European Journal of Operational Research 236 (2014) 891–902

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Facets and valid inequalities for the time-dependent travelling salesman problem

Juan José Miranda-Bront [a,b,*], Isabel Méndez-Díaz [a], Paula Zabala [a,b]

[a] Departamento de Computación, FCEyN, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, C1428EGA, CABA, Argentina
[b] Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

ARTICLE INFO

Article history:
Available online 31 May 2013

Keywords:
Combinatorial optimization
Integer programming
Time-Dependent TSP
Branch and Cut

ABSTRACT

The Time-Dependent Travelling Salesman Problem (TDTSP) is a generalization of the traditional TSP where the travel cost between two cities depends on the moment of the day the arc is travelled. In this paper, we focus on the case where the travel time between two cities depends not only on the distance between them, but also on the position of the arc in the tour. We consider two formulations proposed in the literature, we analyze the relationship between them and derive several families of valid inequalities and facets. In addition to their theoretical properties, they prove to be very effective in the context of a Branch and Cut algorithm.

1. Introduction and literature review

The *Time-Dependent Travelling Salesman Problem* (TDTSP) is a generalization of the classical Travelling Salesman Problem (TSP) in which the cost of the travel between two cities depends not only on the distance between them, but also on the time of the day the arc is travelled.

This problem can be stated as follows: Consider a complete digraph $D = (V,A)$, with $V = \{0,1,\ldots,n\}$ the set of vertices and A the set of directed arcs. Assume that there exists a discrete time horizon 0, 1, ..., T and an $(n+1)\times(n+1)$ time-dependent cost matrix $C(t) = [c_{ij}(t)]$, $0 \leqslant t \leqslant T$, that associates a cost $c_{ij}(t)$ to each arc (i,j) in A, where $c_{ij}(t)$ is a time-dependent cost function for the arc (i,j) if departing from i at instant t. Vertex 0 is a special vertex representing the depot. The TDTSP involves finding a minimum cost tour that visits each vertex exactly once, starting at and returning to vertex 0.

The TDTSP can handle many interesting practical applications. One of the main advantages is that it allows us to model some real world situations that the classical TSP cannot deal with. For example, the travel time between any two cities may be different if the arc is taken early in the morning or at noon, or if it is known that traffic jams may occur during rush hours, etc. The TDTSP can model this kind of scenarios by using an appropriate cost function for each arc.

In its simplest version, TDTSP assumes that the travel time between any two cities is one time period, meaning that the travel time function depends on the distance between the cities and on the position of the arc in the tour. The time-dependent cost function for arc (i,j), $c_{ij}(t)$, can be expressed as a step function with one constant value for each time period $k = 1, \ldots, n$, named c_{ijk}. In addition to the advantages mentioned above, this version of TDTSP can be used to model different scheduling and assignment problems.

Fox (1973) proposes TDTSP formulations to solve the problem of scheduling production with time-dependent staff requirements. Picard and Queyranne (1978) give three integer programming formulations-two of them are linear-for TDTSP to model a scheduling problem with time-dependent transition costs. They use these formulations to minimize the tardiness costs in one machine scheduling and they report instances with up to 20 vertices solved to optimality. Fox et al. (1980) give a new formulation with $O(n^3)$ variables and only n constraints, but no computational results are reported. Gouveia and Voss (1995) present two new formulations for the TDTSP derived from a Quadratic Assignment Problem (QAP), and establish a relation among the previously mentioned formulations and the new ones in terms of the tightness of the linear relaxations.

Almost at the same time, Vander Wiel and Sahinidis (1995) present a new formulation for the TDTSP which results to be the same as the one proposed by Gouveia and Voss (1995). The only difference between them is that Vander Wiel and Sahinidis (1995) prove that the integrality condition of some variables can be relaxed. As a result, their formulation has $O(n^3)$ variables, but only $O(n^2)$ are required to be binaries. The main idea of the model is to reformulate the TDTSP as the problem of finding the shortest

* Corresponding author at: Departamento de Computación, FCEyN, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, C1428EGA, CABA, Argentina.
E-mail addresses: jmiranda@dc.uba.ar (J.J. Miranda-Bront), imendez@dc.uba.ar (I. Méndez-Díaz), pzabala@dc.uba.ar (P. Zabala).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.05.022

constrained path in a directed multi-partite graph. They use a Benders-based heuristic to compute upper and lower bounds for their formulation of TDTSP. Computational experience shows that this procedure achieves very good bounds with minimal computational effort. In a follow up paper, Vander Wiel and Sahinidis (1996) develop an exact algorithm for the TDTSP. Preliminary computational results show instances with at most 20 cities solved to optimality.

A very interesting approach is proposed by Bigras et al. (2008). The authors study the path formulation proposed in Picard and Queyranne (1978) and evaluate the improvements in the LP relaxations of pricing path without four cycles. Cutting planes are applied at the root node, considering different families of valid inequalities of TSP as well as clique inequalities obtained by constructing the conflict graph using incompatibilities specially inferred for the TDTSP. They evaluate a Branch-Cut and Price (B&C&P) algorithm on different instances and are able to solve randomly generated instances having up to 50 vertices as well as some instances from the TSPLIB. In two very recent papers, Abeledo et al. (2010) provide good computational results using a B&C&P based on the approach of Bigras et al. (2008) and Godinho et al. [Godinho et al., 2011] present an extended formulation based on Picard and Queyranne's model for which the lower bound given by the linear programming relaxation has gap close to zero on the instances tested.

This simplified version of TDTSP also generalizes the well-known *Travelling Deliveryman Problem* (TDP). Given a vehicle depot and a number of customers with known travel times between each pair of them, the TDP involves finding a path starting at the depot and visiting every customer exactly once that minimizes the sum of the times required to reach every customer. In particular, this objective function can be captured defining $c_{ijk} = (n - k + 1)c_{ij}$, where c_{ij} denotes the travel time from vertex i to vertex j. Exact solution algorithms for the TDP are available in Lucena (1990); Fischetti et al. (1993); van Eijl (1995); and Méndez-Díaz et al. (2008).

In Picard and Queyranne (1978), one of the models is a three-index integer linear programming formulation. Méndez-Díaz et al. (2008) tested it for TDP instances and the results obtained for instances having up to 30 vertices show a reasonable tradeoff between the quality of the bound provided by the LP relaxation and the computational effort required. In addition, Gouveia and Voss (1995) prove that this formulation is equivalent to the one presented in Vander Wiel and Sahinidis (1995) in terms of the linear relaxation, and that this value is an upper bound for the rest of the formulations considered.

These results suggest that both models look promising to be used in a Branch and Cut (B&C) algorithm. Picard and Queyranne (1978) use a Branch and Bound (B&B) algorithm. Vander Wiel and Sahinidis (1996) developed a B&C algorithm to solve the master problem of the Benders decomposition, but they only use general purpose cuts for a restricted set of inequalities. They suggest that, as future research, it would be interesting to study the polyhedron of the TDTSP. The aim of this paper goes in that direction. We consider the models presented in Picard and Queyranne (1978) and Vander Wiel and Sahinidis (1995). Since both models are a linearization of QAP, it results that their polytopes are strongly related. We derive several families of valid inequalities and facets for both polytopes. In particular, we provide a theoretical framework that allows to derive further families of facets by means of applying a maximum sequential lifting procedure over some specific set of variables. Finally, in order to analyze their computational behavior, we incorporate them in a B&C algorithm which is tested over instances from the related literature, obtaining very good computational results.

The paper is organized as follows. In Section 2 we introduce the models from Picard and Queyranne (1978) and Vander Wiel and Sahinidis (1995) and establish a strong relation between them. In Section 3 we introduce a new family of valid inequalities based on the well known cycle inequalities for the ATSP. We also analyze some polyhedral properties for this family, which is used to derive five families of facets, and present another four families of valid inequalities. In Section 4 we give the outline of the B&C algorithm, which is tested and compared with other exact algorithms in Section 5. Finally, in Section 6, we present some conclusions and future research.

2. Models

2.1. Picard and Queyranne

This model is proposed in Picard and Queyranne (1978). It uses a set of binary decision variables y_{ijk}, where $y_{ijk} = 1$ if city j is visited in time period $k + 1$ after city i was visited in time period k. By forcing vertex 0 to be the depot, we can remove from the formulation the variables $y_{ij0} \forall i \geqslant 1, y_{ijn} \forall j \geqslant 1, y_{i0k} \forall k \leqslant n-1, y_{0jk} \forall k \geqslant 1$, given that they always take value zero. The formulation is shown below

$$(\mathrm{PQ})\ \min \sum_{j=1}^{n} c_{0j0} y_{0j0} + \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{j \neq i} \sum_{k=1}^{n-1} c_{ijk} y_{ijk} + \sum_{j=1}^{n} c_{j0n} y_{j0n} \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} y_{0j0} = 1 \tag{2}$$

$$\sum_{i=1, i \neq j}^{n} y_{ji1} = y_{0j0} \quad j = 1, \ldots, n \tag{3}$$

$$\sum_{i=1, i \neq j}^{n} y_{ijk} = \sum_{i=1, i \neq j}^{n} y_{jik+1} \quad j = 1, \ldots, n; \quad k = 1, \ldots, n-2 \tag{4}$$

$$\sum_{i=1, i \neq j}^{n} y_{ijn-1} = y_{j0n} \quad j = 1, \ldots, n \tag{5}$$

$$y_{0j0} + \sum_{i=1, i \neq j}^{n} \sum_{k=1}^{n-1} y_{ijk} = 1 \quad j = 1, \ldots, n \tag{6}$$

$$y_{ijk} \in \{0,1\} \quad \begin{matrix} i = 1, \ldots, n; \quad j = 1, \ldots, n \\ k = 1, \ldots, n-1; \quad i \neq j \end{matrix}$$

$$y_{j0n}, y_{0j0} \in \{0,1\} \quad j = 1, \ldots, n,$$

The objective function minimizes the total travel cost of the tour. Eq. (2) forces the vehicle to depart from the depot in time period 0. Eqs. (3)–(5) ensure that the selected transitions are travelled in consecutive time periods. Constraints (6) establish that the vehicle must arrive to each vertex $j \in V \backslash \{0\}$ in exactly one time period. Finally, integrality conditions on variables are imposed.

Feasible solutions satisfying constraints (2)–(6) correspond to tours with transitions travelled in consecutive time periods, starting from the depot in time period 0. It is important to remark that this formulation does not allow subtours.

2.2. Vander Wiel and Sahinidis

We also consider the TDTSP formulation of Vander Wiel and Sahinidis (1995) (VW). The model is a linearization of the QAP presented in Picard and Queyranne (1978), and it can be seen as the problem of finding the shortest constrained path in a directed multi-partite graph. We show this formulation in a slightly different way than the one in Vander Wiel and Sahinidis (1995) because we force the vertex 0 to be the depot.

The QAP formulation uses a set of binary decision variables x_{ik}, where $x_{ik} = 1$ if city i is assigned to time period k, and $x_{ik} = 0$ otherwise. This model is quadratic because of the presence of the

product between $x_{i\ k-1}$ and $x_{i\ k}$ in the objective function for each possible combination of (i,j) and k.

In Vander Wiel and Sahinidis (1995), variables x_{ik} are referred as the *assignment variables*. To linearize the objective function of the QAP, they define the *transition variables*, y_{ijk}, which have the same meaning as the ones defined in the previous section. Moreover, they prove (see Proposition 1 of Vander Wiel and Sahinidis (1995) for a detailed proof) that $y_{ijk} = 1$ if and only if $x_{ik-1}x_{jk} = 1$, when variables y_{ijk} are considered as positive continuous variables. The advantage of this linearization is that it only introduces continuous variables to the original formulation of the QAP. See Vander Wiel and Sahinidis (1995, Section 1) for a detailed explanation and examples.

$$(\text{VW})\ \min \sum_{j=1}^{n} c_{0j0}y_{0j0} + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\sum_{k=1}^{n-1} c_{ijk}y_{ijk} + \sum_{j=1}^{n} c_{j0n}y_{j0n} \tag{7}$$

$$\text{s.t.} \sum_{k=1}^{n} x_{ik} = 1 \quad i = 1,\ldots,n \tag{8}$$

$$\sum_{i=1}^{n} x_{ik} = 1 \quad k = 1,\ldots,n \tag{9}$$

$$y_{0j0} = x_{j1} \quad j = 1,\ldots,n \tag{10}$$

$$\sum_{i=1,i\neq j}^{n} y_{ijk} = x_{jk+1} \quad j = 1,\ldots,n; \quad k = 1,\ldots,n-1 \tag{11}$$

$$y_{i0n} = x_{in} \quad i = 1,\ldots,n \tag{12}$$

$$\sum_{j=1,j\neq i}^{n} y_{ijk} = x_{ik} \quad i = 1,\ldots,n; \quad k = 1,\ldots,n-1 \tag{13}$$

$$\sum_{j=1}^{n} y_{0j0} = 1 \tag{14}$$

$$\sum_{j=1}^{n} y_{j0n} = 1 \tag{15}$$

$$y_{ijk} \geqslant 0, x_{ik} \in \{0,1\} \tag{16}$$

Similarly to model (PQ), the objective function minimizes the total travel cost of the tour. Eqs. (8) and (9) establish that each vertex must be assigned to exactly one time period and that each time period can be assigned to exactly one vertex, respectively. Eqs. (10) and (11) ensure that the arrival to a vertex is done according to the time period in which it is assigned. An analogous relation is established in Eqs. (12) and (13) for the departure of each vertex. Eqs. (14) and (15) establish that the vehicle must depart from the depot in time period 0 and arrive in time period n, respectively. Finally, integrality conditions on variables are imposed.

As in the previous formulation, by forcing vertex 0 to be the depot, we can remove from the formulation the variables $x_{i0}\forall i \geqslant 1$, $x_{0k}\forall k \geqslant 1$, $y_{ij0}\forall i \geqslant 1$, $y_{ijn}\forall j \geqslant 1$, $y_{i0k}\forall k \leqslant n-1$, $y_{0jk}\forall k \geqslant 1$, given that they always take value zero.

2.3. Relation between both models

As we mentioned in the introduction, (PQ) and (VW) are strongly related. It is easy to see that (PQ) is the projection of (VW) onto variables y_{ijk}, and Gouveia and Voss (1995) prove that these formulations are equivalent in terms of the linear relaxation. Formulation (VW) expresses each assignment variable, x_{ik}, in terms of the transition variables y_{ijk}. Considering the results shown in Balas and Oosten (1998), we deduce that there is a 1–1 correspondence between the faces of (PQ) and the faces of (VW). Moreover, if P^n_{PQ} and P^n_{VW} are the polytopes associated with models PQ and VW, respectively, we can also state that $dim\left(P^n_{PQ}\right) = dim\left(P^n_{VW}\right)$. From Müller (1996) we also know the dimension of P^n_{PQ}.

Theorem 1 Müller (1996). *The dimension of P^n_{PQ} is $n(n-1)(n-2)$ for $n \geqslant 5$.*

Then, if an inequality is valid for P^n_{PQ}, it is also valid for P^n_{VW} since variables y_{ijk} are considered in formulation (VW). Conversely, if an inequality is valid for P^n_{VW}, the projected version of the inequality is valid for P^n_{VW} since variables x_{ik} can be replaced by their expression in terms of variables y_{ijk}.

3. Polyhedral study

In this section we focus on deriving families of valid inequalities and facets. Based on the results presented in Section 2.3, these inequalities are valid for both P^n_{PQ} and P^n_{VW}, although most of the inequalities consider variables x_{ik}.

In Section 3.1 we propose a new family of valid inequalities exploiting the idea of *time dependent cycles*. For this family, we prove that they are facet defining on a polytope obtained by restricting P^n_{PQ} and P^n_{VW} and, from this characterization, we provide in Section 3.2 a procedure to obtain facets of P^n_{PQ} and P^n_{VW} by means of maximum sequential lifting. Finally, in Section 3.3 we derive further valid inequalities that express certain properties regarding time periods.

3.1. Time-dependent cycle inequalities

In this section we introduce a new family of valid inequalities based on the idea of the well known cycle inequalities for the asymmetric TSP. The main characteristic is that they include the time dependency of the transitions between vertices. Time dependent cycles are not allowed by the formulations considered in this paper. However, this family can be used to cut fractional solutions which can be useful in practice. Indeed, in the next section we will strengthen these inequalities by applying a lifting procedure.

Fig. 1 illustrates the idea behind time dependent cycles. The value associated with each arc stands for the time period in which it is travelled. Let $C = \langle v_1, v_2, v_3, v_4, v_1\rangle$ be a simple cycle with transitions between consecutive vertices travelled in time periods k, $k+1$, $k+2$, $k+3$. If we consider $n \geqslant 4$, then this time dependent cycle cannot be part of a feasible solution.

To forbid this situation, we can consider the following inequality

$$y_{v_1v_2k} + y_{v_2v_3k+1} + y_{v_3v_4k+2} + y_{v_4v_1k+3} \leqslant 3.$$

However, as we show next in Proposition 1, we can replace the right hand side by some of the vertex variables x_{ik} from formulation (VW) involved in the cycle, as shown below. This new expression dominates the previous one, since by definition variables x_{ik} are upper bounded by 1.

$$y_{v_1v_2k} + y_{v_2v_3k+1} + y_{v_3v_4k+2} + y_{v_4v_1k+3} \leqslant x_{v_2k+1} + x_{v_3k+2} + x_{v_4k+3}$$

We will refer to this family of inequalities as the Time-Dependent Cycle Inequalities (TDCIs). For the sake of notation, we express them in terms of both variables x_{ik} and y_{ijk}.

Proposition 1 (*TDCI*). *Let $C = \langle v_1, v_2, \ldots, v_l, v_{l+1} = v_1\rangle$, $l < n$, be a simple cycle with transitions between consecutive vertices travelled in time intervals k, $k+1$, …, $k+l-1$, $k+l \leqslant n$. Then, inequality*

$$\sum_{i=1}^{l} y_{v_iv_{i+1}k+i-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1}k+i} \tag{17}$$

is valid for P^n_{PQ} and P^n_{VW}.

Proof. From Eq. (11) we know that $y_{v_iv_{i+1}k+i-1} \leqslant x_{v_{i+1}k+i}$ for $i = 1,\ldots,l-1$ are valid. Adding these inequalities we get that

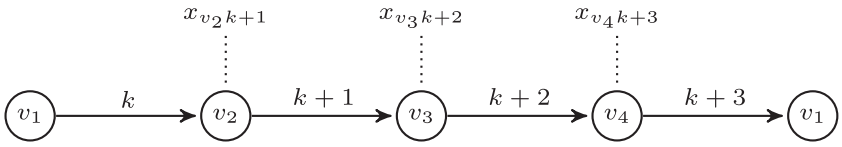


Fig. 1. TDCI for a cycle with four vertices.

$$\sum_{i=1}^{l-1} y_{v_i v_{i+1} k+i-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{18}$$

is a valid inequality. Using a similar argument, but now considering Eq. (13), inequalities $y_{v_i v_{i+1} k+i-1} \leqslant x_{v_i k+i-1}$ for $i = 2, \ldots, l$ are valid. Adding them and rewriting the right hand side we get that

$$\sum_{i=2}^{l} y_{v_i v_{i+1} k+i-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{19}$$

is also a valid inequality.

Considering inequalities (18) and (19), we can rewrite them as

$$y_{v_1 v_2 k} + \sum_{i=2}^{l-1} y_{v_i v_{i+1} k+i-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i}$$

$$\sum_{i=2}^{l-1} y_{v_i v_{i+1} k+i-1} + y_{v_l v_1 k+l-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i}$$

Given that $y_{v_1 v_2 k}$ and $y_{v_l v_1 k+l-1}$ cannot both take value one in a feasible solution, the inequality

$$\sum_{i=1}^{l} y_{v_i v_{i+1} k+i-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i}$$

is valid for P^n_{PQ} and P^n_{VW}. □

The TDCI do not define facets for P^n_{PQ} and P^n_{VW}. However, similarly to the cycle inequalities for the ATSP, TDCI define facets of a projection of P^n_{PQ} and P^n_{VW} over some specific variables. Let $C = \langle v_1, v_2, \ldots, v_l, v_{l+1} = v_1 \rangle$, $l < n$ and k as defined in Proposition 1. For notational convenience, we restrict our analysis to P^n_{VW}, although it is also valid for P^n_{PQ}. We define the following sets of variables.

- $F_1 = \{y_{v_l v_j k+j-2} = 0 : j = 2, \ldots, l-1\}$
- $F_2 = \{y_{v_l v_j k+l-1} = 0 : j = 2, \ldots, l-1\}$
- $F_3 = \{y_{v_i v_j k+j-2} = 0 : i = 3, \ldots, l-1, j = 2, \ldots, i-1\}$
- $F_4 = \{y_{v_i v_j k+i-1} = 0 : i = 2, \ldots, l-1, j = 1, \ldots, i-1\}$
- $F_5 = \{y_{v_1 v_j k+j-2} = 0 : j = 3, \ldots, l\}$,

and

$$P^n_{VW}(C,k) = P^n_{VW} \cap \bigcap_{i=1}^{5} F_i.$$

Variables characterizing subspaces $F_1, \ldots, F_5$ represent chords (v_i, v_j) of C, $v_i, v_j \in C$, travelled in valid time periods of departure from v_i or of arrival at v_j with respect to their corresponding positions in the cycle.

We begin characterizing the dimension of the restricted polytope $P^n_{VW}(C,k)$ in Theorem 2. Then, based on this result, we prove in Theorem 3 that the TDCI (17) are indeed facet defining of $P^n_{VW}(C,k)$. The importance of this result is twofold. First, it implies that the TDCI cannot be strengthened when considering $P^n_{VW}(C,k)$. Secondly, based on the definition of $F_1, \ldots, F_5$ we can devise which variables may be considered to obtain strengthened versions of the TDCI for the general polytope P^n_{VW}. This topic is addressed in detail in Section 3.2.

We now establish the following results. Due to space limitations, the complete proof of Theorem 2 is shown in Appendix A.

Theorem 2. *Let $C = \langle v_1, v_2, \ldots, v_l, v_{l+1} = v_1 \rangle$, $4 \leqslant l < n$ and k the set of vertices and the starting time period associated with a TDCI as defined in Proposition 1. The dimension of $P^n_{VW}(C,k)$ is $n(n-1)(n-2) - (l+1)(l-2)$ for $n \geqslant 5$.*

Based on this theorem, we now state the following result. Similarly to Theorem 2, the detailed proof is shown in the appendix.

Theorem 3. *Let $C = \langle v_1, v_2, \ldots, v_l, v_{l+1} = v_1 \rangle$, $4 \leqslant l < n$ and k as defined in Proposition 1. The TDCI (17) are facet-defining for $P^n_{VW}(C,k)$ for $n \geqslant 5$.*

3.1.1. Separation of time-dependent cycle inequalities

In this section we study the complexity of the separation problem associated with TDCI inequalities. We consider the reformulation of the TDTSP as the problem of finding the shortest constrained path in a directed multi-partite graph, as suggested by Picard and Queyranne (1978) and Vander Wiel and Sahinidis (1995). Let $D_M = (V_M, A_M)$ be this graph. A vertex $u \in V_M$ is a combination between a vertex $i \in V$ and a time period k, where $0 \leqslant k \leqslant n$. We will denote this vertex as $u = ik$. In addition, we will refer to arcs $e \in A_M$ as $e = (ik, jk+1)$, with $ik, jk+1 \in V_M$.

The separation problem for this family of inequalities can be formulated as follows:

	TIME-DEPENDENT CYCLE INEQUALITIES SEPARATION
Instance:	A point $\tilde{z} = (\tilde{y}, \tilde{x}) \in P^n_{VW}$
Question:	Does $\tilde{z}$ violate some Time-Dependent Cycle Inequality?.

We begin showing that this family can be separated considering the support graph, $D_M(\tilde{z})$, instead of the complete graph. The following proposition proves this result.

Proposition 2. *Let $\tilde{z} = (\tilde{y}, \tilde{x})$ be a solution of the relaxation that satisfies (8)–(15) and $\tilde{z} \geqslant 0$, $C = \{v_1, \ldots, v_l\} \subset V$ and k the set of nodes and the starting time period associated with a TDCI, and $i_0 \in \{1, \ldots, l\}$. If $\tilde{y}_{v_{i_0} v_{i_0+1} k+i_0-1} = 0$, then the TDCI induced by C and k is trivially satisfied.*

Proof. From Proposition 1, the TDCI associated with C and k, can be rewritten considering v_{i_0} as

$$\underbrace{\left(\sum_{i=1}^{i_0-1} \tilde{y}_{v_i v_{i+1} k+i-1} - \sum_{i=1}^{i_0-1} \tilde{x}_{v_{i+1} k+i}\right)}_{A} + \tilde{y}_{v_{i_0} v_{i_0+1} k+i_0-1} + \underbrace{\left(\sum_{i=i_0+1}^{l} \tilde{y}_{v_i v_{i+1} k+i-1} - \sum_{i=i_0}^{l-1} \tilde{x}_{v_{i+1} k+i}\right)}_{B} \leqslant 0.$$

It is easy to see that $A \leqslant 0$ and $B \leqslant 0$, since solution $\tilde{z}$ satisfies Eqs. (11) and (13), respectively. Then, knowing that $\tilde{y}_{v_{i_0} v_{i_0+1} k+i_0-1} = 0$, the inequality is satisfied by $\tilde{z}$. □

Based on this result, we first identify some necessary conditions for the existence of a violated TDCI. As we have seen before, the main idea behind TDCI is the presence of a time-dependent cycle, starting and ending in the same node but with different time-periods. Then, there must exist vertices $v\,k$, $v\,k' \in V_M$, with $k' = k + l$, for some $l > 1$, $k + l \leqslant n$. In addition, even when this condition holds, it is also necessary the existence of a simple path-in terms of vertices $v_i \in V$-going from vk to vk', which is not always the case. Let v_1k, $v_1k' \in V_M$ be those vertices as defined before. We want to determine if there exists or not a set of vertices $C = \{v_2, \ldots, v_l\} \subset V$ such that

$$\sum_{i=1}^{l} \tilde{y}_{v_i v_{i+1} k+i-1} - \sum_{i=1}^{l-1} \tilde{x}_{v_{i+1} k+i} > 0, \tag{20}$$

where $v_{l+1} = v_1$. We now restrict our search to the subgraph defined by those vertices in the support graph with time periods between k and k', i.e., $u = vt$ such that $v \in V$ and $k < t < k'$. For this subgraph, we also define the arc weight function $w : A_M \to \mathbb{R}$ as

$$w(it, jt+1) = \begin{cases} \tilde{y}_{v_i v_j t} & \text{if } i = 1, t = k \\ \tilde{y}_{v_i v_j t} - \tilde{x}_{v_i t} & \text{otherwise.} \end{cases}$$

Considering these two definitions, we seek for a maximum-weight path connecting v_1k and v_1k'. This can be done in polynomial time with a straightforward application of dynamic programming.

However, the optimal solution returned by the maximum-weight path algorithm may not be simple in terms of vertices v from the original graph (i.e., it is possible that the path contains vertices v_it, $v_it' \in V_M$, with $t \neq t'$, visiting more than one time the row corresponding to vertex v_i in different time periods). Let P be this optimal solution, and w^* the weight of P. Clearly, if $w^* \leqslant 0$, then $\tilde{z}$ does not violate any TDCI starting and ending in v_1k and v_1k', respectively, since w^* is an upper bound for the value of any simple path connecting these two vertices. On the contrary, if $w^* > 0$, there might exist a TDCI that is violated by $\tilde{z}$. The following proposition proves that if this condition on w^* holds, then we can separate $\tilde{z}$.

Proposition 3. *Let $\tilde{z} = (\tilde{y}, \tilde{x})$ be a solution of the relaxation that satisfies (8)–(15) and $\tilde{z} \geqslant 0$, and $P = \langle v_1, \ldots, v_l \rangle$, $v_{l+1} = v_1$, the sequence of nodes returned by the maximum path algorithm with optimum value $w^* > 0$. Then, there exists a subsequence $P_0 \subseteq P$ and a time period k_0 such that the TDCI defined by P_0' and k_0 is violated by $\tilde{z}$.*

Proof. We start by pointing out that if sequence P has no repetitions, then clearly P and k define a violated TDCI, since $w^* > 0$.

Otherwise, P has at least one repeated vertex and each vertex may appear even more than two times. We also know that these repetitions cannot be in consecutive time periods, since this situation is not allowed even in graph D_M. However, it is not difficult to see that there exists a repeated vertex $v_{i_0} = v_{i_1} \in P$, $i_0 < i_1$, appearing in time periods $k_0 = k + i_0 - 1$ and $k_0' = k + i_1 - 1$, respectively, such that the vertices of P between two of its repetitions defines a subsequence of P which is a simple path connecting $v_{i_0}k_0$ and $v_{i_0}k_0'$. Let $P_0 = \{v_{i_0}, \ldots, v_{i_1-1}\} \subset P$ be this sequence, where $|P_0| \geqslant 3$. We now analyze the TDCI defined by P and k in terms of subsequence P_0 and k_0. Expression (20) can be rewritten as

$$\sum_{i=1}^{i_0-1} \tilde{y}_{v_i v_{i+1} k+i-1} - \sum_{i=1}^{i_0-1} \tilde{x}_{v_{i+1} k+i} + \tag{21}$$

$$\sum_{i=i_0}^{i_1-1} \tilde{y}_{v_i v_{i+1} k+i-1} - \sum_{i=i_0}^{i_1-2} \tilde{x}_{v_{i+1} k+i} + \tag{22}$$

$$\sum_{i=i_1}^{l} \tilde{y}_{v_i v_{i+1} k+i-1} - \sum_{i=i_1-1}^{l-1} \tilde{x}_{v_{i+1} k+i} > 0, \tag{23}$$

where (22) is the expression related to P_0. We consider each part separately:

1. The value of expression (21) is at most zero, since solution $\tilde{z}$ satisfies Eq. (11).
2. Analogously, expression (23) is also upper bounded by zero, given that $\tilde{z}$ satisfies (13).
3. From the previous two items, it follows that (22) is strictly greater than zero. In addition to this fact, by construction, P_0 has no repeated vertices. Moreover, if we consider $k_0 = k + i_0 - 1$, expression (22) defines a TDCI for P_0 and k_0. Then, we have found a sequence of vertices P_0 and time period k_0 that represent a TDCI violated by $\tilde{z}$.

To sum up, given the sequence of nodes P for whose optimal value w^* is strictly greater than zero, we are able to find a TDCI that is violated by $\tilde{z}$. □

Finally, considering all we have seen so far, we can state that the separation problem for the TDCI can be solved in polynomial time. A pseudocode for the separation procedure summarizing all steps from this section is shown in Algorithm 1. The complexity in the worst case is $O(n^5)$, although in practice it requires little effort since the support graph $D_M(\tilde{z})$ tends to be sparse and many cases can be avoided.

Algorithm 1. TDCI SEPARATION ALGORITHM

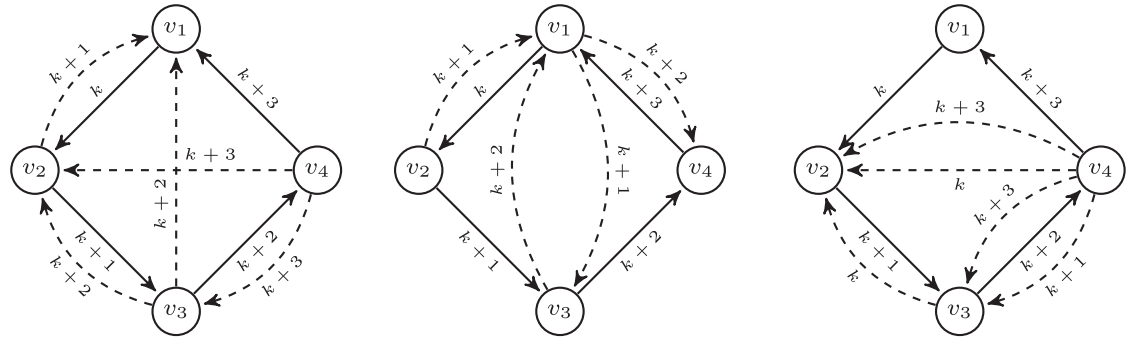
Input: $\tilde{z} = (\tilde{y}, \tilde{x}) \in P_{VW}^n$.
1. for $ik, ik' \in D_M(\tilde{z})$, $k' > k + 1$, **do:**
2. Calculate the maximum path on $D_M(\tilde{z})$ considering $w : A_M \to \mathbb{R}$ as the arc weight function. Let P be the optimal solution and w^* its cost.
3. **If** $w^* > 0$ **then**
4. If P is simple, then answer *yes* and return the TDCI induced by P.
5. **else**
6. Determine the simple subpath $P_0 \subseteq P$ as shown in Proposition 3 answer *yes* and return the TDCI induced by P_0.
7. **end if**
8. end for

3.2. Lifted time-dependent cycle inequalities

Based on the ideas from Balas and Fischetti (1999) and Gutin and Punnen (2002), from Proposition 3 we can derive facets of P_{PQ}^n and P_{VW}^n applying a maximum sequential lifting over the variables present in $F_1, \ldots, F_5$. It is well known that the order in which variables are lifted may generate different inequalities. Indeed, we lifted these variables in five different ways to obtain five families of facets. The support graph for some of these inequalities with $l = 4$ is shown in Fig. 2. In all cases, a solid arc corresponds to a variable in the original TDCI, and a dashed arc represents a variable obtained by the lifting process.

The following proposition shows the five different families of facets. The proof is provided in Appendix.

Proposition 4. *Let $C = \langle v_1, v_2, \ldots, v_l, v_{l+1} = v_1 \rangle$, $l < n$, be a simple cycle with transitions between consecutive vertices taken in time intervals k, $k + 1$, …, $k + l - 1$, $k + l \leqslant n$. Then, inequalities*



(a) Support for Lifted TDCI (24). (b) Support for Lifted TDCI (25). (c) Support for Lifted TDCI (26).

Fig. 2. Support for Lifted TDCI inequalities.

$$\sum_{i=1}^{l} y_{v_i v_{i+1} k+i-1} + \sum_{i=2}^{l-1}\sum_{j=1}^{i-1} y_{v_i v_j k+i-1} + \sum_{j=2}^{l-1} y_{v_l v_j k+l-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{24}$$

$$\sum_{i=1}^{l} y_{v_i v_{i+1} k+i-1} + \sum_{j=3}^{l} y_{v_1 v_j k+j-2} + \sum_{j=2}^{l-1} y_{v_j v_1 k+j-1} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{25}$$

$$\sum_{i=1}^{l} y_{v_i v_{i+1} k+i-1} + \sum_{i=3}^{l-1}\sum_{j=2}^{i-1} y_{v_i v_j k+j-2} + \sum_{j=2}^{l-1} y_{v_l v_j k+l-1} + \sum_{j=2}^{l-1} y_{v_l v_j k+j-2} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{26}$$

$$\sum_{i=1}^{l} y_{v_i v_{i+1} k+i-1} + \sum_{i=3}^{l-1}\sum_{j=2}^{i-1} y_{v_i v_j k+j-2} + \sum_{j=3}^{l} y_{v_1 v_j k+j-2} + \sum_{j=2}^{l-1} y_{v_l v_j k+j-2} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{27}$$

$$\sum_{i=1}^{l} y_{v_i v_{i+1} k+i-1} + \sum_{i=3}^{l-1}\sum_{j=2}^{i-1} y_{v_i v_j k+j-2} + \sum_{j=3}^{l} y_{v_1 v_j k+j-2} + y_{v_{l-1} v_1 k+l-2} \leqslant \sum_{i=1}^{l-1} x_{v_{i+1} k+i} \tag{28}$$

define facets of P_{PQ}^n and P_{VW}^n.

This proposition proves that constraints (24)–(28) are facet defining. We next remark that these five lifting of a TDCI indeed define different facets of P_{PQ}^n and P_{VW}^n. For this purpose, we use feasible solutions satisfying them by equality. Let $C = \langle v_1, v_2, v_3, v_4 \rangle$ be a time dependent cycle. We consider feasible solutions with the following permutations of the vertices.

- $v_4 v_1 v_3 v_2$ satisfies by equality (24), (25), (27) and (28), and strictly (26). Then, constraint (26) is different from the other Lifted TDCIs.
- $v_4 v_2 v_1 v_3$ satisfies by equality (24)–(27), and strictly (28). Then, constraint (28) is different from (24), (25) and (27).
- $v_3 v_2 v_4 v_1$ satisfies by equality (26)–(28), and strictly (24) and (25). Then, constraint (27) is different from (24) and (25).
- $v_3 v_1 v_5 v_4 v_2$ satisfies by equality (24) and (26), and strictly (25), (27) and (28). Then, constraint (24) is different from (25).

These five inequalities can be written as the Admissible Flow Constraints (AFC) proposed in Abeledo et al. (2010). In this paper, the authors conjecture that some of the AFC are facet defining, but that it is difficult to characterize which ones have this property. In Proposition 4 we characterize a subset of the AFC that are facet defining.[1]

3.2.1. Separation procedure

In order to incorporate these inequalities to the B&C algorithm, we use a heuristic separation routine based on the separation algorithm for the TDCI. For every two vertices of the form ik and ik' that are present in the support graph, we execute the maximum path algorithm. If the returned path is simple (in terms of vertices in V of the original graph D), then we add the value of the lifted variables involved in the inequality to the cost of this path.

Although this approach is heuristic, in practice it is quite effective for all families, finding a considerable number of violated cuts.

3.3. Polynomial size families

In this section we present four families of valid inequalities for P_{PQ}^n and P_{VW}^n. The following proposition introduces the first of them. The idea behind this family is to bound a particular variable, $y_{i_0 j_0 k_0}$ based on the values of the variables representing arcs leaving a particular vertex, l_0, with $l_0 \neq i_0 j_0$. The remaining constraints follow a similar idea.

Proposition 5 (*Family 1*). *For* i_0, j_0, $l_0 = 1, \ldots, n$: $i_0 \neq j_0 \neq l_0$, *inequalities*

$$y_{i_0 j_0 k_0} + y_{j_0 l_0 n-1} + y_{i_0 l_0 n-1} \leqslant y_{l_0 i_0 k_0 - 1} + \sum_{\substack{t=1 \\ t \neq k_0-1, k_0, k_0+1}}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w t} + y_{l_0 0 n} \tag{29}$$

for $k_0 = 2, \ldots, n-3$, *and*

$$y_{i_0 j_0 n-2} + \sum_{\substack{w=1 \\ w \neq j_0, l_0}}^{n} y_{w l_0 n-1} \leqslant y_{l_0 i_0 n-3} + \sum_{t=1}^{n-4} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w t} + y_{l_0 0 n} \tag{30}$$

$$y_{i_0 j_0 n-1} \leqslant y_{l_0 i_0 n-2} + \sum_{t=1}^{n-3} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w t} \tag{31}$$

are valid for P_{PQ}^n and P_{VW}^n.

[1] The results in Abeledo et al. (2010) have been obtained independently and simultaneously from ours.

Proof. We start by noting an identity that will be used several times within the proof. If we focus in the case, where $y_{i_0 j_0 k_0} = 1$, and considering Eqs. (3)–(6), it is quite easy to see that for $1 \leqslant k \leqslant n-1$

$$y_{l_0 i_0 k-1} + \sum_{\substack{w=1 \\ w \neq i_0,\ j_0, l_0}}^{n} \sum_{\substack{t=1 \\ t \neq k-1,\ k, k+1}}^{n-1} y_{l_0 w t} + y_{l_0 0 n} = y_{0 l_0 0} + \sum_{\substack{w=1 \\ w \neq l_0}}^{n} \sum_{t=1}^{n-1} y_{w l_0 t} = 1. \tag{32}$$

Now we prove the validity of each inequality separately.

1. For inequality (29), we consider three different cases:
   - If $y_{i_0 j_0 k_0} = y_{j_0 l_0 n-1} = y_{i_0 l_0 n-1} = 0$, the inequality is trivially satisfied since all variables are non-negative.
   - If $y_{i_0 j_0 k_0} = 1$, it follows that $y_{j_0 l_0 n-1}$ and $y_{i_0 l_0 n-1}$ have value zero since $k_0 \leqslant n-3$. Then, by (32) the inequality is satisfied.
   - If either $y_{j_0 l_0 n-1} = 1$ or $y_{i_0 l_0 n-1} = 1$, variable $y_{i_0 j_0 k_0} = 0$ and the inequality is satisfied since by Eq. (5) $y_{l_0 0 n} = 1$.

   These three cases cover all possible situations, and therefore inequality (29) is valid for P^n_{PQ} and P^n_{VW}.
2. For inequality (30), we separate again the proof in three cases:
   - If $y_{i_0 j_0 n-2}$ and $y_{w l_0 n-1}$ are zero, for $w = 1, \ldots, n$, $w \neq l_0, j_0$, then the inequality is trivially satisfied.
   - If $y_{i_0 j_0 n-2} = 1$, $y_{w l_0 n-1} = 0$ for $w = 1, \ldots, n$, $w \neq l_0, j_0$. Using Eq. (32) for $k = n-2$, we get the expression in the right hand side of the inequality, and therefore it is satisfied.
   - If for some $w = 1, \ldots, n$, $w \neq l_0, j_0$, $y_{w l_0 n-1} = 1$, it follows that $y_{i_0 j_0 n-2} = 0$ and from Eq. (5), $y_{l_0 0 n} = 1$ and the inequality is satisfied too.

   Therefore, the inequality is valid from P^n_{PQ} and P^n_{VW}.
3. Finally, we consider inequality (31). If $y_{i_0 j_0 n-1} = 0$, the inequality is satisfied. Otherwise, considering Eq. (32) and the fact $y_{i_0 j_0 n-1} = 1$ implies that $y_{l_0 0 n} = 0$, the inequality is also valid for P^n_{PQ} and P^n_{VW}.

Therefore, inequalities (29)–(31) are valid for P^n_{PQ} and P^n_{VW}. □

The next family uses mainly the same idea as in the previous proposition, with the only difference that instead of looking at the arcs leaving l_0, we consider the entering ones. Regarding the proof for this family, since it is analogous to the one in Proposition 5, we omit the details.

Proposition 6 (*Family 2*). *For i_0, j_0, $l_0 = 1, \ldots, n$: $i_0 \neq j_0 \neq l_0$, inequalities*

$$y_{i_0 j_0 k_0} + y_{l_0 i_0 1} + y_{l_0 j_0 1} \leqslant y_{j_0 l_0 k_0+1} + \sum_{\substack{t=1 \\ t \neq k_0-1, k_0, k_0+1}}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} + y_{0 l_0 0}$$

for $k_0 = 3, \ldots, n-2$, and

$$y_{i_0 j_0 2} + \sum_{\substack{w=1 \\ w \neq i_0, l_0}}^{n} y_{l_0 w 1} \leqslant y_{j_0 l_0 3} + \sum_{t=4}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} + y_{0 l_0 0}$$

$$y_{i_0 j_0 1} \leqslant y_{j_0 l_0 2} + \sum_{t=3}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t}$$

are valid for P^n_{PQ} and P^n_{VW}.

By combining inequalities from Propositions 5 and 6 in a particular way, we derive the two other families. Next we introduce the inequalities resulting of the combination of inequalities from Proposition 6.

Proposition 7 (*Family 3*). *For i_0, j_0, $l_0 = 1, \ldots, n$: $i_0 \neq j_0 \neq l_0$, inequalities*

$$y_{i_0 j_0 k_0} + y_{j_0 i_0 k_0} + y_{l_0 i_0 1} + y_{l_0 j_0 1} \leqslant y_{i_0 l_0 k_0+1} + y_{j_0 l_0 k_0+1} + \sum_{\substack{t=1 \\ t \neq k_0-1, k_0, k_0+1}}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} + y_{0 l_0 0} \tag{33}$$

for $k_0 = 3, \ldots, n-2$, and

$$y_{i_0 j_0 2} + y_{j_0 i_0 2} + \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w 1} \leqslant y_{i_0 l_0 3} + y_{j_0 l_0 3} + \sum_{t=4}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} + y_{0 l_0 0} \tag{34}$$

$$y_{i_0 j_0 1} + y_{j_0 i_0 1} \leqslant y_{i_0 l_0 2} + y_{j_0 l_0 2} + \sum_{t=3}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} \tag{35}$$

are valid for P^n_{PQ} and P^n_{VW}.

Proof. We prove first the validity of inequality (33). From the previous proposition, we know that

$$y_{i_0 j_0 k_0} + y_{l_0 i_0 1} + y_{l_0 j_0 1} \leqslant y_{j_0 l_0 k_0+1} + \sum_{\substack{t=1 \\ t \neq k_0-1, k_0, k_0+1}}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} + y_{0 l_0 0}$$

$$y_{j_0 i_0 k_0} + y_{l_0 i_0 1} + y_{l_0 j_0 1} \leqslant y_{i_0 l_0 k_0+1} + \sum_{\substack{t=1 \\ t \neq k_0-1, k_0, k_0+1}}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 t} + y_{0 l_0 0}$$

are valid inequalities. Given that $y_{i_0 j_0 k_0}$ and $y_{j_0 i_0 k_0}$ are mutually exclusive, the proposed inequality is valid for P^n_{PQ} and P^n_{VW}. The same argument can be used for inequalities (34) and (35). However, the former is slightly modified by excluding both $y_{l_0 i_0 1}$ and $y_{l_0 j_0 1}$ from the sum on the left side of the expression. □

Finally, the remaining family exploits the same idea as Proposition 7 but for inequalities present in Proposition 5. Again, due to the similarity with the previous result, the details of the proof are not provided.

Proposition 8 (*Family 4*). *For i_0, j_0, $l_0 = 1, \ldots, n$: $i_0 \neq j_0 \neq l_0$, inequalities*

$$y_{i_0 j_0 k_0} + y_{j_0 i_0 k_0} + y_{i_0 l_0 n-1} + y_{j_0 l_0 n-1} \leqslant y_{l_0 i_0 k_0-1} + y_{l_0 j_0 k_0-1} + \sum_{\substack{t=1 \\ t \neq k_0-1, k_0, k_0+1}}^{n-1} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w t} + y_{l_0 0 n}$$

for $k_0 = 2, \ldots, n-3$, and

$$y_{i_0 j_0 n-2} + y_{j_0 i_0 n-2} + \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{w l_0 n-1} \leqslant y_{l_0 i_0 n-3} + y_{l_0 j_0 n-3} + \sum_{t=1}^{n-4} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w t} + y_{l_0 0 n}$$

$$y_{i_0 j_0 n-1} + y_{j_0 i_0 n-1} \leqslant y_{l_0 i_0 n-2} + y_{l_0 j_0 n-2} + \sum_{t=1}^{n-3} \sum_{\substack{w=1 \\ w \neq i_0, j_0, l_0}}^{n} y_{l_0 w t}$$

are valid for P^n_{PQ} and P^n_{VW}.

Regarding the separation problems for these families, since each of them is composed by a polynomial number of members, we explicitly enumerate them in order to find a violated cut.

4. B&C algorithm

In order to evaluate the strength of the inequalities introduced in Section 3, we develop a B&C algorithm considering the model (PQ). We focus mainly on the cutting planes and a primal heuristic leaving, among other parameters, the branching and the node selection strategies as CPLEX's defaults. For notational convenience, sometimes we refer to variables x_{ik} from (VW) formulation, the value of which can be calculated in terms of variables y_{ijk}.

We observed that in some instances applying a simple preprocessing improved drastically the overall computing times. This preprocessing phase aims to eliminate feasible solutions for which we can assure that either they are not optimal or there exists an alternative optimal solution.

The main idea is quite simple and looks for identifying time dependent arcs which cannot be present in an optimal solution. Given $i, j \in V\backslash\{0\}$, $i < j$, $1 \leqslant k \leqslant n-2$, if for all $v, w \in V$, $v, w \neq i, j$, if

$$c_{vik-1} + c_{ijk} + c_{jwk+1} \leqslant c_{vjk-1} + c_{jik} + c_{iwk+1}, \tag{36}$$

then we can fix variable $y_{jik} = 0$. Otherwise, we check whether swapping i and j the condition is satisfied to fix $y_{ijk} = 0$. It is important to remark the second test is applied only when the first one fails, since otherwise the optimal solution may be cut off.

For some particular cases of the TDTSP, such as the TSP and the TDP, this test can be executed without considering the particular position of the arc in the tour and, therefore, the implication is valid for all possible time periods k.

4.1. Primal heuristic

The use of heuristic procedures to obtain feasible integer solutions based on the information provided by the solution of the LP relaxation have been proven to be very effective to obtain good quality upper bounds. The main purpose is to reduce the number of nodes explored in the B&C tree, aiming to reduce the overall running time of the algorithm. However, it is important to find a reasonable tradeoff between the effectiveness of the algorithm and the computational effort.

In our algorithm we consider a primal heuristic consisting of two phases. The first one is a construction phase, which generates different feasible solutions based on a heuristic. The second phase consists on applying for each solution an improvement procedure. We give further details about each phase in the following sections.

4.1.1. Construction phase

The greedy heuristic considered in this phase iteratively adds vertices to positions 0, 1, …, $n-1$ using the information of the fractional solution of the current node. In particular, in iteration k selects as the next vertex v to be added to the partial tour the one having the greatest value its corresponding assignment variable $y_{v_{\text{last}}vk}$, where v_{last} is the last vertex added to the current partial path. In case of tie, we select the vertex with minimum cost for the corresponding arc.

In order to generate several possibilities, we execute this procedure considering each vertex $w \in V$ as the first one in the tour.

4.1.2. Improvement phase

The aim of this phase is, given a feasible solution for the problem, to find a new one with a better total cost by applying a local search procedure. We consider two different improvement operators, which are executed sequentially. It is important to remark that this improvement phase is applied to all solutions generated in the construction phase.

First we consider a vertex interchange heuristic. Given $C = \langle 0, v_1, \ldots, v_n, 0\rangle$ a feasible tour, we define the neighborhood of C, $N(C)$, as all the possible tours obtained by interchanging the positions of any two different vertices of C. From all of the tours in $N(C)$ that improve the cost of C, in case there is any, we choose as our new solution the one with the smallest cost. We apply this procedure iteratively until no improvement is achieved or a maximum of fifteen iterations is reached. Secondly, when this procedure is finished, we execute a 3-opt procedure (Lin, 1965).

4.2. Cutting planes

In this section we specify the cutting plane algorithm considered. Based on restricted preliminary computational results, considering together inequalities (24)–(28) slows the resolution of the LP relaxations. This is due to their similarity, given that some of them share many of the variables involved in the inequality. As a consequence, more inequalities are added to the formulation but obtaining similar results in terms of improvements of the lower bound. We observed that the best results are produced considering together constraints (24) and (26).

In addition to the inequalities presented in Section 3 we included also the *Subtour Elimination Constraints* (SEC). In order to include these constraints, variables z_{ij} can be defined as

$$z_{ij} = \sum_{k=1}^{n} y_{ijk},$$

indicating if vertex j is visited immediately after vertex i in the tour.[2] We consider adding more than one of these constraints instead of only the most violated one by means of the separation routine proposed by Letchford et al. (2002) and Lysgaard et al. (2004). At the root node we perform a maximum of fifteen rounds of the cutting plane algorithm and only one round in the internal nodes.

On preliminary computational results, we observed that the best improvements at the root node is produced by a combination of the SEC, the TDCI with $l = 2$ and the Lifted TDCI (24) and (26). For example, in some instances considering only SEC or TDCI with $l = 2$, constraints (24) and (26) the gap at the root node is reduced to nearly an 8% in both cases. However, when considered together, this value drops below 2%. For this reason, we include the three sets of constraints in the cutting plane algorithm. As regards inequalities presented in Section 3.3, although they do not produce big improvements in the objective function, they are useful to find violated Lifted TDCI in successive rounds.

The constraints considered in the cutting plane algorithm are enumerated below.

- SEC. At most 30 per round.
- TDCI with $l = 2$.
- Lifted TDCI (24) and (26). At most 100 per round.
- Families from Section 3.3. At most 100 for each of them per round.

5. Computational results

The experiments were conducted on a workstation with Intel Core i7-2600 with 16 gigabytes of RAM running a Fedora Linux dis-

[2] To avoid confusions with variables x_{ik} from model (VW), we rename variables x_{ij} from the classical formulation for the TSP as z_{ij}. It is important to remark that variables z_{ij} are not included explicitly in the formulation (PQ).

Table 1
Computational times (in seconds) and number of tree nodes explored for TSP instances from TSPLIB.

Type	Instance	n	B&C				CPLEX-PQ			
			Time	Nodes	%rG	%fG	Time	Nodes	%rG	%fG
TSP	burma14	14	0.02	0	0.00	0.00	0.26	125	7.45	0.00
	ulysses16	16	0.07	0	0.00	0.00	0.83	364	10.95	0.00
	gr17	17	0.09	0	0.00	0.00	19.86	20,001	13.25	0.00
	gr21	21	0.09	0	0.00	0.00	8.58	2040	6.75	0.00
	ulysses22	22	0.37	0	0.00	0.00	26.27	6128	16.05	0.00
	gr24	24	0.47	0	0.00	0.00	25.98	3496	10.68	0.00
	fri26	26	0.55	0	0.00	0.00	204.39	29,943	9.14	0.00
	bayg29	29	1.01	0	0.00	0.00	609.94	54,874	7.25	0.00
	bays29	29	1.83	0	0.00	0.00	1179.66	118,294	8.69	0.00
	dantzig42	42	10.88	0	0.00	0.00	***	***	12.50	7.68
	swiss42	42	6.94	0	0.00	0.00	***	***	17.92	12.28
	att48	48	76.29	35	0.17	0.00	***	***	16.70	19.57
	gr48	48	***	***	1.68	0.33	***	***	15.47	21.56
	hk48	48	24.51	0	0.00	0.00	***	***	11.10	14.21
	eil51	51	1306.8	84	0.82	0.00	***	***	9.91	32.64
	berlin52	52	26.1	0	0.00	0.00	***	***	13.15	32.39
	brazil58	58	220.33	25	0.11	0.00	***	***	25.73	81.74
	br17	17	0.03	0	0.00	0.00	0.56	132	16.03	0.00
	ftv33	33	1.81	0	0.00	0.00	646.18	18,158	6.80	0.00
	ftv35	35	30.99	39	0.82	0.00	269.58	11,757	5.44	0.00
	ftv38	38	59.96	55	0.80	0.00	788.7	14,998	5.28	0.00
	p43	43	5080.08	63	0.14	0.00	***	***	84.13	81.84
	ftv44	44	347.23	35	1.43	0.00	5383.61	48,305	5.02	0.00
	ftv47	47	528.55	73	1.50	0.00	***	***	3.63	5.30
	ry48p	48	521.12	95	0.85	0.00	***	***	10.72	9.03
	ftv55	55	3164.68	85	1.28	0.00	***	***	10.17	19.62

Table 2
Computational times (in seconds) and number of tree nodes explored for TDP instances from TSPLIB.

Type	Instance	n	B&C				B&C-R				CPLEX-PQ			
			Time	Nodes	%rG	%fG	Time	Nodes	%rG	%fG	Time	Nodes	%rG	%fG
TDP	burma14	14	0.03	0	0.00	0.00	0.53	0	0.00	0.00	0.58	220	8.70	0.00
	ulysses16	16	0.14	0	0.00	0.00	6.83	7	1.74	0.00	0.87	273	17.95	0.00
	gr17	17	0.1	0	0.00	0.00	1.68	0	0.00	0.00	1.34	380	15.16	0.00
	gr21	21	0.37	0	0.00	0.00	2.98	0	0.00	0.00	14.8	4123	16.29	0.00
	ulysses22	22	4.65	33	4.18	0.00	129.93	32	4.27	0.00	20.33	4648	27.43	0.00
	gr24	24	0.48	0	0.00	0.00	4.54	0	0.00	0.00	22.89	3026	14.68	0.00
	fri26	26	2.74	7	0.71	0.00	47.36	7	1.97	0.00	164.91	21,903	13.67	0.00
	bayg29	29	94.53	95	3.00	0.00	250.07	27	2.18	0.00	705.15	67,719	12.80	0.00
	bays29	29	14.36	39	1.86	0.00	618.08	49	1.62	0.00	345.76	31,839	13.77	0.00
	dantzig42	42	39.93	33	0.90	0.00	2278.71	41	2.30	0.00	***	***	19.10	11.33
	swiss42	42	30.33	13	1.20	0.00	1422.74	21	4.01	0.00	***	***	18.53	13.51
	att48	48	70.79	39	0.85	0.00	3377.31	33	4.30	0.00	***	***	16.12	7.89
	gr48	48	2014.95	407	3.99	0.00	***	***	6.84	8.32	***	***	20.66	17.70
	hk48	48	131.65	39	1.61	0.00	***	***	4.60	4.60	***	***	16.48	12.80
	eil51	51	90.55	25	0.71	0.00	***	***	1.94	4.14	***	***	14.75	19.86
	berlin52	52	***	***	3.54	1.01	***	***	13.45	14.30	***	***	21.83	23.71
	brazil58	58	***	***	8.77	2.96	***	***	17.70	19.15	***	***	21.83	36.87
	br17	17	0.04	0	0.00	0.00	295.08	171	12.45	0.00	0.67	16	7.32	0.00
	ftv33	33	22.68	27	1.57	0.00	887.52	47	2.40	0.00	807.67	41,585	11.62	0.00
	ftv35	35	23.57	19	1.34	0.00	778.47	38	1.59	0.00	1645.47	72,258	7.87	0.00
	ftv38	38	41.74	23	1.38	0.00	341.31	11	2.06	0.00	1426.38	36,485	8.68	0.00
	p43	43	3241.23	691	4.54	0.00	***	***	62.02	62.14	***	***	50.27	42.39
	ftv44	44	449.35	117	3.54	0.00	***	***	4.44	5.25	***	***	9.86	7.19
	ftv47	47	41.27	13	1.52	0.00	***	***	3.20	6.25	703.7	4600	6.44	0.00
	ry48p	48	39.96	7	0.52	0.00	2550.81	27	3.54	0.00	***	***	8.03	6.28
	ftv55	55	156.89	33	1.48	0.00	***	***	7.33	7.33	***	***	12.43	12.18

tribution. The algorithms are coded in C++and combined with Ilog CPLEX 12.2 callable library for the optimization routines.

We use benchmark instances that are divided into three groups. The first group includes instances from TSPLIB. These instances are considered both as TSP instances (i.e., $c_{ijk} = c_{ij}$) and as TDP instances (i.e., $c_{ijk} = (n - k + 1)c_{ij}$). A second group regards randomly generated instances for TDP from Méndez-Díaz et al. (2008). Finally, the third group considers the instances proposed in Rubin and Ragatz (1995) and considered also in Bigras et al. (2008). We slightly modify the original instances of the third group by discarding the corresponding due dates, which results in $1|s_{ij}|\sum C_j$ scheduling instances (equivalent to TDP).

As regards the methods evaluated, we consider the following ones:

- B&C: the B&C algorithm described in Section 4 considering the PQ model. All CPLEX cuts and heuristics are disabled.

Table 3
Average computational times (in seconds) and number of tree nodes explored for Méndez-Díaz et al. instances.

Instances	n	B&C				B&C-R				CPLEX-PQ			
		Time	Nodes	%rG	%fG	Time	Nodes	%rG	%fG	Time	Nodes	%rG	%fG
Asymmetric	20	0.05	0.00	0.00	0.00	0.68	0.00	0.00	0.00	0.20	0.00	0.00	0.00
	22	0.08	0.00	0.00	0.00	0.84	0.00	0.00	0.00	0.48	10.80	0.41	0.00
	24	0.13	0.00	0.00	0.00	2.38	0.60	0.15	0.00	0.75	21.20	2.01	0.00
	26	0.13	0.00	0.00	0.00	2.11	0.00	0.00	0.00	0.74	4.20	0.65	0.00
	28	0.44	0.60	0.08	0.00	14.54	6.40	0.47	0.00	2.35	25.80	1.70	0.00
	30	0.98	2.80	0.50	0.00	34.09	8.00	0.58	0.00	4.30	104.20	3.25	0.00
	35	0.99	0.60	0.12	0.00	21.24	0.80	0.10	0.00	4.28	7.20	0.55	0.00
	40	8.17	10.00	1.22	0.00	489.50	25.00	1.90	0.00	53.42	952.00	5.48	0.00
Symmetric	20	0.15	0.00	0.00	0.00	1.39	0.00	0.00	0.00	3.28	519.40	18.77	0.00
	22	0.28	0.00	0.00	0.00	2.56	0.00	0.00	0.00	8.08	1385.60	17.12	0.00
	24	0.47	0.00	0.00	0.00	7.05	1.00	0.11	0.00	23.94	2943.00	27.27	0.00
	26	0.58	0.00	0.00	0.00	6.82	1.00	0.23	0.00	63.33	6077.20	23.45	0.00
	28	1.00	0.00	0.00	0.00	12.21	0.60	0.04	0.00	113.60	9563.40	26.04	0.00
	30	1.46	0.00	0.00	0.00	24.94	1.00	0.22	0.00	466.21	32789.00	27.55	0.00
	35	3.67	0.00	0.00	0.00	57.52	0.60	0.07	0.00	(4) 1242.66	47,799	34.61	10.05
	40	9.58	2.00	0.19	0.00	391.15	11.00	1.35	0.00	(1) 6749.43	158,253	34.00	19.77
Euclidean	20	0.15	0.00	0.00	0.00	1.27	0.00	0.00	0.00	3.79	526.80	10.47	0.00
	22	0.44	0.00	0.00	0.00	6.75	2.80	0.51	0.00	9.90	1199.80	13.32	0.00
	24	0.78	0.00	0.00	0.00	10.44	2.00	0.60	0.00	34.84	3689.00	16.12	0.00
	26	1.19	1.00	0.13	0.00	19.31	4.40	0.26	0.00	61.93	5772.00	13.37	0.00
	28	5.02	10.80	0.69	0.00	37.17	5.60	0.67	0.00	233.05	17746.20	18.25	0.00
	30	4.01	2.60	0.22	0.00	133.51	14.20	0.64	0.00	234.90	12049.60	16.24	0.00
	35	8.53	3.20	0.32	0.00	139.52	6.20	0.53	0.00	(4) 3164.09	112825.75	20.05	7.74
	40	44.35	27.80	1.50	0.00	1421.73	33.80	1.87	0.00	(1) 4984.88	90,391	21.62	10.75

Table 4
Average computational times (in seconds) for scheduling instances.

Instances	n	B&C			CPLEX-PQ		
		Time	Nodes	%rG	Time	Nodes	%rG
PROB40x.TXT	15	0.02	0.00	0.00	0.23	2.88	0.04
PROB50x.TXT	25	0.15	0.00	0.00	1.06	21.75	0.06
PROB60x.TXT	35	1.68	4.75	0.02	9.24	80.00	0.08
PROB70x.TXT	45	10.35	7.00	0.02	104.56	923.63	0.09

- B&C-R: the B&C algorithm for the TDP proposed in Méndez-Díaz et al. (2008). This algorithm is based on a special formulation for the TDP, whose variables capture the cumulative nature of costs. It includes several facet defining inequalities as well as a primal heuristic, producing good and competitive computational results. Taking to account the benchmark instances tested, the results produced by this algorithm represents a good baseline for the evaluation of B&C. Since we have access to the code, the computational experiments for this algorithm are carried out in the same environment described before.
- CPLEX-PQ: CPLEX's default algorithm considering the PQ model. It also includes the variable fixing phase explained in Section 4 and all CPLEX's general features. The comparison with this algorithm will show the benefits obtained by including special purpose features in B&C.

For each algorithm we report the computational time (in seconds) and the number of tree nodes explored in the B&C algorithm. A cell filled with (***) means that the instance was not solved within 2 hours by that algorithm. We also show the gaps at the root node (%rG) and at the end of the execution (%fG). The %rG is calculated as $100 * (BESTUB - LB)/BESTUB$, where $BESTUB$ is the objective value of the best solution obtained considering all algorithms tested. For %fG we adopted a different criterion, and is computed as $100 * (UB - LB)/UB$, where UB stands for the objective value of the best solution found by the algorithm under consideration. In this way, %fG gives us a measure of the progress made by the algorithm when the time limit is reached. For the B&C-R algorithm we do not report results of TSP instances, since B&C-R is specifically developed for the TDP.

5.1. Comparison of B&C algorithms

In Tables 1 and 2 we show the computational times for the TSPLIB instances considered as TSP and TDP instances, respectively. One of the messages of this table is that our B&C algorithm outperforms CPLEX-PQ in all the instances considered, producing much better results and solving almost all instances considered. This lies on the fact that the inequalities incorporated to the cutting phase are quite effective, specially the combination of the TDCI of size 2, the Lifted TDCI and SECs. The best gains are obtained at the root node, where the gap with respect to the optimal solution is considerably reduced. The average of %rG over all TSP instances considered is 0.36% for B&C while for CPLEX-PQ is 13.84%. For TDP instances, these values are 1.82% for B&C and 15.85% for CPLEX-PQ. It is important to note that in both cases the %rG for TDP instances tend to be larger than for TSP ones. Furthermore, the number of instances solved at the root node by B&C is smaller when considered as TDP.

The most interesting results are the ones regarding instances with 40 vertices or more. CPLEX-PQ algorithm is able to solve to optimality only 2 of the 26 instances (ftv44 TSP; ftv47 TDP) within 2 hours, while our B&C solves 23 of them. This is also expressed in the number of nodes explored in the B&C tree, where CPLEX-PQ requires to visit an extremely higher number of them compared to B&C.

Considering B&C-R algorithm, it is able to solve 4 of the 13 TDP instances with more than 40 vertices, The computational times are considerable higher compared to the ones produced by B&C. This can be explained by the fact that B&C-R LP relaxations are difficult to solve and, even when they produce good lower bounds, the procedure of B&C is more effective.

We now turn our attention to two specific instances for which computational times are affected by the preprocessing explained

Table 5
Comparison of computational times (in seconds) with Bigras et al. [Bigras et al., 2008].

Instance	B& C		Bigras et al.		B& C-R		CPLEX-PQ	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
gr17	0.1	0	3	1	1.68	0	1.34	380
gr21	0.37	0	10	1	2.98	0	14.80	4123
gr24	0.48	0	15	1	4.54	0	22.89	3026
bays29	14.36	39	76	16	618.08	49	345.76	31,839
bayg29	94.53	95	191	51	250.07	27	705.15	67,719
rbg016a	0.03	0	15	1	1.01	0	0.2	0
rbg031a	1.29	0	90	1	279.37	5	21.31	290
rbg050b	620.46	602	***	***	***	***	1239.92	16,125
dTSP40.0	3571.54	1119	6473	377	***	***	***	***
dTSP40.1	224.94	95	1452	50	***	***	***	***
dTSP40.2	342.23	125	1068	28	5662.62	127	***	***
dTSP40.3	6410.05	1431	629	24	***	***	***	***
dTSP40.4	26.83	29	299	4	1184.86	30	***	***
dTSP50.0	363.57	91	1364	5	***	***	***	***
dTSP50.1	3088.25	499	***	***	***	***	***	***
dTSP50.2	594.8	205	3668	8	***	***	***	***
dTSP50.3	***	***	***	***	***	***	***	***
dTSP50.4	***	***	***	***	***	***	***	***

in Section 4. Instance br17, when considered as a TDP instance, without this variable fixing requires 209.24 s and 1549 nodes in the B&C tree to be solved. In Table 2 we can appreciate that it is solved in less than a second at the root node. A similar observation can be done for p43, which cannot be solved within the time limit imposed without the preprocessing, neither in its TSP version nor in its TDP one. Although it is a quite restrictive condition, it shows to be very effective and produces significant changes in the overall computational times.

Regarding the %fG, we can observe that B&C produces the best results in the instances that cannot be solved within the time limit imposed. For TSP instances, B&C is not able to solve only one instance and %fG is 0.33 %, while for CPLEX-PQ the average %fG over the unsolved instances is 28.15%. For TDP ones, the average %fG over the unsolved instances is 1.99% for B&C, 14.61% for B&C-R and 17.64% for CPLEX-PQ. Based on this values and the number of instances solved we can deduce that B&C represents a more robust approach than B&C-R and CPLEX-PQ. This is due to the combination between the improvements obtained by means of the cutting planes and the effectiveness of the primal heuristic in finding good feasible solutions early in the B&C tree.

The results for the instances from Méndez-Díaz et al. (2008) are presented in Table 3. Each row shows the average value for the computational time, number of nodes explored and gaps over five instances. If present, a number between parenthesis indicates how many of the five instances are actually solved to optimality within 2 hours and the value in the cell represents the average considered only these instances.

In general, results are similar to the ones for the TSPLIB instances. The B&C algorithm produces considerable reductions in the computational times, and these differences become larger as the number of vertices of the instances increase. Both B&C and B&C-R are capable of solving all instances, in general enumerating a small number of nodes in the B&C tree. CPLEX-PQ begins to fail in symmetric and euclidean instances with 35 and 40 vertices. The gaps at the root node for B&C-R are very good in general, but the algorithm is more time consuming than B&C because of the time required to solve each LP relaxation. As expected, CPLEX-PQ enumerates a great number of nodes in the B&C tree, showing the benefit of the cutting plane algorithm of B&C.

An interesting observation from the results observed in this table regards the difficulty of the instances depending on its type. Both B&C and B&C-R are able to solve asymmetric and symmetric instances within a reasonable time, and for each algorithm the times are comparable. However, we can observe an increment on the times as well as in the number of nodes explored for euclidean instances, which seem to be harder to solve than the ones mentioned previously.

In Table 4 we show the average computational times for the scheduling instances from Rubin and Ragatz (1995). The average value is calculated over eight instances for each $n = 15, 25, 35, 45$. These results are aligned with the ones from the previous tables. Our B&C performs better than CPLEX-PQ, both in the computational times and the number of nodes explored. It is important to note that this difference is significantly higher when $n = 45$, since the other instances are easily solved by both methods. Again, this behavior is due to the strengthening of the bounds produced by the cutting phase and the primal heuristic.

Finally, we compare our results with the ones reported in Bigras et al. (2008) in Table 5. As mentioned in the introduction, the authors study the path formulation proposed in Picard and Queyranne (1978). The formulation has an exponential number of variables and therefore they consider a B&C&P algorithm. For the pricing phase, they consider adding paths (columns) without four cycles. They also include several families of valid inequalities of the TSP and clique inequalities obtained by constructing the conflict graph using incompatibilities inferred for the TDTSP.

The authors report computational times in all cases, exceeding our time limit of 2 hours in four instances. It is important to remark that the computer used by Bigras et al. (2008) dates from 2008. Based on standard CPU comparisons[3] and considering that our code runs in single thread, our computer has a speed-up of nearly three compared to theirs. Thus, we report (***) for their computing times whenever they exceed $3 * 7200 = 21{,}600$ seconds. Besides this, we can observe some interesting results in the comparison.

First we note that B&C solves more instances than all the other methods. In some cases, computational times obtained by Bigras et al. are reduced by B&C to a 10% of the time. Furthermore, computational times reported by the authors for instances dTSP50.1, dTSP50.3 and dTSP50.4 are 56,240, 47,771 and 109,586 seconds, respectively. Regarding B&C, it can solve dTSP50.1 in less than an hour. It is also interesting the reduction obtained for instance rbg050b, where B&C requires approximately 10 minutes to solve it. Despite the difference on the computers, the reductions on the

[3] Obtained from www.cpubenchmark.net. Bigras et al. CPU obtains a rating of 744, and ours 1922.

computational times are also due to the inclusion of specific purpose cuts and the primal heuristic. On the other hand, for the instance dTSP40.3 their algorithm performs much better than B&C, which is able to find the optimal solution but the initial gap is large and encounters difficulties to close it. This is related to the nature of their approach, which can also be observed in the fact that their B&C&P algorithm tends to enumerate less nodes than B&C. As regards B&C-R and CPLEX-PQ, the behavior is similar to the previous experiments.

6. Conclusions

In this paper we consider the TDTSP formulations of Picard and Queyranne (1978) and Vander Wiel and Sahinidis (1996). We analyze both polytopes and derive several families of valid inequalities for both models. We generalize the idea of the well-known cycle inequalities for the ATSP, and derive five families of facets by applying a lifting procedure. We develop a B&C algorithm in order to evaluate these inequalities which, together with a primal heuristic, prove to be very effective. The overall approach produces good computational results over different benchmark instances compared to a general purpose B&C algorithm for the model proposed in Picard and Queyranne (1978), that inclues also a particular preprocessing for the problem and is solved with CPLEX default algorithm, and the B&C algorithm from Méndez-Díaz et al. (2008). The proposed algorithm is capable of solving instances with up to 58 vertices within 2 hours of computing time.

As future research, it would be interesting to analyze the complexity of the separation problems for the lifted TDCI in order to improve the separation routines implemented so far. Regarding the polyhedral study performed in this paper, although we do not have a proof, we conjecture that valid inequalities proposed in Section 3.3 are facet defining. More work could be done in this direction.

Investigating how to improve the computational times required to solve the LP relaxations of B&C, by means of studying particular characteristics of the formulation (PQ), could lead to significant reductions in the overall computational times, specially for instances with large values of n. In addition, as mentioned in the computational experiments, the preprocessing proposed showed to be very effective on instances satisfying condition (36). It is worth to investigate further this aspect of the problem to derive new rules to fix variables in the model.

Acknowledgements

This work was partly funded by grants PICT 2010-304, PICT 2011-817 and UBACyT 20020100100666. The authors are grateful to two anonymous referees for their valuable comments.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ejor.2013.05.022.

References

Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E., 2010. The time dependent traveling salesman problem: polyhedra and branch-cut-and-price algorithm. In: Festa, P. (Ed.), Experimental Algorithms, Lecture Notes in Computer Science, vol. 6049. Springer, Berlin/Heidelberg, pp. 202–213.

Balas, E., Fischetti, M., 1999. Lifted cycle inequalities for the asymmetric traveling salesman problem. Mathematics of Operations Research 24, 273–292.

Balas, E., Oosten, M., 1998. On the dimension of projected polyhedra. Discrete Applied Mathematics 87, 1–9.

Bigras, L., Gamache, M., Savard, G., 2008. The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discrete Optimization 5, 685–699.

Fischetti, M., Laporte, G., Martello, S., 1993. The delivery man problem and cumulative matroids. Operations Research 41, 1055–1064.

Fox, K. (1973). *Production scheduling on parallel lines with dependencies*. Ph.D. Thesis, John Hopkins University.

Fox, K.R., Gavish, B., Graves, S.C., 1980. An n-constraint formulation of the (time-dependent) traveling salesman problem. Operations Research 28, 1018–1021.

Godinho, M.T., Gouveia, L., Pesneau, P., 2011. Natural and extended formulations for the time-dependent traveling salesman problem. Discrete Applied Mathematics.

Gouveia, L., Voss, S., 1995. A classification of formulations for the (time-dependent) traveling salesman problem. European Journal of Operational Research 83, 69–82.

Gutin, G., Punnen, A., 2002. The Traveling Salesman Problem and Its Variations. Combinatorial Optimization. Kluwer Academic Publishers.

Letchford, A.N., Eglese, R.W., Lysgaard, J., 2002. Multistars, partial multistars and the capacitated vehicle routing problem. Mathematical Programming 94, 21–40.

Lin, S., 1965. Computer solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245–2269.

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

Lysgaard, J., Letchford, A.N., Eglese, R.W., 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100, 423–445.

Méndez-Díaz, I., Zabala, P., Lucena, A., 2008. A new formulation for the traveling deliveryman problem. Discrete Applied Mathematics 156, 3223–3237.

Müller, M. C. (1996). *Das dynamische travelling salesman problem*. Master's thesis, Universität Kaiserslautern, Fachbereich Mathematik.

Picard, J., Queyranne, M., 1978. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Operations Research 26, 86–110.

Rubin, P.A., Ragatz, G.L., 1995. Scheduling in a sequence dependent setup environment with genetic search. Computers & OR 22, 85–99.

Vander Wiel, R.J., Sahinidis, N.V., 1995. Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. Transportation Science 29, 167–183.

Vander Wiel, R.J., Sahinidis, N.V., 1996. An exact solution approach for the time-dependent traveling-salesman problem. Naval Research Logistics 43, 797–820.

van Eijl, C. A. (1995). *A polyhedral approach to the delivery man problem*. Memorandum COSOR 95-19, Eindhoven University of Technology.