$ be the new partition of the time horizon into time intervals (also called time zones) for arc $(i, j) \in A$. We denote the consecutive time

breakpoints defining $T_m^{ij} \in T^{ij}$ as $T_m^{ij} = [w_m, w_{m+1}]$. At this point, we abuse notation and refer to each time zone T_m^{ij} as $m \in T^{ij}$, with $m = 1, \ldots, |T^{ij}|$. By definition, $\tau_{ij}(t)$ becomes a linear function within each time zone that represents the travel time for arc (i, j) starting in time interval m. We denote by θ_{ij}^m and η_{ij}^m to the coefficients of the linear function, such that

$$\tau_{ij}(t_i) = \theta_{ij}^m t_i + \eta_{ij}^m, \quad \forall\, t_i \in T_m^{ij}. \tag{9}$$

To formulate the model, Sun et al. (2015) define binary variables x_{ij}^m taking value 1 iff the vehicle traverses arc $(i, j) \in A$ starting from i within time zone $T_m^{ij} \in T^{ij}$. For each vertex $i \in V$, a continuous nonnegative variable t_i accounts for the time that i is visited in the tour. The value of t_i is decomposed to indicate time for a given arc in a given zone. This is achieved by introducing continuous variables t_{ij}^m defined in the following fashion

$$t_{ij}^m = \begin{cases} t_i & \text{if } x_{ij}^m = 1, \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

We remark that similar decision variables have been considered in the context of the TDTSP by Stecco et al. (2008).

By combining (9) and (10) and considering variables x_{ij}^m, t_{ij}^m and t_i, the travel time function for arc (i, j), $\tau_{ij}(t_i)$, can be defined in an aggregated fashion as

$$\tau_{ij}(t_i) = \sum_{T_m^{ij} \in T^{ij}} \theta_{ij}^m t_{ij}^m + \eta_{ij}^m x_{ij}^m. \tag{11}$$

The formulation proposed in Sun et al. (2015), which we name TTBF, is shown next.

$$\min \ t_{n+1} \tag{12}$$

$$\text{s.t.} \sum_{(0,j)\in A} \sum_{m \in T^{0j}} x_{0j}^m = 1 \tag{13}$$

$$\sum_{(i,j)\in A} \sum_{m \in T^{ij}} x_{ij}^m = 1, \quad j \in V_{n+1} \tag{14}$$

$$\sum_{i \in V\setminus\{k\}} \sum_{m \in T^{ij}} x_{ik}^m = \sum_{i \in V\setminus\{k\}} \sum_{m \in T^{ij}} x_{ki}^m, \ \ k \in V \tag{15}$$

$$t_j \geq \sum_{m \in T^{ij}} \left(1 + \theta_{ij}^m\right) t_{ij}^m + \eta_{ij}^m x_{ij}^m, \quad (i, j) \in A \tag{16}$$

$$t_i = \sum_{(i,j)\in A} \sum_{T_m^{ij} \in T^{ij}} t_{ij}^m, \quad i \in V_0 \tag{17}$$

$$x_{ij}^m \max\{w_m, r_i\} \leq t_{ij}^m \leq x_{ij}^m \min\{w_{m+1}, d_i\}, \quad (i, j) \in A, \ T_m^{ij} \in T^{ij}, T_m^{ij} = [w_m, w_{m+1}] \tag{18}$$

$$r_i \leq t_i \leq d_i, \quad i \in V \tag{19}$$

$$x_{ij}^m \in \{0, 1\}, \quad (i, j) \in A, \ T_m^{ij} \in T^{ij} \tag{20}$$

The objective function (12) minimizes the arrival time at vertex $n+1$, which is indeed the makespan. Constraints (13) forces the vehicle to leave the depot, and (14) stand for the indegree equations. Constraints (15) establish the flow conservation between arcs entering and leaving each vertex. Constraints (16) compute the travel time between two vertices depending on the departure time at the origin and provide a lower bound for the arrival time t_j at the node $j \in V$. Constraints (17) together with (18) establish the correct departure time at each vertex and arc. Note that, in particular, constraints (18) establish the correct relation between variables t_{ij}^m and x_{ij}^m. Finally, constraints (19) ensure that each vertex is visited within its time window and constraints (20) define the domain of the variables.

4. BC algorithm

Based on the TTBF formulation presented in the previous section, we develop a BC algorithm for the TDTSP-TW. In this section we describe the main ingredients of the algorithm, including the preprocessing step, the construction of an initial feasible solution and the cutting plane algorithm.

4.1. Preprocessing

This is one of the key components in many difficult optimization problems and, based on the related literature, in particular for routing problems with time windows. Following the experience reported by Ascheuer et al. (2001), the preprocessing we apply consists of three phases, which are applied in an iterative fashion until no further changes are found. These phases are: arc removal and variable fixing, tightening of the time windows, and the derivation of precedences using the information provided by the time windows.

The elimination of arcs has a major impact on the formulation and allows to reduce the number of variables in the model. We apply a simple reduction using the information of the time windows and the travel time at the release time of the origin vertex, based on the fact that the FIFO condition is satisfied when traversing an arc. Formally, an arc $(i, j) \in A$ is infeasible if

$$r_i + \tau_{ij}(r_i) > d_j \tag{21}$$

and therefore it can be removed from A. This idea can be further generalized to consider time-dependent information and remove additional edges. Let $(i, j) \in A$ and $m \in T^{ij}$ such that $T_m^{ij} = [w_m, w_{m+1}]$. Firstly, if $T_m^{ij} \cap W_i = \emptyset$, then we can set $x_{ij}^m = 0$. Then, if

$$\max\{w_m, r_i\} + \tau_{ij}(\max\{w_m, r_i\}) > d_j \tag{22}$$

traveling form i to j starting in travel time period m is not feasible and therefore this time zone can be removed (or, equivalently, the corresponding variable fixed to 0).

The tightening of the time windows consists in the adaptation of three rules from the classical TSPTW considered in Ascheuer et al. (2001), which allow to adjust the release and deadline times of each vertex. The criteria are:

- If the earliest arrival time at a vertex $k \in V$ is later than the current release time, then the latter can be increased.

$$r_k = \max\left\{r_k, \min_{(i,k)\in A}\{r_i + \tau_{ik}(r_i)\}\right\} \tag{23}$$

- Unnecessary waiting times at the successors of a vertex $k \in V$ can be avoided by adjusting the release times. Consider $k \in V$, and an arc $(k, j) \in A$. We define the set $\alpha_k^j = \{t \leq d_k : t + \tau_{kj}(t) = r_j\} \cup \{d_k\}$.

$$r_k = \max\left\{r_k, \min_{(k,j)\in A} \min\{\alpha_k^j\}\right\} \tag{24}$$

- The due date can also be adjusted in order to remove instants which are not feasible to reach by at least one of its possible successors. Similarly to the previous case, we define $\beta_k^j = \{t \geq r_k : t + \tau_{kj}(t) = d_j\}$. Assuming that at least one of these sets is non-empty,

$$d_k = \min\left\{d_k, \max_{(k,j)\in A, \beta_k^j \neq \emptyset} \min\{\beta_k^j\}\right\}. \tag{25}$$

- If the latest arrival time to $k \in V$ from any possible predecessor is earlier than the actual deadline, then d_k can be decreased as

$$d_k = \min\left\{d_k, \min_{(i,k)\in A}\{d_i + \tau_{ik}(d_i)\}\right\}. \tag{26}$$

Finally, precedences are inferred using the information provided by the time windows. We say that a vertex i precedes vertex j, noted as $i \prec j$, if i must appear before vertex j in any feasible solution of the TDTSP-TW. We remark that in the TDTSP-TW the triangle inequality on the travel times does not necessarily hold. This assumption is usually made in other vehicle routing variants. Therefore, the travel time between two vertices cannot be used directly to define a precedence since it may be possible to find a faster route by traveling through another vertex. We construct an alternative graph $P = (V, R)$ such that arc $(i, j) \in R$ iff $i \prec j$. Precedences are defined in P by applying the following rules

- Let $i, j \in V$ such that $d_i < r_j$, then $i \prec j \in R$.
- Transitive closure: let $i, j, k \in V$ such that $i \prec j$, $j \prec k \in R$, then $i \prec k \in R$.
- $0 \prec j \in R$, $j = 1, \ldots, n$.
- $j \prec n+1 \in R$, $j = 1, \ldots, n$.

Clearly, if $i \prec j \in R$, then the arc (j, i) can be removed from A. We note that this rule is implicitly captured by the arc removal rule. However, the definition of the precedences are exploited, as explained later, by a well-known family of valid inequalities.

4.2. Initial heuristic

Finding a feasible solution for the TDTSP-TW is an $\mathcal{NP}$-Complete problem. Therefore, we adopt a similar strategy as in Ascheuer et al. (2001) and generalize several constructive heuristics as well some local search operators to the time-dependent case in order to find an initial feasible solution. Due to space limitations, we skip the implementation details and refer the reader to Ascheuer et al. (2001) for the general definition of each of them.

We sequentially apply the following constructive heuristics:

- *Sorting heuristics:* A tour is constructed by simply ordering the vertices according to a particular criterion, and then feasibility is checked. Such criteria are: vertex index, release date, due date, mid points of the time windows.
- *Nearest Feasible Neighbor:* A partial path is iteratively extended by including at the end a feasible vertex which results in the smallest increase in the (partial) makespan of the solution, if any exists.
- *Insertion heuristics:* A partial path is enlarged by inserting a vertex not considered so far between two consecutive vertices. Two criteria are considered to decide where and which vertex to insert, resulting in two different insertion heuristics. The first criterion regards the feasible insertion that results in the smallest increase of the makespan. The other criterion aims to the feasibility of the solution, and selects the vertex that has the smallest number of feasible insertion points.

The best solution, if any, is then selected and the following list of operators is applied until no further improvements are found: or-exchange, arc-reversal, swap, arc-reinsertion, node-reinsertion and two-node exchange. The resulting solution is then transferred to the BC algorithm, aiming to have a good upper bound on the makespan of the optimal solution and therefore being able to prune the enumeration tree efficiently.

4.3. Cutting planes

We now describe the cutting plane algorithm developed aiming to improve the quality of the lower bound provided by the LP relaxation. In this sense, as opposed to the approach in Arigliano et al. (2015), none of the following cuts are necessary for the formulation. In addition, as mentioned in the previous section, the approach in Arigliano et al. (2015) considers the computation of an initial lower bound by solving an auxiliary TSPTW subproblem[1], which is also used later as part of a set of restrictions that are included as lazy constraints. As we observed in preliminary experiments, the effectiveness of their approach appears to be tightly coupled to the quality of this initial lower bound.

A different strategy has been adopted and we avoid computing and using in our approach the initial lower bound proposed in Arigliano et al. (2015). Therefore, we resort to traditional BC techniques and evaluate a more aggressive cutting plane algorithm in combination with the TTBF. For this purpose, we included four exponential families of valid inequalities that have been proved to be quite effective when applied to TSP and TSPTW contexts.

We remark that valid inequalities that are feasible for the TSPTW can be easily incorporated and considered for the TTBF. Variables x_{ij} can be defined in terms of the variables x_{ij}^m by the following identity

$$x_{ij} = \sum_{m \in T^{ij}} x_{ij}^m.$$

4.3.1. Subtour elimination constraints

The formulation TTBF does not allow subtours as feasible solutions. Constraints (16) eliminate subtours on integer feasible solutions. However, SECs (6) usually produce good improvements in the value of the LP relaxation by cutting fractional solutions. Therefore, we decided to include the SECs as cutting planes to the TTBF as well. The SECs are separated using the routine proposed in Nagamochi et al. (1994).

4.3.2. Precedence-constrained TSP inequalities

The precedences computed during the preprocessing phase allow us to include valid inequalities from the Precedence-Constrained TSP proposed in Balas et al. (1995).

The first family of cuts we consider are the so-called π-inequalities. For $S \subseteq V \backslash \{0, n+1\}$, define the set of *predecessors* of S as

$$\pi(S) = \{i \in V \backslash \{0, n+1\} \ : \ i \prec j \text{ for some } j \in S\}.$$

The intuition behind the inequality is that, in a feasible solution, any vertex $i \in S \cap \pi(S)$ cannot be the last vertex visited in S. For $S, S' \subseteq V \backslash \{0, n+1\}$, let $\bar{S} = V \backslash S$ and

$$\delta(S, S') = \{(i, j) \in A \ : \ i \in S, j \in S'\}.$$

Then, the π-inequality defined by

$$\sum_{(i,j) \in \delta(S \backslash \pi(S), \bar{S} \backslash \pi(S))} x_{ij} \geq 1 \tag{27}$$

is valid for the TTBF. Similarly, the σ-inequalities exploit information about successors. Let $S \subseteq V \backslash \{0, n+1\}$ and

$$\sigma(S) = \{j \in V \backslash \{0, n+1\} \ : \ i \prec j \text{ for some } i \in S\}.$$

A vertex $j \in S \cap \sigma(S)$ cannot be the first vertex of S visited in the tour. Then, the σ-inequality

$$\sum_{(i,j) \in \delta(\bar{S} \backslash \sigma(S), S \backslash \sigma(S))} x_{ij} \geq 1 \tag{28}$$

is valid for the TTBF.

Finally, we further consider the (π, σ)-inequalities. Let $X, Y \subset V \backslash \{0\}$ be such that $i \prec j$ for every pair $i \in X, j \in Y$, and let $Q := \{0, n+1\} \cup \pi(X) \cup \sigma(Y)$. Then for any $S \subset V$ such that $X \subseteq S$, $Y \subseteq \bar{S}$, the (π, σ)-inequality

$$\sum_{(i,j) \in \delta(S \backslash Q, \bar{S} \backslash Q)} x_{ij} \geq 1 \tag{29}$$

is valid for the TTBF.

[1] We remark that such TSPTW minimizes the makespan and not the travel time or the travel costs.

The BC algorithm includes these inequalities as cutting planes in a similar fashion as in Ascheuer et al. (2001) and Dash et al. (2012). An exact separation procedure is considered for the separation of the *weak* version of the π and σ inequalities, which are obtained from the original inequalities by replacing $\pi(S)$ and $\sigma(S)$ by $\pi(j)$ and $\sigma(j)$, respectively, for $j \in S$. Whenever a violated weak inequality is found, then the corresponding π and σ inequalities (27) and (28) are included in the formulation. Note that the weak π and σ inequalities are dominated by the original version.

Regarding the (π, σ)-inequalities, we consider the *simple* version which accounts for sets $X = \{i\}$ and $Y = \{j\}$ such that $i \prec j$. For the separation routine, we also follow the approach proposed in Balas et al. (1995).

Within the BC, the cutting plane algorithm is executed until no violated cuts are identified.

5. Computational results

We conducted computational experiments in order to evaluate the behavior of the approach proposed in this paper and compare the results with another exact algorithm from the related literature. The algorithms are coded in C++, using `g++ 4.8.4` and an Ubuntu Linux 14.04 LTS as operating system, using CPLEX 12.4 Callable Library as LP and MILP solver. The experiments are run on a Workstation with an Intel Core i7-2600 3.4GHz CPU and 16Gb of RAM.

The algorithms are evaluated on two sets of instances. We first consider the instances reported in Arigliano et al. (2015), which are constructed by extending the instances generated in Cordeau et al. (2012) to the case with time windows. The original instances are randomly generated considering different settings for three concentric zones, aiming to emulate different customers distributions within the center of the city. In addition, alternative scenarios are defined by considering different maximum speeds, congestion factors and traffic patterns for the different zones. Following the notation introduced in (1), the congestion of the instances is partially captured by the values δ_{ijh} and $\Delta = \min_{i,j,h} \delta_{ijh}$. The latter characterizes the interval where the degradation of the congestion factor belongs.

Based on this information, Arigliano et al. (2015) extend the instances by incorporating time windows for the customers in such a way that at least one feasible solution exists. They consider values of $n = 15, 20, 30, 40$, $\Delta = 0.7, 0.8, 0.9, 0.98$ and two different traffic patterns, with 30 instances for each combination of these parameters. This gives a total of 960 instances.[2] We refer the reader to Arigliano et al. (2015) and Cordeau et al. (2012) for the detailed information regarding the construction and characteristics of the instances.

The second set of instances is also proposed by Arigliano et al. (2015)[3] but aim to evaluate different characteristics such as the size of the time windows and present a significantly larger number of time periods. We consider for our experiments a subset of the so-called *w100* instances corresponding to the combinations of $n = 15, 20, 30$ and $\Delta = 0.7, 0.8, 0.9, 0.98$ for the above mentioned patterns. The first 9 instances for each combination are considered, giving a total of 216 instances. We use this to evaluate the impact of the number of travel time periods on a BC algorithm based on TTBF, since it depends on this parameter. The selection of *w100* relies on the fact that, on average, the size of the time windows is similar to the original set of instances.

We begin by analyzing some particular characteristics as well as the impact of the preprocessing phase in both set of instances. The results are aggregated by the number of vertices, n, and reported as averages:

- # tot-edges: total number of time-dependent edges.
- # feas-edges: number of feasible time-dependent edges, that is, feasible combinations among $(i, j) \in A$ and $m \in T_m^{ij}$ after the preprocessing step.
- avg-tt-bp: average number of feasible travel time breakpoints per edge (after preprocessing).
- min-tt-bp: minimum number of feasible travel speeds breakpoints per edge (after preprocessing).
- max-tt-bp: maximum number of feasible travel speeds breakpoints per edge (after preprocessing).

The summary results for the preprocessing phase is shown in Table 1. For each of the metrics described below, we report the average (avg.), minimum (min.) and maximum (max.) over all instances for the corresponding value of n. We first note that the number of feasible edges after applying the preprocessing is significantly reduced with respect to the total number of edges. After this phase, only 10% of the edges remain feasible in the first set of instances and 5% in the second one, *w100*. This difference may be related with the number of avg-tt-bp obtained as a result of the definition of the travel speed profiles, which is significantly larger for the *w100* instances. This is one of the key ingredients of our approach, which showed to be very effective on this set of instances. Secondly, we remark the difference regarding the number of travel time breakpoints between the two sets of instances. The set *w100* presents a higher level of granularity in the definition of the speed profiles and, therefore, this impacts in the number of travel time breakpoints defined for each edge.

Regarding the methods, we consider the following exact algorithms:

- LBF-BC: BC algorithm proposed in Arigliano et al. (2015).
- TTBF-CPLEX: Formulation TTBF using CPLEX' default algorithm. We include SEC for $|S| = 2$ as part of the formulation.
- TTBF-BC: BC algorithm described in Section 4. CPLEX' dynamic search and primal reductions are automatically disabled.
- TTBF-CB: BC algorithm but the cutting plane phase described in Section 4 is run only at the root node. When finished, a new ILP is built including the inequalities added during the cutting phase and solved by CPLEX' default algorithm.

TTBF-CPLEX acts as a baseline to evaluate the other approaches. TTBF-BC and TTBF-CB further investigate the impact of the cutting plane algorithm when combined in two different settings with CPLEX. TTBF-CB is considered to make evident some particular behaviors.

For each combination of parameters, we report the following information:

- OPT: Number of instances solved to optimality within the set considered.[4]
- Time: computational times in seconds.
- Nodes: number of nodes explored in the enumeration tree.
- %rG: % gap at the root node.
- %fG: % gap at the end of the execution.

Except for the number of instances solved, the remaining metrics are reported as averages over the instances in the group.

We impose a limit of 3600 seconds for the execution time running in a single-thread. The computing time (Time) is averaged only over the solved instances. Similarly, %fG is averaged only considering the instances that are not solved to optimality. Gaps %rG and %fG are computed as $(z_{\text{best}} - z)/z$, where z_{best} represents the

[2] We note that the information regarding four particular instances is missing in the package, and therefore in practice we only discarded these instances.

[3] Although no results reported by the time of writing.

[4] We are considering the missing instances as *unsolved*.

Table 1
Preprocessing statistics for both sets of instances.

Set	n	Value	# tot-edges	# feas-edges	avg-tt-bp	min-tt-bp	max-tt-bp
Original	15	avg.	1785	264	1.67	1.00	4.94
		min.	1785	177	1.13	1.00	3.00
		max.	1785	404	2.35	1.00	5.00
	20	avg.	3080	437	1.62	1.00	5.00
		min.	3080	295	1.19	1.00	5.00
		max.	3080	627	2.19	1.00	5.00
	30	avg.	6720	850	1.50	1.00	5.00
		min.	6720	629	1.11	1.00	5.00
		max.	6720	1168	1.95	1.00	5.00
	40	avg.	11,760	1425	1.45	1.00	5.00
		min.	11,760	1042	1.11	1.00	5.00
		max.	11,760	1871	1.86	1.00	5.00
$w100$	15	avg.	36,962	1906	13.01	1.33	88.32
		min.	36,592	1586	10.37	1.00	48.00
		max.	37,199	2493	17.81	3.00	139.00
	20	avg.	63,918	3489	14.33	1.38	112.11
		min.	63,552	3078	12.61	1.00	81.00
		max.	64,201	3876	16.28	3.00	140.00
	30	avg.	139,621	7219	13.80	1.01	137.67
		min.	138,715	6077	12.01	1.00	119.00
		max.	140,229	7666	14.62	2.00	142.00

Table 2
Results for Arigliano et al. instances for Traffic Pattern A, $n = 15, 20, 30, 40$.

Δ	$\|V\|$	LBF-BC					TTBF-CPLEX					TTBF-BC					TTBF-CB				
		OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time
0.98	15	30	0.24	0.00	570	3.80	30	135.96	0.00	35,510	33.83	30	42.92	0.00	546	1.93	30	42.92	0.00	1730	1.98
	20	27	0.03	0.00	1030	11.41	30	173.89	0.00	893	3.92	29	24.45	0.00	213	1.50	30	25.85	0.00	1206	7.19
			0.11	0.11	–	–						1	66.39	0.13	–	–					
	30	25	0.01	0	0	0.50	29	343.21	0.00	6459	30.63	29	45.78	0.00	695	13.62	29	45.78	0.00	1770	10.31
	40	18	0.06	0.00	3	1.76	21	399.74	0.00	3809	36.23	27	62.07	0.00	11,556	383.35	28	61.37	0.00	10,231	119.26
			2.51	2.35	–	–	9	579.48	210.80	–	–	3	162.79	4.94	–	–	2	223.06	0.17	–	–
0.90	15	30	0.41	0.00	2767	16.00	30	66.82	0.00	234	0.23	30	10.52	0.00	28	0.13	30	10.52	0.00	100	0.17
	20	25	0.32	0.00	5927	59.89	30	211.52	0.00	10,565	8.49	30	44.10	0.00	311	1.95	30	44.10	0.00	696	1.45
			0.80	0.71	–	–															
	30	25	0.24	0.00	879	27.87	28	263.65	0.00	883	4.04	30	37.50	0.00	2327	98.27	30	37.50	0.00	6600	85.61
			0.89	0.84		–	2	447.04	66.17	–	–										
	40	10	0.22	0.00	300	40.73	23	367.05	0.00	7297	47.28	27	58.80	0.00	5014	201.46	26	53.40	0.00	5081	58.61
			2.14	1.88	–	–	7	510.79	169.41	–	–	3	196.73	4.08	–	–	4	197.37	4.63	–	–
0.80	15	30	0.76	0.00	813	5.76	30	84.09	0.00	8682	10.01	30	35.74	0.00	9309	89.95	30	35.74	0.00	14,463	16.10
	20	22	0.72	0.00	4021	53.50	30	169.98	0.00	3722	2.54	30	29.28	0.00	362.63	1.74	30	29.28	0.00	639	1.11
			0.99	0.61	–	–															
	30	14	0.10	0.00	1150	37.89	30	312.13	0.00	46,040	83.17	30	45.45	0.00	1297	35.05	30	45.45	0.00	2817	13.64
			2.22	1.17	–	–															
	40	11	0.53	0.00	615	47.66	28	335.59	0.00	23,538	123.96	30	56.52	0.00	1905	94.68	30	56.52	0.00	2269	25.38
			2.10	1.25	–	–	2	571.36	54.80	–	–										
0.70	15	29	1.31	0.00	1387	9.70	30	110.59	0.00	736	0.68	30	32.03	0.00	51	0.26	30	32.03	0.00	152	0.35
			1.93	1.90	–	–															
	20	21	1.00	0.00	116	1.65	29	127.87	0.00	5332	29.98	30	47.09	0.00	20,393	283.32	30	47.09	0.00	12,580	61.95
			1.98	1.14	–	–	1	376.92	1.20	–	–										
	30	16	0.63	0.00	1337	9.21	27	217.50	0.00	21,517	68.21	29	42.28	0.00	5967	134.55	29	42.28	0.00	15,386	63.87
			3.01	1.97	–	–	3	488.82	50.40	–	–	1	176.25	4.20	–	–	1	176.25	2.64	–	–
	40	12	1.00	0.00	6	2.40	26	219.10	0.00	16,686	134.25	25	47.38	0.00	467	141.66	28	44.85	0.00	5075	44.83
			3.11	1.73	–	–	3	580.01	152.75	–	–	4	63.01	0.38	–	–	1	180.78	0.74	–	–

objective function of the best feasible solution for the instance (eventually, the optimal solution) and z the value being considered, i.e. the lower bound at the root node or the lower bound of the objective value available in the enumeration tree when reaching the time limit. For each combination of parameters, the results are disaggregated between solved and unsolved instances.

The results obtained on the first set of instances for methods LBF-BC, TTBF-CPLEX, TTBF-BC and TTBF-CB for traffic patterns *A* and *B* defined in Arigliano et al. (2015) are presented in Tables 2 and 3, respectively. In both cases, the main message of the tables is that TTBF-CB produces the best results in terms of number of instances solved, the average computing time and number of nodes explored during the enumeration. Another observation involves TTBF-CPLEX, which is able to solve more instances that LBF-BC for both traffic patterns. This is somehow an unexpected behavior given the complex developments included in LBF-BC. However, the behavior exhibited regarding the relation between %rG and %fG supports our initial observation with respect to the ILP model behind LBF-BC. The initial lower bound considered is very tight in general, but in terms of a BC algorithm the formulation finds difficulties to improve this bound and close the gap to prove optimality. Indeed, the authors report that the cutting plane algorithm is not able to improve this bound at the root node, and we observed that in many of the instances solved the optimality is proved before starting the enumeration. Based on limited extra experimentation, our conjecture is that constraints (3), despite that LB_p may be tight, is a weak inequality from an ILP perspective. However, fur-

Table 3
Results for Arigliano et al.. instances for Traffic Pattern B, $n = 15, 20, 30, 40$.

Δ	|V|	LBF-BC					TTBF-CPLEX					TTBF-BC					TTBF-CB				
		OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time
0.98	15	27	0.19	0.00	1056	58.09	29	95.62	0.00	493	0.50	29	27.94	0.00	52	0.20	29	27.94	0.00	68	0.18
			0.20	0.21	–	–															
	20	17	0.16	0.00	820	45.92	29	213.62	0.00	1772	3.21	29	39.42	0.00	191	1.32	29	39.42	0.00	460	1.12
			0.22	0.21	–	–															
	30	13	0.24	0.00	511	14.67	28	234.42	0.00	10,857	25.35	29	48.06	0.00	13,377	262.09	29	48.06	0.00	7307	28.37
			1.44	0.58	–	–	1	489.71	122.26	394,706	3605.55										
	40	9	0.13	0.00	0	0.78	25	461.20	0.00	47,470	245.29	29	84.04	0.00	3437	144.36	29	84.04	0.00	8619	51.23
			1.00	0.73	–	–	5	625.61	178.73	–	–	1	116.32	0.10	–	–	1	116.32	0.10	–	–
0.90	15	27	0.99	0.00	800	5.50	30	67.01	0.00	1975	1.66	30	25.34	0.00	75	0.26	30	25.34	0.00	227	0.31
			3.33	1.55																	
	20	20	0.95	0.00	3281	40.16	30	126.24	0.00	4872	8.57	30	28.89	0.00	1070	9.52	30	28.89	0.00	2974	5.22
			1.11	0.99	–	–															
	30	18	1.00	0.00	673	18.41	30	245.46	0.00	1721	11.04	30	27.48	0.00	61	1.87	30	27.48	0.00	469	2.61
			1.03	0.88	–	–															
	40	10	0.70	0.00	205	25.21	25	242.24	0.00	7196	123.41	27	52.56	0.00	1556	135.00	28	51.41	0.00	5707	62.13
			1.93	0.70	–	–	5	616.07	201.78	–	–	3	95.20	0.15	–	–	2	132.54	0.22	–	–
0.80	15	28	2.35	0.00	1072	7.85	30	81.55	0.00	3938	3.17	30	43.81	0.00	607	2.27	30	43.81	0.00	998	1.13
			4.60	3.71	–	–															
	20	13	1.96	0.00	3595	41.58	29	121.78	0.00	1740	2.81	29	26.59	0.00	133	0.98	29	26.59	0.00	346	0.85
			2.72	2.05	–	–															
	30	7	1.91	0.00	592	15.84	30	259.18	0.00	22,822	84.52	29	43.80	0.00	1643	19.41	30	46.35	0.00	9415	37.54
			3.64	2.25	–	–						1	120.13	0.81	–	–					
	40	11	1.48	0.00	26	8.03	28	321.02	0.00	13,829	120.60	29	52.32	0.00	582	58.87	29	57.79	0.00	2654	31.42
			3.40	1.72	–	–	2	648.64	359.83	–	–	1	220.20	5.97	–	–	1	61.80	0.68	–	–
0.70	15	29	4.64	0.00	4561	38.27	30	90.92	0.00	553	0.79	30	31.23	0.00	171	0.61	30	31.23	0.00	307	0.60
			5.90	5.48																	
	20	16	3.69	0.00	1633	27.35	28	157.51	0.00	5480	5.50	28	39.50	0.00	119	0.93	28	39.50	0.00	421	1.14
			4.14	3.24	–	–															
	30	6	3.70	0.00	0	0.44	29	256.41	0.00	10,644	61.09	30	58.28	0.00	2694	29.84	30	58.28	0.00	1478	8.68
			5.35	3.71	–	–	1	478.82	13.62	–	–										
	40	3	0.99	0.00	0	1.08	26	258.27	0.00	18,352	126.08	25	43.86	0.00	326	41.97	30	56.44	0.00	9376	94.45
			0.03	2.80	–	–	4	400.83	115.26	–	–	5	119.36	0.90	–	–					

ther experiments and investigations should be conducted to obtain stronger evidence.

The comparison between TTBF-CPLEX and TTBF-BC also presents some interesting patterns. Firstly, TTBF-BC is able to solve more instances, and this tendency is clearer as n increases. The average %rG and %fG are significantly better for TTBF-BC, sometimes obtaining differences of about one order of magnitude. In addition, we can observe large %rG for some combinations under both traffic patterns and for both algorithms, with larger values for TTBF-CPLEX. Recall that %rG is relative to the lower bound evaluated, and therefore values larger than 100% indicate that when leaving the root node, the value of the LP relaxation after the cutting phase is about 1/3 of the optimal value. However, both TTBF-CPLEX and TTBF-BC, despite these large values, are able to close most of the instances. Regarding the comparison between TTBF-BC and LBF-BC follows a similar tendency as with respect to TTBF-CPLEX, but with a larger difference in favor of TTBF-BC.

At this point, it is important to note that the inclusion of user cuts within CPLEX disables some specific procedures (reductions, *dynamic search*, etc). To assess the impact of the valid inequalities, we consider TTBF-CB that includes the cutting plane at the root node and then the resulting ILP is solved using CPLEX's default algorithm. In this fashion, we are able to capture the impact of improving the %rG produced by the cutting planes. The results show that TTBF-CB is the most effective approach on this, solving more instances to optimality in less computational time.

We now restrict the analysis to the instances with $n = 40$.[5] The results follow the same pattern as the ones described before, but the difference in terms of the number of instances solved to optimality is accentuated. Indeed, when restricted to $n = 40$, from a total of 240 instances TTBF-CB is able to solve to optimality 228, TTBF-BC 219 instances and TTBF-CPLEX 202, while LBF-BC can solve 84. Large average %rG can be observed for the unsolved instances, which are significant when observing %fG. This suggests that both algorithms over TTBF are able to reduce, combining cutting planes with node enumeration, the difference between the lower and upper bounds.

Tables 4 and 5[6] show the results for the second set of instances considered for traffic patterns A and B, respectively. These tables show the results for TTBF-CPLEX, TTBF-BC and TTBF-CB since there are no public reports on LBF-BC. This experiment aims to evaluate the impact of the number of travel time breakpoints in the formulation, which is captured by the definition of the instances considered. Firstly, all methods produce reasonable results although the number of travel time periods per edge is larger than in the first set of instances. We observed an increment in the number of variables and constraints in the TTBF, although based on limited experiments the formulation seems to be more sensitive to the size of the time windows.

The comparison regarding TTBF-CPLEX and TTBF-BC is slightly different from the previous case. TTBF-CPLEX is capable of solving to optimality more instances than TTBF-BC despite the significant improvements in %rG due to the inclusion of specific valid inequalities. However, TTBF-CB is able to solve almost all instances in this set in smaller computing times. Overall, TTBF-CPLEX solves 202 instances out of the 216, TTBF-BC 187 and TTBF-CB 214. These results show that there is a great potential for improvements when com-

[5] We believe the %rG of LBF-BC for $\Delta = 0.7$, Traffic Pattern B is indeed a typo and should be multiplied by 100.

[6] We include some details for $\Delta = 0.7$ and $n = 30$ for TTBF-BC. The value of the LP relaxation when leaving the root node is close to (z_{best}) and, when rouned to two decimals, the resulting value is zero. TTBF-BC exits the root node with a gap of 0.92% and is not able to find the optimal solution during the enumeration, reaching the time limit with a final gap of 0.22%.

Table 4
Results for set $w100$ for Traffic Pattern A, $n = 15, 20, 30$.

Δ	|V|	TTBF-CPLEX					TTBF-BC					TTBF-CB				
		OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time
0.98	15	9	103.03	0.00	6753	31.52	9	61.91	0.00	2336	18.12	9	61.91	0.00	1571	9.28
	20	9	118.23	0.00	3158	22.12	9	45.88	0.00	12,201	102.78	9	45.88	0.00	18,034	104.03
	30	7	177.01	0.00	40,940	483.12	3	37.02	0.00	16,311	273.60	8	55.29	0.00	5306	87.30
		2	459.93	3.82	–	–	6	73.97	0.11	–	–	1	112.48	6.55	–	–
0.9	15	9	107.48	0.00	20,794	75.19	9	53.82	0.00	2810	17.60	9	53.82	0.00	1630	9.27
	20	9	72.28	0.00	2579	15.40	8	34.46	0.00	6319	53.96	9	35.97	0.00	1914	14.39
							1	48.06	0.24	–	–					
	30	7	146.10	0.00	70,178	495.79	7	36.66	0.00	24,407	475.71	9	52.19	0.00	24,054	227.78
		2	160.95	0.35	–	–	2	106.55	0.08	–	–					
0.8	15	9	108.42	0.00	7268	25.66	9	48.76	0.00	4003	24.56	9	48.76	0.00	1464	7.61
	20	9	83.21	0.00	10,345	57.79	9	36.85	0.00	16,370	139.96	9	36.85	0.00	4886	33.38
	30	7	85.18	0.00	10,202	187.26	5	21.50	0.00	298	19.53	9	42.19	0.00	41,493	311.34
		2	140.37	4.61	–	–	4	68.05	0.88	–	–					
0.7	15	9	94.93	0.00	2433	9.36	9	43.26	0.00	5037	31.71	9	43.26	0.00	2854	20.84
	20	8	52.06	0.00	9446	54.21	8	13.41	0.00	32,113	273.97	9	28.98	0.00	4345	26.75
		1	191.94	2.28	–	–	1	153.53	0.14	–	–					
	30	8	56.15	0.00	19,060	169.49	7	23.22	0.00	21,797	460.29	9	37.37	0.00	40,394	349.54
		1	281.37	0.08	–	–	2	86.90	0.04	–	–					

Table 5
Results for set $w100$ for Traffic Pattern B, $n = 15, 20, 30$.

Δ	|V|	TTBF-CPLEX					TTBF-BC					TTBF-CB				
		OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time	OPT	%rG	%fG	Nodes	Time
0.98	15	9	102.01	0.00	6956	22.80	9	54.86	0.00	39,266	226.50	9	54.86	0.00	2474	10.98
	20	9	124.80	0.00	3360	35.03	9	40.37	0.00	24,601	197.37	9	40.37	0.00	2426	17.66
	30	8	219.78	0.00	85,279	767.11	5	26.24	0.00	41,713	805.62	9	59.48	0.00	51,077	282.95
		1	331.06	0.08	–	–	4	101.03	0.05	–	–					
0.9	15	9	106.17	0.00	3427	14.40	9	40.20	0.00	3114	19.53	9	40.20	0.00	1465	8.77
	20	9	86.06	0.00	4401	29.82	9	39.39	0.00	12,436	129.32	9	39.39	0.00	2483	16.57
	30	7	109.38	0.00	4902	73.66	6	40.29	0.00	18,956	346.44	9	42.98	0.00	3274	62.02
		2	138.99	0.06	–	–	3	48.37	0.20	–	–					
0.8	15	9	78.20	0.00	1228	4.92	9	37.22	0.00	2597	12.39	9	37.22	0.00	973	4.23
	20	8	55.04	0.00	1608	9.02	9	24.02	0.00	35,523	320.04	9	24.02	0.00	722	10.33
		1	212.64	1.91	–	–										
	30	7	91.14	0.00	18,160	283.68	6	39.45	0.00	64,414	929.72	8	40.20	0.00	29,023	242.36
		2	0.96	0.51	–	–	3	28.92	0.68	–	–	1	1.82	0.93	–	–
0.7	15	9	62.44	0.00	917	3.07	9	12.34	0.00	445	2.93	9	12.34	0.00	894	3.45
	20	9	44.77	0.00	76,853	414.20	7	0.43	0.00	20	1.42	9	24.71	0.00	49,460	295.91
							2	109.70	1.93	–	–					
	30	9	30.02	0.00	36,105	195.43	8	10.84	0.00	18,857	294.06	9	9.63	0.00	14,222	161.79
							1	0.00	0.00	–	–					

bining problem-specific with general purpose ILP techniques which should be exploited and investigated in more detail.

Finally, we would like to make a comment with respect to a particular behavior observed during the experimentation. The travel time calculation proposed in Algorithm 1 assumes continuous (rational) information for computing travel times, and the models allow arrivals and departure to occur at fractional time instants. Therefore, the computation of the arrival/departure at each vertex is subject to numerical errors. This may affect not only the values of the objective function but, given the presence of time windows, the feasibility of a solution. Indeed, we experienced feasibility issues in some particular cases with both formulations TTBF-BC and TTBF-CB, related mainly to CPLEX's feasibility tolerance parameter. Furthermore, we noted some isolated cases where the default value of the parameter *relative MIP gap tolerance* produced some of the algorithms to terminate before proving optimality. In these cases, the optimal solution can be found when readjusting this parameter. Therefore, special considerations must be taken into account regarding numerical problems when tackling this version of the TDTSP-TW.

6. Conclusions and future research

This article presents an exact algorithm for the TDTSP-TW, a generalization of the TSPTW where the travel time between two cities is not constant along the day. We propose an ILP formulation following the research in Sun et al. (2015). Based on this formulation, we develop a two tailored BC algorithms including preprocessing rules, initial heuristics and valid inequalities, which proved to be effective. Compared to the approach proposed in Arigliano et al. (2015), the proposed BC approaches TTBF-BC and TTBF-CB are able to solve 929 and 940 instances, respectively, out of a total of 960, which represents a difference of more than 300 instances solved. In addition, computing times and the number of nodes explored are significantly reduced.

As future work, several research lines are worth investigating based on the results shown in this paper. Firstly, further research is needed regarding formulations and exact algorithms for time-dependent problems in general, and for the TDTSP-TW in particular. Alternative models which are able to effectively incorporate the time dependency could have a significant impact from an algorithmic perspective. In this same direction, further investigations regarding particular valid inequalities that account for the time dependency may produce improvements in the lower bounds pro-

vided by the LP relaxation which, combined with effective heuristic techniques, would improve the computation times and increase the size of the instances consistently solved. In addition, it would be very interesting to evaluate the behavior of extensions of the proposed approach in other time-dependent problems, such as the natural extension to the multiple vehicle case as well as in the TDTSP.

Regarding the experimental settings, it would be interesting to construct a larger set of benchmark instances considering also different construction patterns. Ideally, it would be interesting to include instances with real travel time information as well. This would provide a more diverse environmental context for the evaluation and comparison of the algorithm, and in particular regarding their applicability in practice.

Finally, we remark that VRPs usually assume data to be integer. Considering the numerical instability during the experimentation, it would be interesting to adapt the current travel speed model, and its corresponding travel time computation, to be able to work with discretized times while preserving its main characteristics. From a practical point of view, travel times could be represented as minutes, half minutes, seconds, etc., depending on the level of granularity required by the operations involved.

Acknowledgments

This research is partially supported by FONC$_y$T grant PICT-2013-2460 from the Government of Argentina, and by UBAC$_y$T grant 20020100100666 from Universidad de Buenos Aires, Argentina. The authors are grateful to the anonymous referees, the associate editors and the general editor for their careful reading and valuable comments, which helped improving a previous version of the article.

References

Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E., 2012. The time dependent traveling salesman problem: polyhedra and algorithm. Math. Programm. Comput. 5 (1), 27–55. doi:10.1007/s12532-012-0047-y.

Albiach, J., Sanchis, J.M., Soler, D., 2008. An asymmetric tsp with time windows and with time-dependent travel times and costs: an exact solution through a graph transformation. Eur. J. Oper. Res. 189 (3), 789–802. http://dx.doi.org/10.1016/j.ejor.2006.09.099.

Arigliano, A., Ghiani, G., Grieco, A., Guerriero, E., 2015. Time Dependent Traveling Salesman Problem with Time Windows: Properties and an Exact Algorithm. Technical Report.

Ascheuer, N., Fischetti, M., Grötschel, M., 2001. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Math. Program. 90 (3), 475–506.

Balas, E., Fischetti, M., Pulleyblank, W.R., 1995. The precedence-constrained asymmetric traveling salesman polytope. Math. Program. 68, 241–265. doi:10.1007/BF01585767.

Cordeau, J.-F., Ghiani, G., Guerriero, E., 2012. Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. Transp. Sci. 48 (1), 46–58.

Dabia, S., Ropke, S., van Woensel, T., De Kok, T., 2013. Branch and price for the time-dependent vehicle routing problem with time windows. Transp. Sci. 47 (3), 380–396.

Dash, S., Günlük, O., Lodi, A., Tramontani, A., 2012. A time bucket formulation for the traveling salesman problem with time windows. INFORMS J. Comput. 24 (1), 132–147. doi:10.1287/ijoc.1100.0432.

Fischetti, M., Laporte, G., Martello, S., 1993. The delivery man problem and cumulative matroids. Oper. Res. 41 (6), 1055–1064.

Furini, F., Kidd, M.P., Persiani, C.A., Toth, P., 2015. Improved rolling horizon approaches to the aircraft sequencing problem. J. Scheduling 18 (5), 435–447.

Gendreau, M., Ghiani, G., Guerriero, E., 2015. Time-dependent routing problems: a review. Comput. Oper. Res. 64, 189–197.

Ghiani, G., Guerriero, E., 2014. A note on the ichoua, gendreau, and potvin (2003) travel time model. Transp. Sci. 48 (3), 458–462. doi:10.1287/trsc.2013.0491.

Godinho, M.T., Gouveia, L., Pesneau, P., 2014. Natural and extended formulations for the time-Dependent traveling salesman problem. Discrete Appl. Math. 164, 138–153. doi:10.1016/j.dam.2011.11.019.

Gouveia, L., Voß, S., 1995. A classification of formulations for the (time-dependent) traveling salesman problem. Eur. J. Oper. Res. 2217 (93).

Hill, A.V., Benton, W., 1992. Modelling intra-city time-dependent travel speeds for vehicle scheduling problems. J. Oper. Res. Soc. 343–351.

Ichoua, S., Gendreau, M., Potvin, J.-Y., 2003. Vehicle dispatching with time-dependent travel times. Eur. J. Oper. Res. 144 (2), 379–396.

Lucena, A., 1990. Time-dependent traveling salesman problem—the deliveryman case. Networks 20 (6), 753–763.

Malandraki, C., Daskin, M.S., 1992. Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. Transp. Sci. 26 (3), 185–200.

Melgarejo, P.A., Laborie, P., Solnon, C., 2015. A time-dependent no-overlap constraint: application to urban delivery problems. In: Integration of AI and OR Techniques in Constraint Programming. Springer, pp. 1–17.

Méndez-Dıaz, I., Miranda-Bront, J., Toth, P., Zabala, P., 2011. Infeasible path formulations for the time-dependent tsp with time windows. In: 10 th Cologne-Twente Workshop on Graphs and Combinatorial Optimization CTW 2011, pp. 198–202.

Méndez-Díaz, I., Zabala, P., Lucena, A., 2008. A new formulation for the traveling deliveryman problem. Discrete Appl. Math. 156 (17), 3223–3237.

Miranda-Bront, J.J., 2012. Integer Programming approaches to the Time Dependent Travelling Salesman Problem. Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires.

Miranda-Bront, J.J., Méndez-Díaz, I., Zabala, P., 2013. Facets and valid inequalities for the time-dependent travelling salesman problem. Eur. J. Oper. Res. doi:10.1016/j.ejor.2013.05.022.

Nagamochi, H., Ono, T., Ibaraki, T., 1994. Implementing an efficient minimum capacity cut algorithm. Math. Program. 67 (1), 325–341. doi:10.1007/BF01582226.

Picard, J.-C., Queyranne, M., 1978. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Oper. Res. 26 (1), 86–110.

Stecco, G., Cordeau, J.-F., Moretti, E., 2008. A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times. Comput. Oper. Res. 35 (8), 2635–2655. http://dx.doi.org/10.1016/j.cor.2006.12.021.

Sun, P., Dabia, S., Veelenturf, L.P., Van Woensel, T., 2015. The Time-Dependent Pro_table Pickup and Delivery Traveling Salesman Problem with Time Windows. Technical Report. Eindhoven University of Technology.

P. Toth and D. Vigo, editors. Vehicle Routing: Problems, Methods, and Applications, Second Edition. MOS-SIAM Series on Optimization. 2014.