Albiach, J.; Sanchís Llopis, JM.; Soler Fernández, D. (2008). An asymmetric TSP with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation. European Journal of Operational Research. 189(3):789-802. https://doi.org/10.1016/j.ejor.2006.09.099

An asymmetric TSP with time windows and with time-dependent travel times and costs: an exact solution through a graph transformation

José Albiach[*], José María Sanchis[†], David Soler[‡]

Departamento de Matemática Aplicada and IMPA-UPV.

Universidad Politécnica de Valencia. Valencia, Spain

Abstract

In this paper we deal with an extended version of the Asymmetric Traveling Salesman Problem with Time Windows (ATSPTW) that considers time-dependent travel times and costs, for a more accurate approximation of some routing problems inside large cities, in which the time or cost of traversing certain streets (e.g. main avenues) depends on the moment of the day (for example rush-hours).

Unlike other existing papers about time-dependent routing problems, we focus on an exact method for solving this new problem. For this end we first transform the problem into an Asymmetric Generalized TSP and then into a Graphical Asymmetric TSP. In this way, we can apply a known exact algorithm for the Mixed General Routing Problem, which seems to run well with our resulting instances. Computational

[*]Corresponding author: José Albiach, Departamento de Matemática Aplicada and IMPA-UPV. Universidad Politécnica de Valencia. Camino de Vera s/n, 46022 Valencia, Spain. e-mail: jalbiach@mat.upv.es. Fax: +34-963877669.

[†]e-mail: jmsanchis@mat.upv.es

[‡]e-mail: dsoler@mat.upv.es

results are presented on a set of 270 adapted instances from benchmark ATSPTW instances.

Keywords: Traveling Salesman Problem, time window, time-dependence.

1 Introduction

The Asymmetric Traveling Salesman Problem with Time Windows (ATSPTW) is a well-known routing problem that can be defined as follows:

Given a directed graph $G = (V, A)$ with nonnegative costs associated with its arcs, each vertex i has an associated time window $[a_i, b_i]$, the vertex i_0 is the depot, and traversing arc $(i, j) \in A$ implies a travel time $t_{ij} > 0$, find a minimum cost circuit in G starting at i_0 at time a_{i_0} and passing through each vertex exactly once, such that the circuit leaves each vertex in its associated time window and ends at i_0 before b_{i_0}.

Note that it is allowed to arrive at vertex i before a_i (waiting time), but in this case the circuit will leave i at time a_i. For simplicity, if a service time is necessary at a vertex i, this time will be included in the travel times t_{ij}, $j \neq i$.

The ATSPTW has important applications, especially in sequencing and distribution problems. For this reason many papers have studied this topic in the last decade. See for example [1], [2], [5], [14], [19], [21], [27] and [29].

Like in most routing problems found in the OR literature, in the ATSPTW the arc costs or times are considered constant throughout the day. This assumption may result in a weak approximation to real-world conditions, since, for example, in distribution problems inside large cities, the time or cost of traversing some streets (e.g. main avenues) depends on the moment of the day (for example at rush-hours).

Routing problems with time-dependent costs have hardly been studied because they are more difficult to model and solve; however, more and more papers on vehicle routing problems are taking into account time-dependent travel costs for a more accurate approximation of the mathematical models

to the real problems. The research in [20] includes a detailed review of the literature on time-dependent routing problems, and as recent papers we may mention [18], [22] and [28]. These articles are based on heuristic procedures for the solution of multivehicle routing problems with different kinds of time-dependent travel times.

With respect to vehicle routing problems with a single vehicle including time-dependent travel times, we find papers [24] and [25], which study the Time-Dependent Traveling Salesman Problem (TDTSP), a problem similar to the one presented here, but with considerable differences which will be commented following the definition of our problem. These authors focused on heuristic procedures for the TDTSP without time windows, tested on instances with up to 55 vertices. In none of these instances the optimality of the solution was proved.

Some papers on routing with time-dependent travel times (e.g. [18] and [20]) discuss the "non-passing" (or "first-in-first-out") property of the travel times, which guarantees that if a vehicle leaves a vertex i for a vertex j at a given time, leaving vertex i for vertex j at a later time implies arriving later at vertex j.

In this paper we present the Asymmetric Traveling Salesman Problem with Time-Dependent Costs (ATSPTDC), a generalization of the ATSPTW in which, in addition to time windows, the cost and the travel time of each arc depend on the moment at which we start traversing it. Because of this, the circuit can start at the depot node i_0 at time $t_{i_0} > a_{i_0}$. For example, if time a_{i_0} belongs to a rush-hour, instead of starting the route at a_{i_0}, if possible, we can be working inside the warehouse for a short period of time until the traffic be moving quite freely.

The main difference of this paper with respect to the above mentioned articles is that we focus on an exact approach to the problem; we find a way of solving the new problem by transforming it into the classical Asymmetric Traveling Salesman Problem (ATSP), for which several exact procedures have

been tested, even for large-scale instances (see for example [6], [7], [15], and [16]). A comparison of the performance of ATSP exact solvers is given in [17].

The idea of transforming routing problems into an ATSP is not new. For example, it was proposed in [23] to both optimally and heuristically solve instances belonging to several classes of arc routing problems. It was also proposed in [3] and [9] to optimally solve arc routing problems with turn penalties and forbidden turns with the purpose of comparing the results obtained with specific heuristic procedures, and also in [4] to solve the General Routing Problem, for which, at that time, there was no exact specific procedure for the mixed case. According to the computational results presented in these four papers, ATSP transformation works best on asymmetric problems or mixed problems with few edges.

The main advantage of the ATSP transformation is that efficient algorithms can be applied directly, as a black box, without any modification, to solve problems for which this is the only available methodology (this is our case), or to measure the efficiency of proposed heuristics.

To transform the new problem into an ATSP, we first transform it into another combinatorial optimization problem studied in the OR literature, namely the Asymmetric Generalized Traveling Salesman Problem (AGTSP). The AGTSP is defined as follows:

Given a directed graph $G = (V, A)$ *with nonnegative costs associated with its arcs, such that* V *is partitioned into* k *nonempty subsets* $\{S_i\}_{i=1}^{k}$*, find a minimum cost circuit passing through exactly one vertex of each subset* S_i $\forall i \in \{1, \ldots, k\}$.

To solve the AGTSP several polynomial time transformations of this problem into an ATSP have been described in the literature. The most efficient seems to be the transformation defined in [26]. As we will use this transformation, let us describe it briefly:

From G *construct a new directed graph with the same vertex set but order the vertices of each* S_i *consecutively in an arbitrary way,* $\{v_1^i, \ldots, v_{l(i)}^i\}$*,* $l(i)$

being the number of vertices in S_i. *For* $j = 1, \ldots, l(i) - 1$ *define the cost* $c^i_{j,j+1}$ *of arc* (v^i_j, v^i_{j+1}) *as zero, define* $c^i_{l(i),1}$ *as zero and for every* $v^i_j \in S_i$ *and every* $w \notin S_i$ *set* $c_{v^i_j,w}$ *equal to the cost in* G *of the arc from* $v^i_{j+1(mod\ l(i))}$ *to* w *plus a fixed positive large quantity* M *if* $|S_i| > 1$ *and equal to the cost in* G *of the arc from* v^i_j *to* w *plus* M *if* $|S_i| = 1$, *any other arc has infinite cost.*

Solving the AGTSP in G is equivalent to solving the ATSP in the new digraph.

Finally, as our aim is to find an exact solution to the new problem, we will use the exact algorithm for the Mixed General Routing Problem (MGRP) described in [11] to solve the resulting ATSP instances. The MGRP basically consists of finding a minimal closed walk on the edges and arcs (links) of a mixed graph G (G has simultaneously edges and arcs) which traverses a given subset of "required" links and a given subset of "required" vertices. This problem contains a large number of important arc and node routing problems as particular cases. For example, if G is a directed graph, the subset of required arcs is empty and all the vertices are required, we have the Graphical Asymmetric Traveling Salesman Problem (GATSP) (see for example [8]). The GATSP is a generalization of the pure ATSP in which graph G does not need to be complete (few variables are needed) and then, the solution does not need to be a Hamiltonian cycle but a closed walk passing through each vertex at least once. Note that an ATSP instance can be transformed into a GATSP instance by simply adding a large positive number L to each arc cost in order to assure the occurrence of a Hamiltonian cycle in the optimal solution. Therefore, as the GATSP is a particular case of the MGRP, the exact algorithm in [11] can be used to optimally solve the ATSPTDC. It is a cutting-plane algorithm based on the polyhedral study of the MGRP presented in [10] and [11].

Figure 1 shows the overall procedure followed to optimally solve ATSPTDC instances.

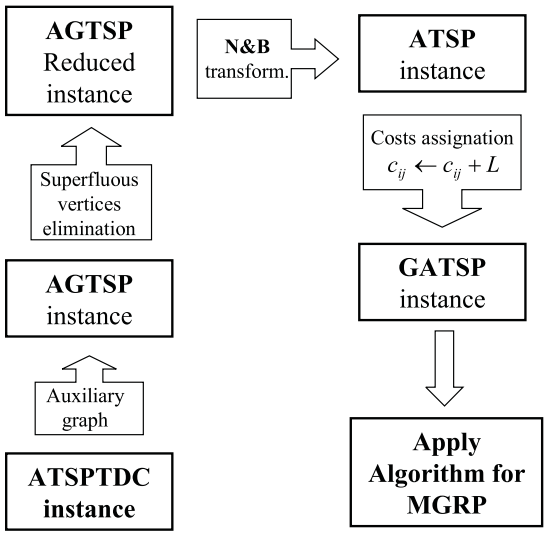

Fig. 1. Overall procedure to optimally solve the ATSPTDC.

The rest of the paper is organized as follows. In Section 2 we formally define the ATSPTDC and we show the construction of an auxiliary digraph from a ATSPTDC instance. In Section 3 we prove that the ATSPTDC can be transformed in pseudo-polynomial time into an AGTSP on the auxiliary digraph. The size of the auxiliary digraph is then considerably reduced in order to make this transformation more competitive. In Section 4 we present computational results on the exact resolution on a set of 270 ATSPTDC instances obtained by modifying benchmark ATSPTW instances (see for example [14], [21] and [29]), by using the exact MGRP algorithm in [11]. In this section we also justify the advantage of using this exact procedure over other existing codes for optimally solving the ATSP. Finally, Section 5 presents some conclusions.

2 Definition of the ATSPTDC and auxiliary digraph

We define the ATSPTDC in the following way:

Let $G = (V, A)$ *be a simple directed graph,* $V = \{v_i\}_{i=0}^{n}$ *being its set of vertices, where* v_0 *is the depot vertex. Each vertex* v_i *has an associated time window* $[a_i, b_i]$ *verifying that* $a_i, b_i \in \mathbb{Z}^+ \cup \{0\}$ *and* $[a_i, b_i] \subseteq [a_0, b_0]$ $\forall i \in$

$\{1, \ldots, n\}$. *Every time window* $[a_i, b_i]$ *has associated* $p_i = b_i - a_i + 1$ *instants of time* $\{a_i + k - 1\}_{k=1}^{p_i}$. *For simplicity we will denote* $t_i^k = a_i + k - 1$ *and therefore,* $t_i^k \in \mathbb{Z}^+ \cup \{0\}$.

On the other hand, the time and the cost of traversing an arc $(v_i, v_j) \in A$ *depend on the instant of time* t_i^k $(k \in \{1, \ldots, p_i\})$ *at which we start traversing it. Let us denote by* $t_{i,j}^k \in \mathbb{Z}^+$ *and* $c_{i,j}^k \geq 0$ *the time and the cost respectively of traversing arc* (v_i, v_j) *starting at instant* t_i^k.

The goal in the ATSPTDC is to find a Hamiltonian circuit in G, *starting and ending at* v_0 *at integer instants of time inside* $[a_0, b_0]$ *such that:*

- *Starting the circuit at time* $t_0^k \geq a_0$ *involves a waiting time cost* $cwt_0(t_0^k - a_0) \geq 0$ *with* $cwt_0(0) = 0$.

- *The circuit leaves each vertex* $v_i \in V$ *with* $i > 0$ *inside its associated time window.*

- *If the circuit arrives at vertex* v_i *with* $i > 0$ *at time* $t \in \mathbb{Z}^+$ *with* $t \leq a_i$, *it is allowed a waiting time* $a_i - t$ *with cost* $cwt_i(a_i - t) \geq 0$ *(with* $cwt_i(0) = 0$*). In this case the circuit must leave vertex* i *at time* a_i.

- *The sum of the costs of traversing arcs and of the waiting time costs be minimum.*

Some relevant aspects of this definition are:

- This definition allows the circuit to start after instant a_0 with a waiting time cost. We think that this is very important to minimize costs.

- Like in the ATSPTW, this definition also allows a waiting time $a_i - t$ if the circuit arrives at vertex v_i with $i > 0$ at time $t \leq a_i$. This waiting time has an associated cost (a waiting penalty) which is normally given by a non-decreasing linear function.

- As we stated for the ATSPTW and following some authors (see for example [5], [13] and [21]), if a service time is necessary at a vertex i with $i > 0$, this time is included in the travel times t_{ij}^k for all $j \neq i$ and for all k.

- From a practical point of view, the fact that the travel times must take integer values is not a strong restriction with respect to the continuous case,

because we can define an appropriate and as-small-as required time unit for each instance.

- In contrast to other papers, this definition distinguishes between two magnitudes: the time-dependent travel time and the time-dependent cost (which could be equal), focusing on cost minimization. In the particular case of the ATSPTDC in which $t_{ij}^k = t_{ij}^s = c_{ij}^k = c_{ij}^s \ \forall k, s \in \{1, ..., p_i\}$ and $\forall (v_i, v_j) \in A$ with $i \neq 0$, $c_{0,j}^k = \infty \ \forall k > 1$ and $\forall j > 0$ (the circuit must start at time a_0), and all waiting time costs equal to zero, we obtain an ATSPTW. Thus, the ATSPTDC is an NP-hard problem.

Some important differences with the problem studied in [24] and [25] are: in the TDTSP the circuit must start from the depot at exactly instant a_0, the travel costs coincide with the travel times, and as its objective function is the difference between the return time and the starting time of the circuit solution, it does not distinguish between travel costs and waiting time costs. In addition, the heuristics used work only on TDTSP without time windows. On the other hand, the main difference is that we focus on an optimal solution to the problem.

Note that given an ATSPTDC instance, we present a way to solve it independently of the characteristics of its travel time functions. Therefore, in this paper we do not discuss if the instance does or does not verify the non-passing property, which is an implicit characteristic of the instance data.

Consider then an ATSPTDC defined on a directed graph $G = (V, A)$ with all the corresponding data. We construct a directed auxiliary graph $G' = (V', A')$ in the following way:

- For each vertex v_i with $i \in \{0, ..., n\}$ and for each instant of time t_i^k for all $k \in \{1, ..., p_i\}$ create a vertex v_i^k.

- For each pair of vertices $v_i^k, v_j^l \in V'$ with $i \neq j$ and such that $t_j^l = max\{a_j, t_i^k + t_{ij}^k\}$ add to G' an arc (v_i^k, v_j^l) with a cost equal to:

  - $c_{i,j}^k$ if $l > 1$ and $i \neq 0$.
  - $c_{i,j}^k + cwt_0(t_0^k - a_0)$ if $l > 1$ and $i = 0$.

- $c^k_{i,j} + cwt_j(a_j - (t^k_i + t^k_{ij}))$ if $l = 1$ and $i \neq 0$.

- $c^k_{i,j} + cwt_j(a_j - (t^k_i + t^k_{ij})) + cwt_0(t^k_0 - a_0)$ if $l = 1$ and $i = 0$.

- Divide $\{1, ..., p_0\}$ into four subsets I_1, I_2, I_3, I_4 in the following way:

1) $k \in I_1$ if v^k_0 has only leaving arcs in G'. In this case, replace also name v^k_0 by v^k_{0s} (starting vertex).

2) $k \in I_2$ if v^k_0 has only entering arcs in G'. In this case, replace also name v^k_0 by v^k_{0e} (ending vertex).

3) $k \in I_3$ if v^k_0 has both entering and leaving arcs in G'. In this case, split v^k_0 into two vertices v^k_{0s} and v^k_{0e} such that v^k_{0s} will be only incident with the leaving arcs from v^k_0 in G' and v^k_{0e} will be only incident with the entering arcs to v^k_0 in G'.

4) $k \in I_4$ if v^k_0 has neither entering arcs nor leaving arcs in G'. Then, delete v^k_0 from G' for all $k \in I_4$.

- Add to G' a new vertex v_d, which will be the depot, with the following arcs, all of them with zero cost:

For each $k \in I_1 \cup I_3$, an arc (v_d, v^k_{0s}).

For each $k \in I_2 \cup I_3$, an arc (v^k_{0e}, v_d).

An ATSPTDC example with $n = 3$ illustrates the construction of this auxiliary digraph. In this example we consider that all waiting times have zero cost and that the travel cost is proportional to the travel time except for a little deviation due, for instance, to the characteristics of the different routes. More specifically, the travel cost is about 40 times the travel time with 5% maximum deviation, that is, $c^t_{ij} \in [0.95 \cdot 40 \cdot t^t_{ij}, 1.05 \cdot 40 \cdot t^t_{ij}] = [38t^t_{ij}, 42t^t_{ij}]$. The time windows are given in Figure 2.

Table 1 shows the time-dependent travel times and costs corresponding to this example, in a particular format, trying to simplify the construction of the auxiliary digraph. Each t^k_i shows in brackets its corresponding time instant. For example, the ordered pair corresponding to the row t^1_0 and to the column

t_1^2 means that if we traverse arc (v_0, v_1) starting at time t_0^1, which corresponds to instant 1, $t_{0,1}^1 = 2$ and $c_{0,1}^1 = 82$. A dash inside the cell corresponding to row t_i^k and column t_j^l means that if we traverse edge (v_i, v_j) starting at time t_i^k we will not arrive at v_j at time t_j^l if $l > 1$ or that we will arrive after a_j if $l = 1$. Note that the table does not include the rows and columns with not possible paths.

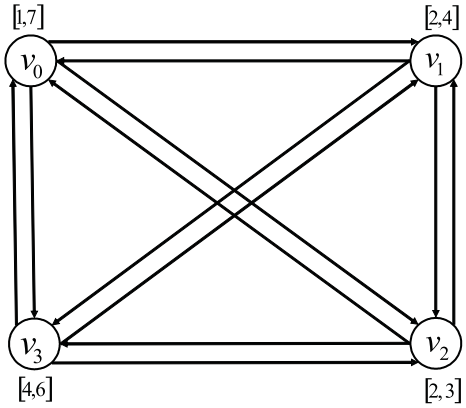

Fig. 2. Graph G.

Figure 3 shows the corresponding auxiliary digraph G' in which the vertices are denoted by numbers, indicating the order of the time instants in their corresponding time window; the vertices are clustered into subsets S_i corresponding to original vertices v_i. The arc costs have been omitted in this figure; they can be easily obtained from Table 1 and from the construction of G'.

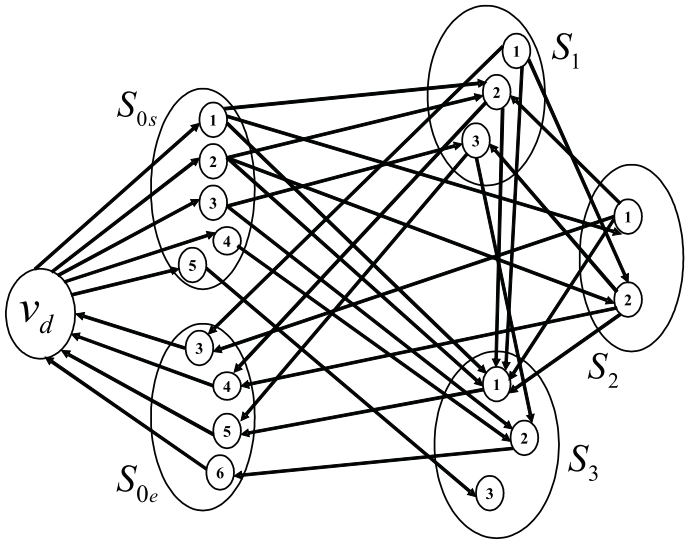

Fig. 3. Auxiliary digraph G'.

Table 1: Time-dependent travel times and costs (t_{ij}^k, c_{ij}^k) of graph G; $t_i^k(t)$ means that instant t_i^k is equal to t

	$t_1^2(3)$	$t_1^3(4)$	$t_2^1(2)$	$t_2^2(3)$	$t_3^1(4)$	$t_3^2(5)$	$t_3^3(6)$
$t_0^1(1)$	(2,82)	-	(1,42)	-	(3,115)	-	-
$t_0^2(2)$	(1,42)	-	-	(1,39)	(2,78)	-	-
$t_0^3(3)$	-	(1,42)	-	-	-	(2,77)	-
$t_0^4(4)$	-	-	-	-	-	(1,40)	-
$t_0^5(5)$	-	-	-	-	-	-	(1,41)

	$t_0^3(3)$	$t_0^4(4)$	$t_0^5(5)$	$t_2^2(3)$	$t_3^1(4)$	$t_3^2(5)$
$t_1^1(2)$	(1,41)	-	-	(1,40)	(2,79)	-
$t_1^2(3)$	-	(1,40)	-	-	(1,41)	-
$t_1^3(4)$	-	-	(1,42)	-	-	(1,39)

	$t_0^3(3)$	$t_0^4(4)$	$t_1^2(3)$	$t_1^3(4)$	$t_3^1(4)$
$t_2^1(2)$	(1,41)	-	(1,41)	-	(2,83)
$t_2^2(3)$	-	(1,40)	-	(1,38)	(1,42)

	$t_0^5(5)$	$t_0^6(6)$
$t_3^1(4)$	(1,42)	-
$t_3^2(5)$	-	(1,41)

3 Transformation of the ATSPTDC into an ATSP

Once the auxiliary digraph G' has been defined, we present a way to solve the ATSPTDC by first transforming it into an AGTSP and then transforming the resulting AGTSP into an ATSP using the transformation in [26] that does not increase the size of the graph.

Theorem 1 *The ATSPTDC can be transformed in pseudo-polynomial time into an AGTSP defined in the auxiliary digraph.*

Proof: Let $G = (V, A)$ be the digraph where an ATSPTDC is defined and let $G' = (V', A')$ be its auxiliary digraph. Consider an AGTSP in G' corresponding to the partition of V' into the following subsets: $S_d = \{v_d\}$, $S_i = \{v_i^k\}_{k=1}^{p_i}$ $\forall i \in \{1, \ldots, n\}$, $S_{0s} = \{v_{0s}^k\}_{k \in I_1 \cup I_3}$ and $S_{0e} = \{v_{0e}^k\}_{k \in I_2 \cup I_3}$, that is, $n + 3$ subsets.

By construction of G' there is a one-to-one correspondence between the set of feasible AGTSP solutions in G' and the set of feasible ATSPTDC solutions in G. It is enough to identify the circuit AGTSP solution in G' $T' = \{v_d, v_{0s}^{k_0}, v_{i_1}^{k_1}, v_{i_2}^{k_2}, \ldots, v_{i_n}^{k_n}, v_{0e}^{k_{n+1}}, v_d\}$ with the feasible ATSPTDC solution H in G consisting of the Hamiltonian circuit $\{v_0, v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_0\}$ starting at v_0 at time $k_0 \in [a_0, b_0]$, leaving vertex v_{i_r} at time $t_{i_r}^{k_r} = a_{i_r} + k_r - 1 \in [a_{i_r}, b_{i_r}]$ $\forall r \in \{1, \ldots, n\}$ and ending at v_0 at time $a_0 + k_{n+1} - 1 \in [a_0, b_0]$ (note that two feasible ATSPTDC solutions in G with the same Hamiltonian circuit but at least one different time instant t_i^k for leaving v_i are taken as different solutions). Both T' and H have the same cost; so an optimal AGTSP solution in G' gives rise in an easy way to an optimal ATSPTDC solution in G and because of the construction of G', no better solution for the ATSPTDC than the optimal one should exist.

Finally, $|V'|$ is upper bounded by a linear function on $\sum_{i=0}^{n}(b_i - a_i + 1)$, that is, $|V'|$ is $O(|V|p^*)$, p^* being the average of the values $b_i - a_i + 1$ with $i \in \{0, 1, \ldots, n\}$. Therefore, this transformation is pseudo-polynomial. ∎

In this way, from a theoretical point of view the ATSPTDC can be solved. Nevertheless, the size of the auxiliary digraph could be too large to apply known procedures for solving its associated ATSP in certain real distribution problems inside large cities: the servicing time windows of the customers could have a relatively small size (for example one or two hours) after preliminary studies and negotiations, but the depot time window should be opened during the entire working day. If there are customers to be serviced early and customers to be serviced at the end of the working day, each one of the sets S_{0s} and S_{0e} will contain about $b_0 - a_0$ vertices. For example, with this condition, for an 8-hour working day with a time unit equal to 1 minute (this is the smallest time unit normally considered in real vehicle routing problems inside large cities), the depot could generate about $8 \times 60 \times 2 = 960$ vertices in the auxiliary digraph.

Although nowadays there are exact procedures capable of solving large-scale ATSP instances with thousands of vertices, as we mentioned in the introduction, this transformation does not seem very attractive to solve the ATSPTDC because of the large size of the depot time window.

We show next that the size of the auxiliary digraph can be considerably reduced thus becoming more competitive. In fact, in the "reduced" auxiliary digraph, the number of vertices generated from the depot will always be 1, independently of the size of the depot time window. Thus, in the example given above, we would only have 1 vertex vs the about 960 vertices (a very considerable reduction), and in our example of Figure 2, we would have 1 vertex vs the 10 vertices in Figure 3.

Let then $G' = (V', A')$ be the auxiliary digraph obtained from the original ATSPTDC instance. From G' we construct a reduced auxiliary digraph $G" = (V", A")$ in the following way:

- Remove all vertices of G' corresponding to the subsets S_{0s} and S_{0e}.
- Maintain the rest of vertices of G' including v_d.
- For every vertex $v \in G"$ different from v_d do $cost(v_d, v) = \min_k \{cost(v_{0s}^k, v)\}$.

- For every arc (v, v_{0e}^k) with finite cost in G' do $cost(v, v_d) = cost(v, v_{0e}^k)$. Note that this cost is well defined because for a given v there is at most one arc going from v to the vertex set S_{0e}.

- Maintain the arc costs between vertices belonging to different sets S_i with $i \in \{1, ..., n\}$.

- Remove all vertices $v_i^k \in G$" that satisfy one of the following three conditions, assuming that an arc (u, v) exists in G" if a finite value had previously been assigned to $cost(u, v)$:

i) $d^+(v_i^k) = 0$ or $d^-(v_i^k) = 0$.

ii) $d^+(v_i^k) = d^-(v_i^k) = 1$ corresponding to arcs (v_d, v_i^k) and (v_i^k, v_d).

iii) $d^-(v_i^k) \geq 1$, $d^+(v_i^k) = 1$ corresponding to arc (v_i^k, v_d) and it exists at least one index j $\; j \neq i$ that satisfies $a_i + k - 1 \leq a_j$.

Figure 4 shows the reduced auxiliary digraph G" from G' in Figure 3 corresponding to our example. As we have said, the number of vertices generated from the depot is 1 vs 10 vertices in Figure 3, and vertices v_1^1 and v_3^3 have been removed, so G" has 7 vertices while G' has 18 vertices.

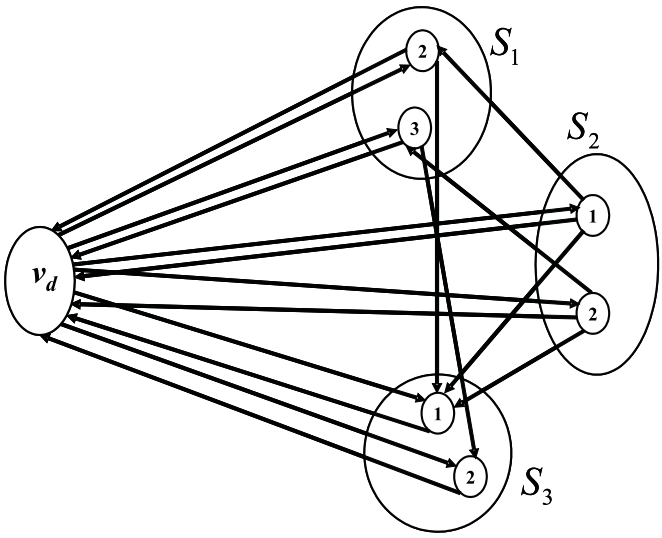

Fig. 4. Reduced auxiliary digraph G".

We define in G" an AGTSP in the same terms as the AGTSP defined in G' (see the proof of Theorem 1), except that the subsets S_{0s} and S_{0e} have been removed and that some S_i may contain fewer elements than in G' (the removed vertices).

Theorem 2 *Solving the AGTSP in G' is equivalent to solving the AGTSP in G".*

Proof: Given a feasible AGTSP solution in G', if we replace its initial sequence $\{v_d, v_{0s}^k, v\}$ by the sequence (arc) $\{v_d, v\}$ in G" and we replace its final sequence $\{u, v_{0e}^m, v_d\}$ by the sequence (arc) $\{u, v_d\}$ in G", it is evident that we have a feasible AGTSP solution in G". Moreover, the costs of $\{u, v_{0e}^m, v_d\}$ and $\{u, v_d\}$ are the same, and in an optimal solution in G', the cost of $\{v_d, v_{0s}^k, v\}$ must necessarily be $\min_k\{cost(v_{0s}^k, v)\}$, which is the cost of $\{v_d, v\}$ in G".

On the other hand, there is no feasible AGTSP solution in G' containing a vertex v_i^k that satisfies one of the conditions (i), (ii) and (iiii) given above: it is evident for condition (i); it is evident for condition (ii) except for the trivial case $n = 1$ that should not be considered as an AGTSP; and condition (iii) means that subset S_j will not be visited by the solution.

Thus, an optimal AGTSP solution in G' gives rise to a feasible AGTSP solution in G" with the same cost, and following the same reasoning, a feasible AGTSP solution in G" gives rise to a feasible AGTSP solution in G' with the same cost; therefore, an optimal AGTSP solution in G' results in an optimal AGTSP solution in G" and vice versa. ∎

Therefore, we can solve an ATSPTDC in graph G by solving an AGTSP in G". Following with our example, once we have the reduced auxiliary digraph G", using the transformation of G" given by Noon and Bean, we obtain the ATSP optimal solution shown in Figure 5, from which we can easily generate the AGTSP optimal solution in G" presented in Figure 6 and then, the ATSPTDC optimal solution in G illustrated in Figure 7 $(v_0, v_2, v_1, v_3, v_0)$ with time sequence $(2, 3, 4, 5, 6)$ and with cost $39 + 38 + 39 + 41 = 157$. Note that

this optimal circuit does not start at time a_0, which is equal to 1 in this case (there is a waiting time in the warehouse).

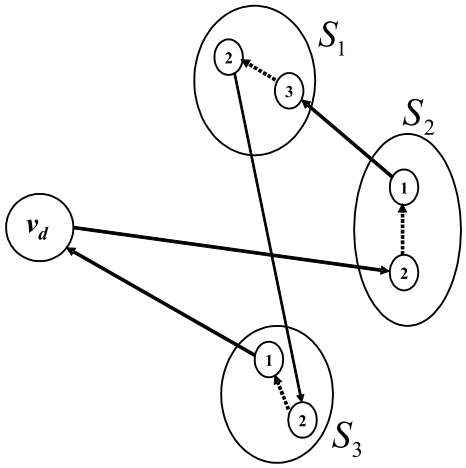

Fig. 5. ATSP optimal solution.

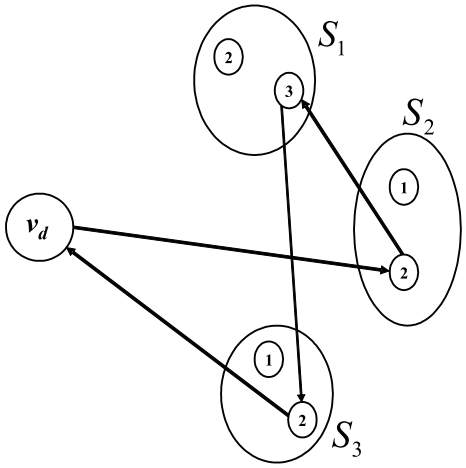

Fig. 6. AGTSP optimal solution in G".

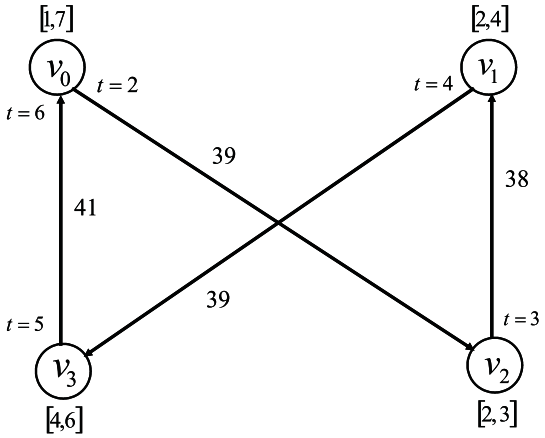

Fig. 7. ATSPTDC optimal solution in G.

4 Computational experiments

In order to verify the efficiency of this transformation, some computational experiments were performed on 270 instances obtained by modifying benchmark ATSPTW instances (see for example [14], [21] and [29]) as follows:

- The waiting time before leaving the depot and the waiting time when arriving at a vertex i before a_i always have zero cost.

- Since in the ATSPTDC the travel time of each arc $\left(v_i^k, v_j^l\right)$ is greater than or equal to a time unit (5, 2 or 1 minute in these instances), and some original ATSPTW instances have several vertices with the same or similar tight time windows, these instances may have no ATSPTDC solution; this will happen if it is impossible to visit all vertices in their time windows consuming at least one time unit each time an arc is traversed, especially for the 5-minute time unit. Therefore, in order to guarantee a priori the existence of a solution in the generated ATSPTDC instances, we decided to remove some vertices from each original instance, obtaining $|V|$-vertex ATSPTDC instances from ATSPTW instances with the smallest number of greater-than $|V|$ vertices.

- We weighted the time windows corresponding to a working day from 8:00h. to 20:00h. in a department store in all the resulting instances. That is, the depot time window $[a_0, b_0]$ in the benchmark problem is always considered to be the time interval from 8:00 to 20:00, expressed in minutes as $[0, 720]$. Therefore, for all i, a_i and b_i are multiplied by $720/b_0$ and rounded to be multiple of 10 (divisible by all the time units considered: 5, 2 and 1). Table 2 shows an example of this weighting of the time windows, with one depot and two customers: as $b_0 = 547$, for example, the original time window $[139, 147]$ is first transformed into $[139 \cdot 720/547, 147 \cdot 720/547] = [182.96, 193.49]$ and then rounded to $[180, 190]$.

- Finally, in all these ATSPTDC instances we considered the cost and the travel time of traversing an arc (v_i, v_j) at time t_i^k as the integer part of $p(t_i^k) \cdot |\,(v_i, v_j)\,|$, p being a weighting function that depends on the time interval

Table 2: Example of weighting of the time windows

Original windows		Weighted windows	
a_i	b_i	a_i	b_i
0	547	0	720
139	147	180	190
62	83	80	110

to which t_i^k belongs, and $|(v_i, v_j)|$ being the Euclidean distance between v_i and v_j. We have partitioned the working day into time intervals (see Table 3) considering traffic density in a large city of Spain, i.e. the rush-hours, such as leaving from or going to school or work. Note that in the original ATSPTW instances the travel time between two vertices is equal to the Euclidean distance between them given the coordinates of the vertices. Thus, through Table 3 and the definition of our travel times, we have the relation between the original ATSPTW travel times and the time-dependent travel times for each time interval of the working day.

Table 3: Time intervals and weights

Time interval	$p(t_i^k)$
[8:00, 9:40[	$p(t_i^k) = 1$
[9:40, 11:40[	$p(t_i^k) = 0.5$
[11:40, 12:40[	$p(t_i^k) = 0.75$
[12:40, 13:30[	$p(t_i^k) = 0.65$
[13:30, 15:20[	$p(t_i^k) = 1$
[15:20, 16:20[	$p(t_i^k) = 0.5$
[16:20, 18:40[	$p(t_i^k) = 0.75$
[18:40, 20:00]	$p(t_i^k) = 1$

We constructed ATSPTDC instances with 10, 20, 30, 40, 50 and 60 vertices taking into account three of the different maximum widths of the time windows in the original ATSPTW instances (20, 40 and 60) and three time units (5,

2 and 1 minute). With all these data we generated sets of five instances for each number of vertices, each maximum width and each time unit, i.e. a total of $5 \times 6 \times 3 \times 3 = 270$ ATSPTDC instances, grouped into 3 sets of 90 instances depending on the time unit. It is worth commenting that as the original maximum width of the time windows (except for the depot) is 20, 40 or 60, the tightness of the time window constraints depends on each instance. For example, in the original instance from which Table 2 is obtained, $[a_0, b_0] = [0, 547]$; then, in our instances obtained from this ATSPTW instance, the maximum width is $20 \cdot 720/547 = 26.32$ minutes (very tight for a 5-minute time unit), 52.65 minutes or 78.97 minutes respectively.

For each one of these 270 instances we first constructed the auxiliary digraph G', then the reduced auxiliary digraph G"; in G" we defined the corresponding AGTSP; using Noon and Bean transformation, we constructed the ATSP instance to be solved, and finally we transformed the ATSP instance into a GATSP instance by adding a large positive number to each arc cost in order to assure the occurrence of a Hamiltonian cycle in the optimal solution provided a Hamiltonian cycle exists (note that from an ATSPTDC instance we obtain an ATSP instance whose digraph is far from being a complete digraph).

As the GATSP is a particular case of the MGRP, we used the exact algorithm for the MGRP given in [11] to optimally solve our ATSP instances. This algorithm is a cutting-plane procedure based on the polyhedral study on the MGRP presented in [10] and [11] in which the branch-and-bound option of CPLEX [12] is invoked when violated inequalities are not found. The algorithm is coded in *C* and run on a PC with a 1.8 GHz Pentium IV processor, using CPLEX 8.0 as an LP solver. Note that we have its original code available, thus we have implemented an algorithm that performs all the transformations cited above until we obtain a standard MGRP instance as input-data for the code.

At this moment, a logical question is: why do we use the MGRP exact procedure mentioned above (hereinafter CMS), to solve the resulting ATSP

instances instead of using one of the existing ATSP exact procedures mentioned in Section 1?

The following considerations should be made:

- The branch and bound ATSP solvers given in [7] and [15] (hereinafter CDT and FT-*b*&*b* respectively, both based on an AP lower bound), are not useful for non-asymmetric instances, e.g. quasi symmetric TSP instances, as recognized the authors. As for the branch-and-cut ATSP solver given in [16] (hereinafter FT-*b*&*c*), which works better than the other two algorithms according to the computational experiments presented in [17], its authors also admit that for almost symmetric instances a substantial improvement can be obtained by exploiting new separation procedures.

- The ATSP is defined in a complete digraph and the input data for these three ATSP solvers is then a $|V| \times |V|$ matrix; this means that the solvers work worse with sparse digraphs because each non-existing arc must be considered as an existing arc with a high-enough cost. This could lead to a quasi symmetric instance in which almost all the arcs have a very high cost.

- Our ATSP instances are very sparse: the smallest instances have at most 20% finite cost arcs, while the biggest instances only have less than 2% finite cost arcs. For example, we have several ATSP instances with more than 2000 vertices and about 60000 arcs, which means that the input data for those ATSP solvers will have more than 4000000 arcs with very high (infinite) cost.

- The CMS code only requires the existing arcs as input data in the GATSP. For example, for an instance with 2000 vertices and 60000 arcs we save 3938000 high costs in the input data of the CMS code as compared to the three ATSP solvers analyzed.

- FT-*b*&*c* was implemented by its authors to solve ATSP instances with up to 200 vertices; however many of our ATSP instances have more than 1000 vertices, and some of them more than 2000 vertices.

With all these considerations, especially due to the characteristics of our generated ATSP instances, one could expect that the CMS code obtained

better results in these instances than the three ATSP solvers. Additionally, we have compared the FT-*b*&*b* and the CDT codes with the CMS code on 60 of our smallest ATSP instances: those belonging to the sets with 5-minute and 2-minute time units and with 10 and 20 vertices in the ATSPTDC instance. To avoid symmetry in the complete ATSP digraphs in order to obtain a better performance of both ATSP solvers, and taking into account the size of the costs, we have identified an infinite cost with a random integer value between 1000 and 2500. In our computational experiments the CDT code obtained better results than FT-*b*&*b*, which in some instances could not obtain any upper bound and in others caused system errors (probably due to the data size and not to the dimension of the *b*&*b* storage vector). In tables 4 and 5 we show the running times obtained with the CMS code and with the CDT code, for which the dimension of the *b*&*b* storage vector has been changed from the original 15000000 to 90000000 (a greater dimension gives rise to system errors in our computer), with the following notation:

- Name: Name of the problem.

- V: number of vertices in the resulting ATSP instance.

- % Arcs: percentage of existing arcs with respect to the complete digraph in the ATSP instance.

- Cost: Cost of the optimal solution.

- CMS: Running time in seconds to obtain the optimal solution with the CMS code.

- CDT: Running time in seconds to obtain the optimal solution with the CDT code, having as a limit the minimum between 2 hours and the time consumed until the saturation of the *b*&*b* storage vector. When the optimal solution is not obtained, we show in brackets the best upper bound obtained.

We have that the CDT code obtains good running times in the instances obtained from ATSPTDC instances with 10 vertices, even with lower running times than the CMS code in the smallest instances, but only in three out of the 30 instances obtained from 20-vertex ATSPTDC instances the optimal

Table 4: Comparison of CMS and CDT in 5-minute time unit instances with 10 and 20 vertices

Name	V	% Arcs	Cost	CMS	CDT
P10w201	38	19.35	1175	1.37	0.10
P10w202	50	14.90	1132	0.77	0.16
P10w203	47	15.77	1162	1.37	0.21
P10w204	49	15.35	1195	1.05	0.43
P10w205	41	15.77	1161	0.77	0.10
P10w401	79	9.53	1128	1.10	0.98
P10w402	76	9.96	1158	0.93	0.82
P10w403	73	10.41	1193	1.04	0.21
P10w404	75	9.95	1148	1.10	0.38
P10w405	80	9.02	1161	0.94	0.38
P10w601	97	8.08	1159	1.21	1.26
P10w602	117	6.37	1103	1.37	2.36
P10w603	102	7.82	1128	1.21	12.19
P10w604	107	7.26	1090	1.59	4.33
P10w605	79	9.53	1085	1.32	1.36
P20w201	95	13.21	2276	3.68	>2h. (2811)
P20w202	94	13.20	2268	2.03	>2h. (3680)
P20w203	95	12.86	2252	1.10	>2h. (3713)
P20w204	93	13.15	2250	1.43	64.70
P20w205	95	13.38	2208	2.26	>2h. (2632)
P20w401	160	7.81	2251	3.24	>4044 (4653)
P20w402	163	7.56	2167	3.57	>4158 (3788)
P20w403	150	8.70	2193	3.35	>4352 (2836)
P20w404	165	7.27	2280	3.07	>3437 (2281)
P20w405	144	8.44	2286	6.04	>4483 (3216)
P20w601	199	6.10	2166	7.25	>2776 (3770)
P20w602	222	6.21	2235	21.70	>2267 (2360)
P20w603	207	6.35	2217	6.48	>2319 (3235)
P20w604	216	6.02	2207	7.85	>2086 (2701)
P20w605	214	5.90	2195	5.49	>2152 (2432)

Table 5: Comparison of CMS and CDT in 2-minute time unit instances with 10 and 20 vertices

Name	V	% Arcs	Cost	CMS	CDT
P10w201	77	9.86	1175	1.76	0.00
P10w202	98	7.61	1133	0.98	0.05
P10w203	90	8.05	1162	0.99	0.21
P10w204	97	8.46	1195	1.15	0.00
P10w205	90	8.05	1161	0.93	0.21
P10w401	172	4.39	1128	1.43	4.44
P10w402	167	4.61	1157	1.81	8.84
P10w403	157	4.90	1192	1.10	0.49
P10w404	161	4.62	1148	1.60	0.10
P10w405	172	4.21	1161	1.10	0.76
P10w601	198	3.748	1159	2.42	0.43
P10w602	275	2.68	1103	1.70	4.00
P10w603	238	3.40	1125	2.03	34.43
P10w604	231	3.18	1090	2.31	432.97
P10w605	178	4.28	1084	1.53	65.74
P20w201	193	6.54	2279	4.12	>3192 (6049)
P20w202	187	6.71	2268	2.86	577.10
P20w203	188	6.54	2252	2.31	>2991 (5069)
P20w204	173	7.12	2151	1.48	89.14
P20w205	183	6.93	2211	2.80	>2840 (3208)
P20w401	342	3.65	2252	3.85	>1208 (6147)
P20w402	356	3.41	2165	6.70	>1099 (6945)
P20w403	331	3.95	2197	5.83	>940 (4266)
P20w404	367	3.27	2280	6.15	>789 (2288)
P20w405	317	3.76	2286	7.52	>1324 (5005)
P20w601	463	2.59	2163	20.76	>777 (5007)
P20w602	524	2.60	2235	67.34	>620 (4033)
P20w603	491	2.64	2210	11.86	>547 (4150)
P20w604	511	2.50	2224	21.31	>647 (5960)
P20w605	505	2.48	2195	19.01	>646 (4180)

solution was obtained within a reasonable running time (64, 577 and 89 seconds respectively against 1.43, 2.86 and 1.48 seconds respectively consumed by the CMS code), while the average running time of the CMS code was only of few seconds for the 30 instances.

Note that, as expected, the time limit to obtain the optimal solution with the CDT code (minimum between 2 hours and the time consumed until saturation of the $b\&b$ storage vector) decreases with the size of the instance, its minimum value, 547 seconds, being for problem p20w603 in Table 2. But even for this problem, the running time with CMS is 11.86 seconds, i.e. the advantage of using CMS is evident.

Nevertheless, we are convinced that in very asymmetric instances with few infinite cost arcs, the three ATSP solvers mentioned here show a better performance than the CMS code. For instance, we have compared the results obtained with the four exact procedures in 21 ATSP instances collected in TSPLIB [30]. Although CMS solved the 21 instances within a reasonable time, according to the results given in [17], FT-$b\&c$ ran faster than CMS in all instances except in the classical real-word instance p43: 9.3 seconds in their machine against 3.57 seconds in ours. Except for few cases, FT-$b\&b$ and CDT were also faster than CMS in these 21 instances. Note that instance p43 (that could not be solved optimally by FT-$b\&b$ nor by CDT) has about 10% arcs with cost greater than 1000 and according to the rest of the costs and the cost of the optimal solution (595) this 10% can be considered with infinite cost.

In tables 6 to 8 we show the average running times obtained with the CMS code in the 270 generated ATSPTDC instances. Each row corresponds to a set of 5 similar instances, with the following notation:

- V: number of vertices in the ATSPTDC instance including the depot.

- W: maximum width of the time windows in the original ATSPTW instance from which the ATSPTDC instance has been constructed.

- VG": average number of vertices in G" rounded to integer.

- ANA: average number of arcs in the GATSP instances obtained from G"

rounded to integer.

- O: number of instances optimally solved in less than three hours.

- AT: average time in seconds to obtain the optimal solution in the solved (in less than three hours) GATSP instances obtained from G".

- WT: worst time in seconds to obtain the optimal solution in the solved GATSP instances obtained from G".

As expected, the running time of this exact and exponential algorithm increases with the number of vertices in the ATSPTDC instance, with the width of the time windows and with a smaller time unit, because all of them increase the number of vertices and the number of arcs in the corresponding GATSP instance to be solved by the algorithm.

We point out that instances with 10 vertices are optimally solved in less than one second or within few seconds, even for 1-minute time unit and instances with 30 vertices are solved in less than one minute or few minutes. Furthermore, we have been able to optimally solve instances with up to 60 vertices, although with larger running times. From the tables we can see that the instances with 60 vertices produce very large-scale GATSP instances, even with more than 2,000 vertices and 60,000 arcs. Nevertheless, only 10 out of the 45 instances with 60 vertices were not solved after 3 hours of running time. Note that for a real delivery route inside a large city with traffic problems, it seems very improbable to serve more than 40 or 50 customers in a working day.

According to the size of the instances solved here, we believe that the exact procedure for the MGRP proposed in [11] is a good tool to optimally solve at least real ATSPTDC instances with several dozens of customers within a reasonable time, even with a time unit equal to 1 minute, which is the smallest time unit normally considered in real vehicle routing problems inside large cities.

By the same reason, although from a theoretical point of view the ATSPTW is a particular case of the ATSPTDC, we can not expect a good behavior

Table 6: Computational results with the 5-minute time unit

V	W	VG"	ANA	O	AT	WT
10	20	45	320	5	1.06	1.37
10	40	76	564	5	1.02	1.10
10	60	100	766	5	1.34	1.59
20	20	94	1166	5	2.10	3.68
20	40	156	1926	5	3.85	6.04
20	60	211	2727	5	9.75	21.70
30	20	138	2409	5	3.33	4.40
30	40	225	4051	5	16.13	27.41
30	60	253	4600	5	36.31	86.24
40	20	184	4192	5	7.90	13.84
40	40	314	7278	5	46.68	67.99
40	60	447	10384	5	918.31	2135.61
50	20	239	6678	5	18.95	24.39
50	40	386	11007	5	180.07	365.97
50	60	513	14711	5	468.36	1010.69
60	20	297	9805	5	39.58	51.02
60	40	476	16052	4	2929.64	5450.47
60	60	622	20956	4	1128.59	1438.70

Table 7: Computational results with the 2-minute time unit

V	W	VG"	ANA	O	AT	WT
10	20	90	668	5	1.16	1.76
10	40	165	1240	5	1.40	1.81
10	60	224	1687	5	1.99	2.42
20	20	184	2295	5	2.71	4.12
20	40	342	4206	5	6.01	7.52
20	60	498	6370	5	28.05	67.34
30	20	269	4699	5	10.18	21.04
30	40	493	8775	5	38.77	71.46
30	60	574	10325	5	83.56	219.21
40	20	360	8129	5	15.03	24.16
40	40	690	15792	5	277.78	822.12
40	60	1041	24676	5	1997.25	6937.36
50	20	474	13120	5	39.81	46.96
50	40	850	23973	5	1017.48	2878.21
50	60	1200	33893	5	2039.66	5197.22
60	20	601	19722	5	112.05	162.36
60	40	1053	35066	4	3601.94	6295.39
60	60	1451	48293	4	5515.66	10385.97

Table 8: Computational results with the 1-minute time unit

V	W	VG"	ANA	O	AT	WT
10	20	180	1438	5	1.68	3.13
10	40	314	2349	5	2.29	2.75
10	60	422	3113	5	4.45	5.27
20	20	357	4415	5	4.74	8.41
20	40	707	8603	5	18.31	30.54
20	60	970	12266	5	94.49	146.38
30	20	517	8789	5	12.68	16.86
30	40	940	16582	5	114.08	228.28
30	60	1113	19813	5	182.58	354.76
40	20	741	16676	5	45.72	84.09
40	40	1316	29704	5	895.57	1670.34
40	60	1943	43937	5	3049.60	5497.44
50	20	909	24869	5	96.04	135.89
50	40	1631	45368	5	1740.35	3283.83
50	60	2343	65114	5	5916.77	8936.43
60	20	1145	37219	5	242.88	389.25
60	40	2021	66419	4	7735.53	10252.78
60	60	2655	86238	0	-	-

of our procedure to solve general ATSPTW instances, especially if the time windows are not tight. For example, in the set of ATSPTW instances given in [2], which are real-life stacker crane instances, some of the smallest instances (with respect to the number of customers) contain many customers having time windows $[a_i, b_i]$ with $b_i - a_i$ about 4800. If we think of daily routes with 1-minute time unit, most of the customers will have time windows with about 80 hours width.

Nevertheless, if we think of ATSPTW instances with at most several dozens of customers with tight time windows, although our procedure can not compete in running time with the specific exact procedures for the ATSPTW (see for example [2] and [14]), it can optimally solve these instances within a reasonable time (seconds or minutes). As an example, in Table 9 we show the average running times obtained with our procedure for the exact resolution of 30 of the benchmark ATSPTW instances we have used. We do not show the results for larger instances because the tendency of the running time is evident. In Table 9 each row corresponds to a set of 5 similar instances, with the following notation:

- n: number of customers in the ATSPTW instance.

- W: maximum width of the time windows in the ATSPTW instance.

- AT_{CMS}: average time in seconds to obtain the optimal solution with the CMS code for the MGRP.

- AT_{DDGS}: average time in seconds given in [14] to obtain the optimal solution with the ATSPTW code presented in that paper.

5 Conclusions

Routing problems with time-dependent costs have hardly been studied because they are very difficult to model and solve. In this paper we have presented a generalization of the well-known ATSPTW in which the time and the cost of traversing an arc depend on the period of time at which we start traversing

Table 9: Average running times for ATSPTW instances

n	W	AT_{CMS}	AT_{DDGS}
20	20	4.48	0.02
20	40	22.10	0.05
40	20	57.28	0.08
40	40	962.57	0.24
60	20	222.44	0.15
60	40	8482.44	0.85

it; in this way more accurate solutions can be obtained for some real vehicle routing problems inside large cities, in which the time or cost of traversing certain streets depends on the moment of the day. This generalization can be transformed into an AGTSP and then into the classical ATSP for which several heuristic and exact procedures exist, that have been applied with good performance also to large-scale instances with several thousands of vertices.

We have presented a computational study on a set of 270 ATSPTDC instances adapted from benchmark ATSPTW instances. To obtain the optimal solutions we have applied the exact algorithm for the MGRP proposed in [11] to the conveniently modified instances. Based on our findings, we believe that this exact algorithm is a good tool to optimally solve real ATSPTDC instances with several dozens of customers within a reasonable time -as no more customers are likely to be served in a working day- even with a 1-minute time unit. Nevertheless, as in many real-world routing problems a single tour is part of a multivehicle problem with several hundreds of customers, in future research we will try to extend these results to the Vehicle Routing Problem with Time Windows.

We are convinced that as computer power and speed increase, more and more researches on routing problems will take into account time-dependent costs in order to move the mathematical models closer to real-world problems. Recent papers cited here justify our intuition. In this way, the theoretical

results presented here can be used in the future as ideas or tools to test the efficiency of specific procedures for routing problems with time-dependent costs.

Acknowledgements

Authors would like to thank Michel Gendreau, Alain Hertz, Gilbert Laporte and Mihnea Stan for providing us the set of benchmark ATSPTW instances. We are also grateful to Mateo Fischetti and Norbert Ascheuer for their suggestions and help about the computational experiments.

This work has been partially supported by the Ministerio de Ciencia y Tecnología of Spain (project TIC2003-05982-C05-01) and the Generalitat Valenciana (Ref: GRUPOS03/189).

References

[1] N. Ascheuer, M. Fischetti, M. Grötschel, A polyhedral study of the asymmetric traveling salesman problem with time windows, Networks 36 (2000) 69-79.

[2] N. Ascheuer, M. Fischetti, M. Grötschel, Solving the asymmetric travelling salesman problem with time windows by branch-and-cut, Mathematical Programming Ser A 90 (2001) 475-506.

[3] E. Benavent, D. Soler, The directed rural postman problem with turn penalties, Transportation Science 33 (1999) 408-418.

[4] M. Blais, G. Laporte, Exact solution of the generalized routing problem through graph transformations, Journal of the Operational Research Society 54 (2003) 906-910.

[5] W.B. Carlton, J.W. Barnes, Solving the traveling-salesman problem with time windows using tabu search, IIE Transactions 28 (1996) 617-629.

[6] G. Carpaneto, M. Dell'Amico, P. Toth, Exact solution of large-scale, asymmetric traveling salesman problems, ACM Transactions on Mathematical Software 21 (4)(1995a) 394-409.

[7] G. Carpaneto, M. Dell'Amico, P. Toth, Algorithm 750: CDT: A subroutine for the exact solution of large-scale, asymmetric traveling salesman problems, ACM Transactions on Mathematical Software 21 (4)(1995b) 410-415.

[8] S. Chopra, G. Rinaldi, The graphical asymmetric traveling salesman polyhedron: symmetric inequalities, SIAM Journal on Discrete Mathematics 9 (4)(1996) 602-624.

[9] A. Corberán, R. Martí, E. Martínez, D. Soler, The rural postman problem on mixed graphs with turn penalties, Computers & Operations Research 29 (2002) 887-903.

[10] A. Corberán, A. Romero, J.M. Sanchis, The mixed general routing polyhedron, Mathematical Programming Ser A 96 (2003) 103-137.

[11] A. Corberán, G. Mejía, J.M. Sanchis, New results on the mixed general routing problem, Operations Research 53 (2005) 363-376.

[12] ILOG S.A., ILOG CPLEX 8.0, 2002.

[13] G. Desaulniers, D. Villeneuve, The shortest path problem with time windows and linear waiting costs, Transportation Science 34 (2000) 312-319.

[14] Y. Dumas, J. Desrosiers, E. Gelinas, M.M. Solomon, An optimal algorithm for the traveling salesman problem with time windows, Operations Research 43 (2)(1995) 367-371.

[15] M. Fischetti, P. Toth, An additive bounding procedure for the asymmetric traveling salesman problem, Mathematical Programming 53 (1992) 173-197.

[16] M. Fischetti, P. Toth, A polyhedral approach to the asymmetric traveling salesman problem, Management Science 43 (1997) 1520-1536.

[17] M. Fischetti, A. Lodi, P. Toth, 2002. Exact methods for the ATSP. In G. Gutin, A.P. Punnen (Eds), The traveling salesman problem and its variants. Kluwer Academic Publishers; 2002. 169-206.

[18] B. Fleischmann, M. Gietz, S. Gnutzmann, Time-varying travel times in vehicle routing, Transportation Science 38 (2004) 160-173.

[19] F. Focacci, A. Lodi, M. Milano, A hybrid exact algorithm for the TSPTW, INFORMS Journal on Computing 14 (4)(2002) 403-417.

[20] S. Ichoua, M. Gendreau, J.Y. Potvin, Vehicle dispatching with time-dependent travel times, European Journal of Operational Research 144 (2003) 379-396.

[21] M. Gendreau, A. Hertz, G. Laporte, M. Stan, A generalized insertion heuristic for the traveling salesman problem with time windows, Operations Research 46 (3)(1998) 330-335.

[22] A. Haghani, S. Jung, A dynamic vehicle routing problem with time-dependent travel times, Computers & Operations Research 32 (2005) 2959-2986.

[23] G. Laporte, Modeling and solving several classes of arc routing problems as traveling salesman problems, Computers & Operations Research 24 (1997) 1057-1061.

[24] C. Malandraki, M.S. Daskin, Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms, Transportation Science 26 (3)(1992) 185-200.

[25] C. Malandraki, R.B. Dial, A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem, European Journal of Operational Research 90 (1996) 45-55.

[26] C.E. Noon, J.C. Bean, An efficient transformation of the generalized traveling salesman problem, INFOR 31 (1993) 39-44.

[27] G. Pesant, M. Gendreau, J.Y. Potvin, J.M. Rousseau, An exact constraint logic programming algorithm for the traveling salesman problem with time windows, Transportation Science 32 (1)(1998), 12-29.

[28] J.Y. Potvin, Y. Xu, I. Benyahia, Vehicle routing and scheduling with dynamic travel times, Computers & Operations Research 33 (4)(2006) 1129-1137.

[29] R. Wolfer Calvo, A new heuristic for the traveling salesman problem with time windows, Transportation Science 34 (1)(2000) 113-124.

[30] http://www.iur.uni-heidelberg.de/groups/comopt/software/TSPLIB95/