The Time Dependent Traveling Salesman Problem: Polyhedra and Algorithm

Hernán Abeledo Ricardo Fukasawa Artur Pessoa
Eduardo Uchoa

December, 2010

Abstract

The Time Dependent Traveling Salesman Problem (TDTSP) is a generalization of the classical Traveling Salesman Problem (TSP), where arc costs depend on their position in the tour with respect to the source node. While TSP instances with thousands of vertices can be solved routinely, there are very challenging TDTSP instances with less than 100 vertices. In this work, we study the polytope associated to the TDTSP formulation by Picard and Queyranne, which can be viewed as an extended formulation of the TSP. We determine the dimension of the TDTSP polytope and identify several families of facet-defining cuts. We obtain good computational results with a branch-cut-and-price algorithm using the new cuts, solving almost all instances from the TSPLIB with up to 107 vertices.

1 Introduction

The Time-Dependent Traveling Salesman Problem (TDTSP) is a generalization of the Traveling Salesman Problem (TSP) where arc costs depend on their position in the tour. This work departs from a formulation by Picard and Queyranne [23] (proposed earlier in [28] as an extended formulation for the TSP), to define and study the TDTSP polytope. Our motivations are the following:

- The TDTSP itself is a rich problem, with a number of important applications. It includes as special cases routing problems, like the Traveling Deliveryman Problem (TDP), known also as the Cumulative TSP or as the Minimum Latency Problem; and scheduling problems, such as the $1|s_{ij}|\sum C_j$.
- The formulation in [23], called here PQ, can be generalized to provide very effective formulations to be used in branch-cut-and-price algorithms for several Vehicle Routing Problem (VRP) variants [24] (including "nasty" cases, like the heterogenous fleet VRP [25]) and also complex single and

multi-machine scheduling problems [26]. The TDTSP facet-defining inequalities studied in this paper can be readily generalized and used on those problems.

- The PQ formulation can be used for solving the TSP. Of course, every known valid inequality for the TSP could still be added to the PQ formulation. However, we verified that inequalities known to define facets of the TSP polytope [3] correspond to disappointingly low dimensional faces of the TDTSP polytope and are usually dominated by the newly proposed TDTSP inequalities. This means that adding TDTSP inequalities to the PQ formulation yields a TSP formulation that is potentially stronger than those usually used, at the expense of having n times more variables. Furthermore, we believe the TDTSP inequalities may be projected into more complex, yet unknown, valid inequalities for the TSP polytope. Our hope is supported by some precedents. For example, [2] derived new Symmetric TSP (STSP) facets from known Asymmetric TSP (ATSP) facets. Similarly, [11] provides another case where relatively simple facets of an extended formulation are combined and projected into new complex facets of the original formulation.

Polyhedral studies of the TSP have been very productive, both theoretically and because of their algorithmic implications. Results for the STSP are surveyed in [13] and for the ATSP in [3]. Formulations for the TDTSP have been proposed or studied in [23, 20, 7, 10, 29, 9]. Exact algorithms for the TDTSP are presented in [20, 4, 29] and, for the special case of the TDP, in [6, 17, 18].

This paper is organized as follows. The TDTSP polytope is defined in Section 2, where its dimension is also established. There is also a discussion about the decision of investigating the actual TDTSP polytope and not its monotone relaxation. Section 3 presents Admissible Flow Constraints, a family of strong inequalities, including an important subfamily of inequalities proven to define facets of the TDTSP polytope and with nice theoretical properties related to flow decomposition. Section 4 introduces Lifted Subtour Elimination Constraints, which are a new family of facet-defining inequalities. Section 5 deals with Triangle Clique Constraints. Those last inequalities were already introduced in the VRP context [24], but now we show that some, and perhaps all, define facets of the TDTSP polytope. Section 6 describes a branch-cut-and-price algorithm for the TDTSP, separating the newly proposed inequalities, while Section 7 presents computational results. Some lengthier proofs are shown in the Appendix.

2 Preliminaries

Let $N = \{1, 2, \ldots n\}$ and let $N_0 = N \cup \{0\}$. For a set of nodes S, $K(S)$ shall denote the complete (loopless) digraph over S. It is known that there is a one-to-one correspondence between Hamiltonian tours of $K(N_0)$ and Hamiltonian paths (with free ends) of $K(N)$.

The TDTSP on a complete graph $K(N_0)$ can be modeled as an optimization problem over a layered graph (V, A), where V consists of a source node 0, a terminal node T, and intermediate nodes (i, t) for $i, t \in N$. The first index of an intermediate node (i, t) identifies vertex i of the graph $K(N)$ and the second index will represent the position of vertex i in a path between nodes 0 and T. The arc set A is composed of three types of arcs. For $i \in N$, $(0, i, 0)$ denotes an arc from node 0 to node $(i, 1)$ and (i, T, n) denotes an arc from node (i, n) to node T. Given $i, j \in N$ such that $i \neq j$ and $1 \leq t \leq n-1$, (i, j, t) will denote an (intermediate) arc from node (i, t) to node $(j, t+1)$. The third index of an arc is its *layer*. Likewise, the second index of an intermediate node identifies its node layer.

It is convenient to define $G(n)$ to be the subgraph of (V, A) induced by $V \setminus \{0, T\}$. Thus, $G(n)$ has n^2 nodes $\{(i, t) : i, t \in N\}$ and all the $n(n-1)^2$ intermediate arcs of A. A path with n vertices in $G(n)$ is of the form $\{(v_t, t) : v_t \in N, 1 \leq t \leq n\}$. Since consecutive nodes in the path are in consecutive layers, we can describe such paths by an ordered array $(v_t : t \in N)$. Such a path can be extended to a $0 - T$ path of (V, A) by appending nodes 0 and T as first and last nodes, respectively. A path in $G(n)$ with node sequence $(v_t : t \in N, v_i \neq v_j \text{ for } i \neq j)$ that corresponds to a permutation of the elements of N will be called an *s-path*. A $0-T$ path of (V, A) will also be called an s-path if it contains an s-path of $G(n)$. Clearly, there is a one-to-one correspondence between s-paths of $G(n)$ and Hamiltonian paths of $K(N)$. Similarly, an s-path of (V, A) corresponds to a Hamiltonian tour of $K(N_0)$, where nodes 0 and T both represent node 0 of $K(N_0)$. Figure 1 represents the arcs belonging to a certain s-path in $G(6)$ as lines in a 6×6 grid. A similar representation will be used over this paper to illustrate the support graph of several families of constraints.

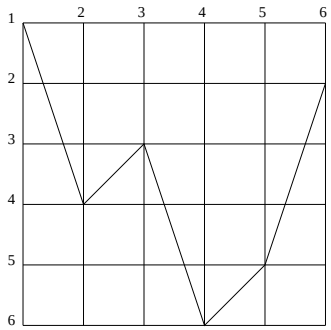

Figure 1: Graphical representation of the s-path $(1, 4, 3, 6, 5, 2)$ in $G(6)$.

Picard and Queyranne [23] formulated the TDTSP over (V, A) as a linear integer program with the following set of constraints, where variable $x_{i,j}^t$ with cost $c_{i,j}^t$ indicates if arc (i, j, t) is used and $N \setminus i$ denotes $N \setminus \{i\}$.

$$\sum_{j \in N} x^0_{0,j} = 1 \tag{1a}$$

$$x^0_{0,j} = \sum_{k \in N \setminus j} x^1_{j,k},\ j = 1 \dots n \tag{1b}$$

$$\sum_{i \in N \setminus j} x^t_{i,j} = \sum_{k \in N \setminus j} x^{t+1}_{j,k},\ j = 1 \dots n, t = 1 \dots n-2 \tag{1c}$$

$$\sum_{i \in N \setminus j} x^{n-1}_{i,j} = x^n_{j,T},\ j = 1 \dots n \tag{1d}$$

$$x^0_{0,j} + \sum_{t=1}^{n-1} \sum_{i \in N \setminus j} x^t_{i,j} = 1,\ j = 1 \dots n \tag{1e}$$

$$x \geq 0 \text{ and integer} \tag{1f}$$

We can use equations (1a) and (1d) to eliminate the $2n$ variables corresponding to arcs incident to nodes 0 and T, obtaining the following equivalent system of constraints whose solutions correspond to s-paths in $G(n)$.

$$\sum_{i \in N} \sum_{j \in N \setminus i} x^1_{i,j} = 1 \tag{2a}$$

$$\sum_{i \in N \setminus j} x^t_{i,j} = \sum_{k \in N \setminus j} x^{t+1}_{j,k},\ j = 1 \dots n, t = 1 \dots n-2 \tag{2b}$$

$$\sum_{k \in N \setminus j} x^1_{j,k} + \sum_{t=1}^{n-1} \sum_{i \in N \setminus j} x^t_{i,j} = 1,\ j = 1 \dots n \tag{2c}$$

$$x \geq 0 \text{ and integer} \tag{2d}$$

Definition 1 *Let $P(n)$ be the convex hull of the incidence vectors of s-paths of $G(n)$ and refer to it as the TDTSP polytope.*

Clearly, $P(n)$ and the convex hull of s-paths of (V, A) are equivalent polytopes with the same dimension. By enumerating all integer vectors in $P(n)$, one can determine computationally that $\dim P(1) = 0$, $\dim P(2) = 1$, $\dim P(3) = 5$, $\dim P(4) = 22$, and $\dim P(5) = 60$. We establish the dimension of $P(n)$ below.

Lemma 1 *The system of equations (1a, 1b, 1c, 1d, 1e) has rank $n^2 + n$.*

Proof The subsystem composed by the $n^2 + 1$ flow conservation equations (1a, 1b, 1c, 1d) has (full) rank n^2+1 since they are the flow conservations constraints for the connected digraph $G = (V, A)$ which has $n^2 + 2$ nodes (note that the flow conservation constraint for terminal node T is absent).

We prove next that exactly one of the n equations (1e) is redundant in the system (1a, 1b, 1c, 1d, 1e). Equation (1a) combined with flow conservation constraints (1b, 1c) imply that the total flow in each arc layer is equal to 1. That is,

$$\sum_{i \in N} \sum_{j \in N \setminus i} x^t_{i,j} = 1, t = 1 \dots n-1.$$

The sum of all flow variables in the first n of layers $G = (V, A)$ equals n:

$$\sum_{j \in N} x^0_{0,j} + \sum_{t=1}^{n-1} \sum_{i \in N} \sum_{j \in N \setminus i} x^t_{i,j} = n \tag{3}$$

Thus, we can eliminate from (1e) the equation corresponding to an arbitrary single index $k \in N$ and (3) would imply that it still holds true:

$$x^0_{0,k} + \sum_{t=1}^{n-1} \sum_{i \in N \setminus k} x^t_{i,k} = 1.$$

To complete our proof, we show that if more than one equation from (1e) is eliminated then the set of solutions to the remaining system is enlarged, implying that the rank of the system of equations decreased. Suppose then that equations (1e) corresponding to indices k and l are eliminated. Consider the incidence vector of the path $(0, k, S, k, T)$, where S is any permutation of $N \setminus \{k, l\}$. Finally, note that this vector satisfies all remaining constraints but is not a feasible solution for the original system of equations (1a, 1b, 1c, 1d, 1e) since constraints (1e) are violated for k and l. ∎

Lemma 2 *The system of equations (2a, 2b, 2c) has rank $n^2 - n$.*

The above result follows immediately from Lemma 1. The removal of any single equation from (2), such as (2a), yields a full rank system of equations.

Theorem 3 *If $n \geq 5$, then dimension of $P(n) = n(n-1)(n-2)$.*

Proof Lemma 2 implies that $\dim P(n) \leq n(n-1)(n-2)$. We prove by induction that we can choose $n(n-1)(n-2)+1$ linearly independent (LI) vectors in $P(n)$. The induction basis for $n = 5$ is established computationally. Suppose the result is true for $n \geq 5$, we need to show that $\dim P(n+1) = (n+1)n(n-1)$. Consider $G(n+1)$ and its subgraph $G(n)$. The induction hypothesis asserts that $G(n)$ contains $\dim P(n) + 1 = n(n-1)(n-2) + 1$ LI s-paths. Each of these s-paths can be trivially extended to an s-path of $G(n+1)$ by simply appending node $(n+1, n+1)$ as the last node of the path. This yields $\dim P(n) + 1 = n(n-1)(n-2) + 1$ LI s-paths in $G(n+1)$. To complete the required set of LI s-paths in $G(n+1)$ we need $3(n^2 - n) = \dim P(n+1) - \dim P(n)$ additional s-paths. We iteratively construct the required set of LI s-paths by including, in each successive s-path, an arc that was not used previously. Note that, compared with $G(n)$, $G(n+1)$ has $3n^2 - n$ additional arcs and we need $3n^2 - 3n$ new s-paths. Thus, there are $2n$ "surplus" arcs available.

Extending the set of LI s-paths in the induction hypothesis from $G(n)$ to $G(n+1)$ consumed the n arcs incident to node $(n+1, n+1)$. Thus, we have only n surplus arcs remaining in order to construct the $3(n^2 - n)$ s-paths we still need. All additional s-paths defined next will contain a node of type $(n+1, t)$ for $t = 1, \ldots, n$ and will use one surplus arc per node $(n+1, t)$.

We consider two cases: (1) s-paths that contain node $(n+1,1)$ and (2) s-paths that contain a node $(n+1,t)$, for $t>1$. For case 1, we will construct a set of n^2-1 linearly independent s-paths that use the n incident arcs at node $(n+1,1)$ and the $n(n-1)$ arcs $\{(i,j,n),\ 1\leq i,j\leq n,\ i\neq j\}$. Our construction rests on the following observation: given a pair of arcs $(n+1,i,1)$ and (j,k,n) such that $i\neq j$ and $i\neq k$, we can always find an s-path that contains these two arcs.

We begin by fixing arc $(n+1,1,1)$ and generate a set of s-paths, each one terminating with a different arc (i,j,n). Thus, the possible final arcs are $\{(j,k,n),\ 2\leq j,k\leq n,\ j\neq k\}$, yielding $(n-1)(n-2)$ LI s-paths. Next, for each arc $(n+1,i,1)$, $2\leq i\leq n$, we construct an s-path that terminates in one of the (already used) arcs (j,k,n), $2\leq j,k\leq n$, $j\neq k$. This produces $n-1$ additional s-paths. Next, we select arc $(n+1,2,1)$ and use it to produce $2(n-2)$ s-paths terminating with either arcs $(j,1,n)$, $j=3,\ldots,n$ or $(1,k,n)$ $k=3,\ldots,n$. Finally, we fix arc $(n+1,3,1)$ and use it to generate two s-paths, terminating with arcs $(2,1,n)$ and $(1,2,n)$, respectively. This procedure created $n^2-1=(n-1)(n-2)+(n-1)+2(n-2)+2$ LI s-paths.

For case 2, where $t=2,\ldots,n$, each node $(n+1,t)$ has n incoming arcs $\{(i,n+1,t-1):i=1,\ldots,n\}$ and n outgoing arcs $\{(n+1,j,t):j=1,\ldots,n\}$. We will construct $2n-1$ LI s-paths that contain node $(n+1,t)$. Note that, given a pair of arcs $\{(i,n+1,t-1),(n+1,j,t)\}$, such that $i\neq j$, it is always possible to include this pair of arcs within an s-path.

We first fix the incoming arc $(1,n+1,t-1)$. This arc can be combined with an outgoing arc of the form $(n+1,j,t)$ for $j=2,\ldots,n$ to be part of an s-path of $G(n+1)$. Similarly, outgoing arc $(n+1,n,t)$ can be combined with incoming arcs of the form $(i,n+1,t-1)$ for $i=2,\ldots n-1$ to produce $n-2$ s-paths. Pairing arc $(2,n+1,t-1)$ with $(n+1,1,t)$ yields one more path. Finally, we combine arcs $(n,n+1,t-1)$ and $(n+1,1,t)$ to obtain the last s-path needed to complete the set of $2n-1$ LI s-paths containing node $(n+1,t)$. Repeating this procedure for each $t=2,\ldots n$ yields $(n-1)(2n-1)$ LI s-paths.

In summary, we created a total of $3n^2-3n=(n^2-1)+(n-1)(2n-1)$ LI s-paths, in addition to the ones obtained from the induction hypothesis. ∎

We remark that we could have used the n^2+n LI equalities identified in Lemma 2 to remove n^2+n variables from (2). The potential advantage would be an alternative definition for the TDTSP polytope, that would be full-dimensional. However, we could not find any "simple" way of choosing those variables. For example, it is not possible to remove all the n^2+n variables from the same layer. Removing variables from different layers, in a complex way, would define a complex full-dimensional TDTSP polytope, where the facets are represented (uniquely up to scalar multiplication) by inequalities that have an obscure interpretation. Therefore, we preferred to work with a polytope that is not full-dimensional, but has facets that can be described by inequalities with a clear interpretation.

Another alternative to avoid the technical difficulties of non-full-dimensional polytopes would be working with monotonized polytopes:

Definition 2 *Let $P_{mon}(n)$ be the convex hull of the incidence vectors that can be extended to an s-path of $G(n)$ and refer to it as the monotonized TDTSP polytope.*

Is is clear that $P_{mon}(n)$ is full-dimensional and that $P(n) = P_{mon}(n) \cap$ (2a, 2b, 2c). Many of the classical polyhedral results for the STSP and ATSP were actually obtained on monotonized polytopes [13, 3]. However, we realized that a similar approach in the TDTSP would have its own pitfalls. The first pitfall is that the relaxation from $P(n)$ to $P_{mon}(n)$ may create many redundant facets. Indeed, the following Lemma shows an example of an exponential familty of distinct facets of $P_{mon}(n)$ that define the same facet of $P(n)$. The proof of Lemma 4 will be given by construction of an example in Section 5.

Lemma 4 *There are cases where $\Omega(2^n)$ inequalities defining distinct facets of $P_{mon}(n)$ define a single facet of $P(n)$.*

When the STSP or ATSP polytopes are monotonized, only $\Theta(n)$ LI equalities are relaxed. On the other hand, the monotonization of the TDTSP polytope relaxes $\Theta(n^2)$ LI equalities. This larger "dimension gap" makes the relation between original and monotonized polytopes less direct. In particular, Lemma 4 suggests that the relaxation from $P(n)$ to $P_{mon}(n)$ creates many redundant facets.

Besides that, there is also the natural difficulty that appears when dealing with monotonized polytopes: determining whether an inequality that defines a facet of the monotonized polytope also defines a facet of the original polytope. In the TSP case, there are some known simple sufficient conditions that may help a lot on that [3]. We could not find similar conditions for the TDTSP. In our experience, the knowledge that an inequality defines a facet of $P_{mon}(n)$ provides little help on proving that it defines a facet of $P(n)$, this being as difficult as making the proof from the scratch. For those reasons, we chose to work in the original polytope $P(n)$.

3 Admissible Flow Constraints

Let $p = (0, v_1, v_2, \ldots, v_n, T)$ be a $0 - T$ path in (V, A). We define an r-cycle in p as a subpath $(v_i, \ldots, v_{i+r})$ such that $v_i = v_{i+r}$. Note that no path p contains 1-cycles, since A does not has arcs of type (j, j, t). Also note that integral solutions of (1) are s-paths and do not contain r-cycles. A network flow in an acyclic digraph can be decomposed as a sum of flows along paths. In particular, a fractional solution satisfying equalities (1a, 1b, 1c, 1d) can be decomposed into a set of $0 - T$ paths. However, these paths may contain r-cycles, for some $r \geq 2$. The Admissible Flow Constraints are devised to improve the formulation by restricting the occurrence of r-cycles.

Consider t such that $1 \leq t \leq n - 2$. The flow on arc (i, j, t) should exit node $(j, t+1)$ using arcs other than $(j, i, t+1)$ to avoid creating a 2-cycle. Constraints below model this observation.

$$x_{i,j}^t \leq \sum_{k \in N \setminus \{i,j\}} x_{j,k}^{t+1}, \quad (i,j,t) \in A, 1 \leq t \leq n-2. \tag{4}$$

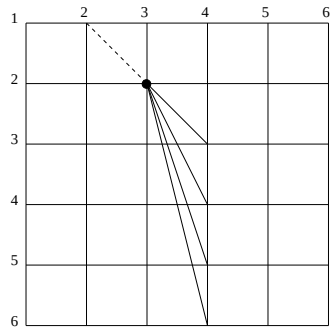

Figure 2: Example of a 2-cycle constraint: $x_{1,2}^2 \leq x_{2,3}^3 + x_{2,4}^3 + x_{2,5}^3 + x_{2,6}^3$.

We next show that inequalities (4) define a family of facets of $P(n)$.

Theorem 5 *If $n \geq 6$, then each constraint of (4) defines a distinct facet of $P(n)$.*

Proof We first prove that each constraint of (4) defines a facet of $P(n)$. Assume without loss of generality (by potentially relabeling vertices) that $i = 1$ and $j = n$. Note that arc $(1, n, t)$ enters node $(n, t+1)$ where $1 \leq t \leq n-2$. To show that constraint (4) is not satisfied as an equation for all vectors of $P(n)$, we note there exists an s-path that contains arcs $(2, n, t)$ and $(n, 3, t+1)$. The incidence vector of such a path satisfies constraint (4) for arc $(1, n, t)$ as strict inequality.

Next, since $\dim P(n) = n(n-1)(n-2)$, we need to show that $n(n-1)(n-2)$ linearly independent vectors in $P(n)$ satisfy constraint (4) associated with arc $(1, n, t)$ as equality. The construction in the proof of Theorem 3 shows that we can find $\dim P(n) + 1$ linearly independent s-paths in $G(n)$. We partition this set of linearly independent s-paths into two subsets, B and $\overline{B}$, such that B are the s-paths that do not contain node $(n, t+1)$ and $\overline{B}$ are the s-paths that contain node $(n, t+1)$.

The s-paths in B all satisfy constraint (4) associated with arc $(1, n, t)$ as equality with value 0 since they do not contain node $(n, t+1)$. In the proof of Theorem 3, $|\overline{B}| = 2n - 3$. We replace $\overline{B}$ by a set B' composed of $2n - 4$ linearly independent s-paths, all of which satisfy constraint (4) for arc $(1, n, t)$ as equality.

The set B' is constructed as follows. For each $k \in N \setminus \{1, n\}$ we combine incoming arc (k, n, t) with outgoing arc $(n, 1, t+1)$ to generate $n - 2$ linearly independent s-paths which satisfy (4) as equation with value 0. Finally, we combine incoming arc $(1, n, t)$ with the $n - 2$ outgoing arcs $(n, k, t+1)$, for

$k \in N \setminus \{1, n\}$ to obtain $n-2$ independent s-paths which satisfy (4) as equation with value 1. Note $B \cup B'$ is linearly independent by construction and $|B \cup B'| = \dim P(n)$.

To prove that each inequality in (4) defines a different facet, it suffices to show that, given two distinct (4) constraints, there is an s-path satisfying one constraint at equality while satisfying the other at strict inequality. This same type of argument will be used throughout the paper.

There exists one inequality (4) for each (i, j, t). To simplify notation assume that one of the constraints is defined by $i = 1$, $j = 2$, that is, we have two distinct constraints $(1, 2, t)$ and (i, j, t').

Notice that if an s-path does not go through node $(2, t+1)$ then that s-path satisfies constraint $(1, 2, t)$ at equality. Also if an s-path uses arc $(1, 2, t)$ then that s-path also satisfies constraint $(1, 2, t)$ at equality.

So if $j \neq 2$ or $t \neq t'$ one can easily construct an s-path that does not go through node $(2, t+1)$ and that goes through node $(j, t'+1)$ from node (k, t') and exits through node $(l, t'+2)$ with $k, l \notin \{2, i, j\}$ and $k \neq l$.

If $t' = t$ and $j = 2$, then one can just consider the s-path going through $(2, t+1)$ from node $(1, t)$ and leaving to node $(k, t+2)$ for $k \neq i$. This s-path uses arc $(1, 2, t)$ and arc $(2, k, t+1)$ which is in constraint (i, j, t') and therefore does not satisfy the latter at equality. ∎

The next result is that inequalities (4) suffice to eliminate all 2-cycles, that is, any solution satisfying the original Picard and Queyranne TDTSP constraints plus (4) will be in the convex hull of $0-T$ paths without 2-cycles. The following lemma will be used in our proof. It relies on a characterization of feasible network flow problems obtained by Gale and Hoffman which, if applied to balanced transportation problems on incomplete bipartite graphs, yields a generalization of Hall's marriage theorem [14].

Lemma 6 *[8, 15] Let S and D be the set of supply and demand nodes of a balanced transportation problem and suppose $N(R) = D$ for every subset $R \subset S$ such that $|R| = 2$ ($N(R)$ is the set of neighbors of R). Then the transportation problem is feasible if and only if $b(v) \leq b(N(v))$ for each supply node $v \in S$ ($b(v)$ is the supply of v, $b(N(v))$ is the total demand of $N(v)$).*

We proceed to show the desired result.

Theorem 7 *Let $x \in R^A_+$ satisfy constraints (1a, 1b, 1c, 1d) and (4). Then x can be decomposed into flows along $0-T$ paths such that none of them contains a 2-cycle.*

Proof Let $x \in {R^A}_+$ satisfy the assumptions of the theorem. We show that x is a convex combination of incidence vectors of $0-T$ paths which do not contain two-cycles.

Let (j, t) be a node of (V, A) such that $1 < t < n$. We consider the flow on arcs incident to node (j, t), along incoming arcs $\{x^{t-1}_{i,j} : i \in N \setminus j\}$ and outgoing arcs $\{x^t_{j,k} : k \in N \setminus j\}$. Associated with node (j, t), we construct a

transportation problem with $(n-1)$ source nodes $S = N \setminus j$ and a symmetrical set of demand nodes $D = N \setminus j$. Each node $i \in S$ has its available supply set equal to $x_{i,j}^{t-1}$ and each node $k \in D$ has its demand set equal to $x_{j,k}^{t}$. For each pair $i, k \in N \setminus j$ such that $i \neq k$, we place an arc (i, k) connecting node $i \in S$ with $k \in D$.

The above transportation problem satisfies the condition of Lemma 6. In particular, a feasible solution to this transportation problem provides a way of decomposing the entire flow along arcs incident to node (j, t) into flow along paths of length two, where each one is of the form $(i, t-1), (j, t), (k, t+1)$ and $i \neq k$. None of these paths of length two in the decomposition, when viewed as a path in the graph $K(N)$, forms a two-cycle in $K(N)$.

Given a flow in x of (V, A) and a feasible solution for each of the transportation problems described above, we combine them to construct a feasible flow for a larger network as follows. The new network is created by substituting each node (j, t) in $G = (V, A)$ with $j \in N$ and $2 \leq i \leq n-1$, by a bipartite digraph with uncapacitated arcs as described above. The arc flows in each of these bipartite graphs are set equal to the feasible solutions of the corresponding transportation problem.

Nodes 0, T, and (j, t) for $j \in N$ and $t = 1$ or n remain in the new network without further node splitting. Each arc (i, j, t) in the original graph (V, A) would have a corresponding arc, with flow value set equal to $x_{i,j}^{t}$, connecting the appropriate nodes in the new network. Clearly, the flow thus defined in the new network is feasible for a problem where all nodes, except 0 and T, are transshipment nodes. Thus, the flow can be decomposed as a sum of flows on paths (in the new network) that start at node 0 and end at node T. Finally, by shrinking each of the bipartite graphs to its original node in (V, A), each of these paths can be shortened to a path of (V, A) that does not contain a two-cycle. This procedure yields the desired decomposition of x. ∎

Inequalities (4) may be aptly called *2-cycle elimination constraints*. The complete elimination of larger r-cycles by means of inequalities appears to be much more difficult, even for $r = 3$. Nevertheless, the following generalization of those inequalities proved to be a rich source of strong cuts.

Definition 3 *Let X be a connected set of vertices of $G = (V, A)$ not containing vertices in $\{0, T\}$. If $e \in \delta^-(X)$, define $C(X, e) \subseteq \delta^+(X)$ as the set of leaving arcs that are* admissible *for e with respect to X: those arcs f that belong to an s-path entering X at e and leaving X for the first time at f. For a set $E \subseteq \delta^-(X)$, define $C(X, E) \subseteq \delta^+(X)$ as $\cup_{e \in E} C(X, e)$. For a given X and E, the following valid inequality is called an Admissible Flow Constraint (AFC):*

$$\sum_{e \in E} x_e \leq \sum_{f \in C(X,E)} x_f \tag{5}$$

The AFCs where $|E| = 1$ are called unitary AFCs.

Notice that (4) is a unitary AFC with $|X| = 1$. Another subset of AFCs that is worth mentioning is the r-cycle elimination constraints, which also generalize (4).

Definition 4 *Let* $((i,t),(u_1,t+1),\ldots,(u_{r-1},t+r-1),(u_r,t+r))$, $u_r = i$, *be a minimal* r*-cycle in* G*. The unitary AFCs where* $X = \{(u_1,t+1),\ldots,(u_{r-1},t+r-1)\}$ *and* $E = \{(i,u_1,t)\}$ *are called* r-cycle elimination constraints *and are written as:*

$$x^t_{i,u_1} \leq \sum_{k=1}^{r-1} \sum_{j \in N \setminus \{i,u_1,\ldots,u_{k+1}\}} x^{t+k}_{u_k,j}.$$

Computational experiments and partial results not stated here support the conjecture that all r-cycle elimination constraints (not only the 2-cycle elimination) are facet-defining. Figure 3 illustrates a 4-cycle elimination constraint that is facet-defining. Figure 4 depicts an example of a unitary AFC over a certain set X that is also facet-defining. Figure 5 shows an example of an AFC over the same X, but with $|E| = 5$, that is not facet-defining (it has dimension 119). While the more general AFCs are usually not facet-defining (it seems pretty hard to determine which ones are), they are still very useful as cuts.

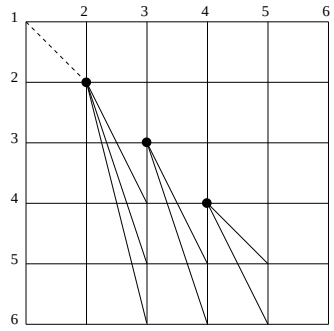

Figure 3: Example of a 4-cycle constraint: $x^1_{1,2} \leq x^2_{2,4} + x^2_{2,5} + x^2_{2,6} + x^3_{3,5} + x^3_{3,6} + x^4_{4,5} + x^4_{4,6}$.

4 Lifted Subtour Elimination Constraints

The classical Subtour Elimination Constraints (SECs) [5] are known to define facets of the STSP polytope [12] and also of the ATSP polytope [13]. SEC inequalities can be expressed in terms of the TDTSP variables as follows:

$$\sum_{j \in S} x^0_{0,j} + \sum_{t=1}^{n-1} \sum_{i \notin S} \sum_{j \in S} x^t_{i,j} \geq 1, \quad S \subset N, |S| > 1. \tag{6}$$

Eliminating the variables of arcs not in $G(n)$, we obtain the equivalent inequalities:

$$\sum_{i \in S} \sum_{j \in N \setminus j} x^1_{i,j} + \sum_{t=1}^{n-1} \sum_{i \notin S} \sum_{j \in S} x^t_{i,j} \geq 1, \quad S \subset N, |S| > 1. \tag{7}$$

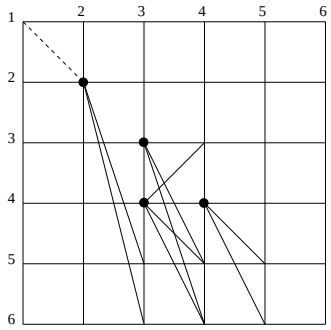

Figure 4: Example of a unitary AFC constraint: $x^1_{1,2} \leq x^2_{2,5} + x^2_{2,6} + x^3_{3,5} + x^3_{3,6} + x^3_{4,3} + x^3_{4,5} + x^3_{4,6} + x^4_{4,5} + x^4_{4,6}$.

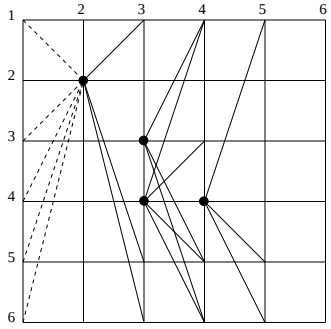

Figure 5: Example of a general AFC constraint: $x^1_{1,2}+x^1_{3,2}+x^1_{4,2}+x^1_{5,2}+x^1_{6,2} \leq x^2_{2,1}+x^2_{2,5}+x^2_{2,6}+x^3_{3,1}+x^3_{3,5}+x^3_{3,6}++x^3_{3,1}+x^3_{4,1}+x^3_{4,5}+x^3_{4,6}+x^4_{4,1}+x^4_{4,5}+x^4_{4,6}$.

Even though SECs define facets of the ATSP polytope, (7) may define quite low-dimensional faces of the TDTSP polytope. In fact, our next class of inequalities is a much stronger family of valid TDTSP inequalities that we call *Lifted Subtour Elimination Constraints* (LSECs):

$$\sum_{i \in S} \sum_{j \in N \setminus j} x^1_{i,j} + \sum_{t=1}^{n-|S|} \sum_{i \notin S} \sum_{j \in S} x^t_{i,j} \geq 1, \quad S \subset N, |S| > 1. \tag{8}$$

Figure 6 depicts the support graph of a LSEC over a set S of size 3.

The above inequality states that an s-path $\{v_t : t \in N\}$ must satisfy $v_1 \in S$ or $\{v_k, v_{k+1} : v_k \notin S, v_{k+1} \in S, 1 \leq k \leq n-|S|\}$. That is, an s-path either starts at a vertex in S or it must enter S no later than layer $n - |S|$. This constraint is valid because an s-path entering S for the first time after layer $n - |S|$ will not be able to cover all elements of the set S. Similarly, the inequality below states that an s-path either ends at a vertex in S or leaves S at layers greater

or equal to $|S|$. This is a valid constraint because an s-path that exits the set S before arc layer $|S|$ will not have covered the set S completely and, thus, must return to it.

$$\sum_{t=|S|}^{n-1} \sum_{i \in S} \sum_{j \notin S} x_{i,j}^t + \sum_{j \in S} \sum_{i \in N \setminus j} x_{i,j}^{n-1} \geq 1, \qquad S \subset N, |S| > 1. \tag{9}$$

Let $\bar{G}(n)$ be the graph obtained from $G(n)$ by reversing all its arcs and the order of the node layers. Clearly, each s-path in $G(n)$ corresponds to a unique s-path in $\bar{G}(n)$. Note that constraint (9) can be viewed as a constraint of type (8) for the s-paths of the graph $\bar{G}(n)$, using the same set S in both inequalities. We can conclude that inequality (8) for a fixed set S defines a facet of $P(n)$ if and only if inequality (9), for the same set S, defines a facet of $P(n)$.

Lemma 8 *Inequality* (8) *defines a facet of* $P(n)$ *if and only if inequality* (9) *also does.*

Our main result for this section establishes that lifted subtour elimination constraints define a family of distinct facets. The proof of Theorem 9 is left to the Appendix.

Theorem 9 *If* $n \geq 6$*, and* $3 \leq |S| \leq n-3$*, then each constraint* (8) *and* (9) *defines a distinct facet of* $P(n)$*.*

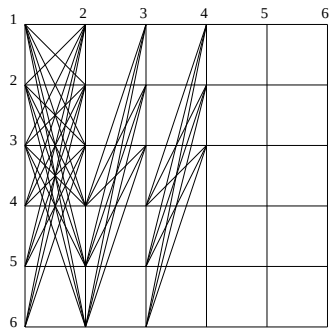

Figure 6: Support of a lifted subtour elimination constraint for set $S = \{1, 2, 3\}$.

As a final result on this section, we show that the facets defined by the lifted subtour inequalities are different than the ones defined by the two-cycle inequalities (4).

Lemma 10 *Assume that* $n \geq 6$*. Let* $S \subset N$ *be such that* $3 \leq |S| \leq n-3$ *and let* (i, j, t) *be an arc of* $G(n)$ *such that* $t \leq n-2$*. Then, the two-cycle elimination constraint (4) defines a facet of* $P(n)$ *that is different from the facets defined by the lifted subtour inequalities (8) and (9).*

Proof Regardless of whether $j \in S$ or not, it easy to find an s-path that does not pass by node $(j, t+1)$ and that enters the set S exactly twice before arc layer $n - |S| + 1$, say firstly at layer 1 and secondly at layer 3. Such an s-path satisfies the lifted subtour inequality (8) as strict inequality ($2 > 1$) and the two-cycle elimination constraint (4) as equality ($0 = 0$). Similarly, we can find an s-path that avoids node $(j, t+1)$ and that exits the set S exactly twice after arc layer $|S|-1$, say firstly at layer $|S|$ and secondly at layer $|S|+2$. Again, such an s-path satisfies the lifted subtour inequality (9) as strict inequality ($2 > 1$) and the two-cycle elimination constraint (4) as equality ($0 = 0$). ∎

5 Triangle Clique Constraints

A well-known way of deriving strong cuts for binary integer programs is by analyzing the incompatibility graph of the variables. This graph has a vertex for each binary variable and an edge for each pair of variables that are incompatible, i.e., they can not have both value 1 in any solution. As each solution must induce an independent set in this graph, known facets of the independent set polytope, like clique and odd-hole inequalities [22], yield potentially strong cuts. This approach can not be used on the STSP, since any pair of edge variables can appear in some tour. However, the arc variables in the ATSP define an interesting incompatibility graph. While no clique cuts exist, in fact they are dominated by degree constraints or SECs, the facet-defining Odd Closed Alternating Trail Constraints correspond to odd-holes in the incompatibility graph. The arc-time variables in the TDTSP provide an even richer incompatibility graph, where even simple cliques provide new families of facet-defining cuts.

Let $S \subset N$ satisfy $|S| = 3$ and consider two arcs (i, j, t), (i', j', t') of $G(n)$ such that $(i, j), (i', j') \in A(S)$. Note that these two arcs are compatible if and only if they are adjacent and do not form a 2-cycle. Since few such pairs of arcs are compatible, it is more convenient to work over the compatibility graph, the complement of the incompatibility graph. Given $S = \{i, j, k\} \subset N$, let $\mathcal{G}(\mathrm{S}) = (\mathcal{V}, \mathcal{E})$ be the compatibility graph associated to S, where each vertex of $\mathcal{V}$ is an arc (i, j, t) of $G(n)$ with $(i, j) \in A(S)$ and each edge of $\mathcal{E}$ is a compatible pair $\{(i, j, t), (j, k, t+1)\}$ (for ease of notation, we omit the dependence on S from $\mathcal{V}$ and $\mathcal{E}$). An independent set $\mathcal{I}(S) \subset \mathcal{V}$ is a maximal set of vertices in $\mathcal{G}(\mathrm{S})$ which are all pairwise incompatible. It is clear that the following inequality is valid for any S and any $\mathcal{I}(S) \subset \mathcal{V}$:

$$\sum_{(i,j,t) \in \mathcal{I}(S)} x_{i,j}^t \leq 1 \tag{10}$$

These constraints were proposed in [24] for a more general setting and were named *Triangle Clique Constraints*. We prove here that constraints (10) define facets of the TDTSP polytope when I has a certain regular structure, and conjecture that this result remains true for all triangle clique inequalities.

The independence sets we consider here induce bipartite subgraphs on alternating layers of $G(n)$. Let $A(S, t) = \{(i, j, t) : (i, j) \in A(S)\}$. Let $p \in \{0, 1\}$. We

define the set $\mathcal{I}(S,p) := \bigcup_{k=1}^{\lfloor \frac{n-p}{2} \rfloor} A(S, 2k-1+p)$ as an *alternating independence set*. Note that p defines the parity of the alternating layers of $\mathcal{I}(S,p)$.

The following Lemma will only be used to prove that (10) defines a facet of $P(n)$ if $\mathcal{I}$ is an alternating independence set. It will be useful in showing that one can construct by induction, in a manner similar to what was done in Theorem 3, enough linearly independent points satisfying (10) at equality. We state it here for a complete presentation of all results.

Lemma 11 *Let $n \geq 7$. Let $S \subset N$ with $|S| = 3$ and $\mathcal{I}(S,p)$ be an alternating independence set. Let $a = (i,j,s)$ and $b = (k,l,t)$ be two compatible arcs such that $k,l \notin S$ and either $|t-s| = 1$ or $\{s,t\} = \{1, n-1\}$. Then there exists an s-path containing $a = (i,j,s)$ and $b = (k,l,t)$ which also contains exactly one arc in $\mathcal{I}(S,p)$.*

The following theorem will show the desired result that (10) defines a facet for alternating independence sets. The proof that it defines a facet is left for the appendix. We only prove here that the facets are distinct.

Theorem 12 *If $n \geq 7$, then (10) defines a distinct facet of $P(n)$ for every distinct alternating independence set, $\mathcal{I}(S,p) \subset \mathcal{V}$.*

Proof Consider two distinct alternating independence sets $\mathcal{I}(S,p)$ and $\mathcal{I}(S',p')$.

Assume without loss of generality (by potentially relabeling vertices) that $S = \{1,2,3\}$ and that $7 \notin S \cup S'$.

If $S = S'$, then we may assume that $p = 0$ and $p' = 1$, so pick an s-path that goes through vertices $(1,1)$, $(2,2)$ and $(4,3)$. Such s-path cannot use any other arcs in $A(S,t)$ for any $t > 1$. Therefore it satisfies (10) for S at equality and for S' as a strict inequality.

If $S \neq S'$, then we may assume that $2 \notin S'$ and $4 \in S'$. Assume, in addition that $p = 0$ (the proof for $p = 1$ is similar). If $|S \cap S'| \leq 1$, we may assume $3 \notin S'$ and $5 \in S'$. If $|S \cap S'| = 2$, then $S' = \{1,3,4\}$. Either way, we can pick $i,j \in \{3,5\}$ with $i \neq j$ such that $i \notin S'$, $j \in S'$. So pick an s-path that goes through vertices $(1,1)$, $(2,2)$, $(4,3)$, $(i,4)$, $(j,5)$, $(7,6)$. Such an s-path satisfies (10) for S at equality and for S' as a strict inequality since it cannot use any arcs of $A(S',t)$ for any t. ∎

We next show that the facets defined by (10) are not the same as the ones defined by (8) and (9).

Lemma 13 *Assume that $n \geq 7$. Let $\mathcal{I}(S,p)$ be an alternating independent set and $S' \subseteq N$ such that $3 \leq |S'| \leq n-3$. Then the facet defined by inequality (10) for $\mathcal{I}(S,p)$ and the facets defined by inequality (8) and (9) for S' are not the same.*

Proof Assume without loss of generality that $S = \{1,2,3\}$. We consider only the cases where $p = 0$ and of (8). (the proof for $p = 1$ and (9) will be similar).

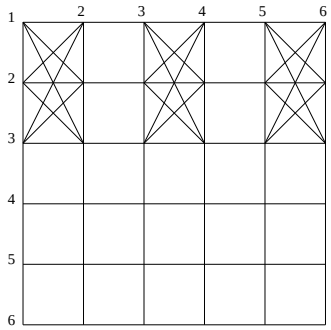

Figure 7: Support of an alternating triangle clique constraint.

If $1 \leq |S \cap S'| \leq 2$, we can assume that $2 \notin S'$, $1 \in S'$ and $4 \in S'$. So we can pick an s-path going through vertices $(1,1)$, $(2,2)$, $(4,3)$.

If $S \cap S' = \emptyset$, then we can assume $\{4,5,6\} \subseteq S'$, so pick an s-path going through vertices $(4,1)$, $(1,2)$, $(5,3)$, $(6,4)$, $(2,5)$, $(3,6)$.

If $S = S'$, then pick an s-path going through vertices $(1,1)$, $(2,2)$, $(4,3)$, $(3,4)$.

If $S \neq S'$ and $S \subset S'$, then we may assume that $4 \in S'$ and $5,6,7 \notin S'$. So pick an s-path going through vertices $(1,1)$, $(5,2)$, $(4,3)$, $(6,4)$, $(2,5)$, $(3,6)$.

Since $|S'| \leq n-3$ we have $n-|S'| \geq 3$ so all the s-paths constructed enter S' twice before $n-|S'|$. Hence, all the s-paths constructed satisfy (10) at equality and (8) as strict inequality. ∎

We also show that the facets defined by (10) are not the same as the ones defined by 2-cycle inequalities.

Lemma 14 *Assume that $n \geq 7$. Let $\mathcal{I}(S,p)$ be an alternating independent set and $(i,j,t) \in A$ with $1 \leq t \leq n-2$. Then the facet defined by inequality* (10) *for $\mathcal{I}(S,p)$ and the facet defined by inequality* (4) *for (i,j,t) are not the same.*

Proof Assume without loss of generality that $S = \{1,2,3\}$.

If $i,j \notin S$, then construct an s-path that goes through nodes $(1,1+k)$, $(i,2+k)$, $(2,3+k)$, $(j,4+k)$, $(3,5+k)$ which does not satisfy (10) at equality and satisfies (4) at equality for some value of $k \in \{0,1\}$ since it does not go through node $(j,t+1)$.

If $i \notin S, j \in S$, then assume $j = 3$ and construct an s-path that goes through nodes $(1,2-p+2k)$, $(i,3-p+2k)$, $(2,4-p+2k)$, $(3,5-p+2k)$ which does not satisfy (10) at equality since it goes through arc $(2,3,t)$ at the incorrect parity and satisfies (4) at equality for some value of $k \in \{0,1\}$ since it does not go through node $(j,t+1)$.

Similar constructions cover the remaining cases. ∎

The last polyhedral results in this paper are intended to show some pitfalls that would arise if one works with the monotonized TDTSP polytopes. We

define another family of clique constraints. Let i be a vertex in N. Let $\mathcal{I}$ be a maximal set of incompatible arcs of $G(n)$ having one endpoint in i. The constraint $\sum_{(i,j,t)\in\mathcal{I}} x^t_{i,j} \leq 1$ is called a *star-clique*. Many star-cliques do define facets of $P(n)$, but they are not interesting because the facets are the same already defined by 2-cycle constraints. For example, Figure 8 depicts the support graph of a star-clique that is equivalent to the 2-cycle shown in Figure 2. The following result for the monotonized TDTSP polytope is easily obtained:

Theorem 15 *The clique constraints corresponding to any maximal set of incompatible arcs of $G(n)$ (including all triangle-cliques and all star-cliques) define distinct facets of $P_{mon}(n)$.*

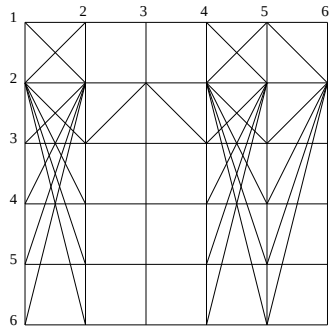

Figure 8: Support of a star-clique constraint equivalent to the 2-cycle in Fig. 2.

Consider again the star-clique constraint illustrated in Figure 8. By subtracting the equality from (2b) corresponding to vertex $(2, 2)$, one obtains another star-clique (the arcs entering $(2, 2)$ are flipped into arcs leaving $(2, 2)$). Combining similar flipping operations over vertices $(2, 4)$ and $(2, 5)$ one obtains 2^3 star-clique constraints, all defining the same facet of $P(6)$. A similar construction for general n proves Lemma 4.

We remark that Lemma 4 has no counterpart in the monotonization of the STSP and ATSP polytopes. For example, there are only pairs of subtour elimination constraints ($\sum_{(i,j)\in S} x_{i,j} \leq |S| - 1$ and $\sum_{(i,j)\in V\setminus S} x_{i,j} \leq |V| - |S| - 1$) that define distinct facets of the monotonized STSP polytope, but define the same facet of the original STSP polytope.

6 Branch-Cut-and-Price Algorithm

The new cuts could be possibly applied in a branch-and-cut algorithm over the PQ formulation. However, tigther and more practical formulations can be obtained by using a Dantzig-Wolfe decomposition, described as follows. Define $q^{t,l}_{i,j}$ as a binary coefficient indicating whether arc (i, j, t) appears in the l-th $0-T$

path, and λ_l as the positive variable associated to that path.

$$\text{Minimize} \quad \sum_{l=1}^{p} \Big(\sum_{(i,j,t)\in A} q_{i,j}^{t,l} c_{i,j}^{t} \Big) \lambda_l \tag{11a}$$

S.t.

$$\sum_{l=1}^{p} \Big(\sum_{(i,j,t)\in A} q_{i,j}^{t,l} \Big) \lambda_l = 1, \quad j = 1, \ldots, n \tag{11b}$$

$$\lambda \geq 0 \text{ and integer} \tag{11c}$$

The linear relaxation of this reformulation can be efficiently solved by column generation, since the pricing subproblem consists in finding shortest $0-T$ paths in (V, A). This can be done in $O(n^3)$ time by dynamic programming. Moreover, significantly stronger linear relaxations can be obtained by only pricing paths without r-cycles, for small values of r. Changing the dynamic programming procedure in order to avoid paths with 2-cycles is simple and only adds a small factor to the pricing time. On the other hand, pricing paths without larger r-cycles is much more complex and only practical for quite small values of r. Anyway, a fractional solution $\bar{\lambda}$ of (11) can be translated into a fractional solution $\bar{x}$ of (1) using the following identities:

$$\bar{x}_{i,j}^{t} = \sum_{l=1}^{p} q_{i,j}^{t,l} \bar{\lambda}_l, \qquad (i,j,t) \in A. \tag{12}$$

Cuts, like those presented in the previous sections, can then be separated, translated back to the space of the λ variables and added to the linear relaxation of (11). Embedding this column and cut generation scheme within a branch-and-bound method yields a Branch-Cut-and-Price (BCP) algorithm.

Bigras, Gamache and Savard [4] recently implemented a BCP algorithm for the TDTSP that also uses formulation (11), pricing paths without 4-cycles. They separate families of TSP cuts (using procedures from Concorde [1]) and also non-structured clique cuts obtained by explicitly building the incompatibility graph and looking for maximum weighted cliques in it (using the CLIQUER package [21]). In contrast, our BCP separates only the specific TDTSP cuts presented in this paper. The main elements of our BCP are described in the next subsections.

6.1 Separation Procedures

Five separation procedures were implemented:

- **r-cycle elimination AFCs.** The proposed separation procedure is based on the flow decomposition of a fractional solution into $0-T$ paths. In a BCP context this decomposition comes directly from the fractional solution of (11). For each path in the decomposition, all minimal r-cycles are identified and the corresponding inequality (definition 4) is checked.

- **Unitary AFCs.** The procedure first performs exact separation of a family of inequalities that are a weakening of the unitary AFCs. For every arc (i, j, t), $1 \leq t < n$, and for every set X containing node $(j, t+1)$, the following inequality is valid:

$$x_{i,j}^{t} \leq \sum_{(k,l,t') \in \delta^{+}(X), l \neq i, l \neq j} x_{k,l}^{t'}. \tag{13}$$

  For example, if $n = 6$ and arc $(1, 2, 1)$ and $X = \{(2, 2), (3, 3), (4, 3), (4, 4)\}$ are taken, the valid inequality is $x_{1,2}^{1} \leq x_{2,5}^{2} + x_{2,6}^{2} + x_{3,5}^{3} + x_{3,6}^{3} + x_{4,3}^{3} + x_{4,5}^{3} + x_{4,6}^{3} + x_{4,3}^{4} + x_{4,5}^{4} + x_{4,6}^{4}$. This is a weakening of the unitary AFC depicted in Fig. 4, the coefficient of variable $x_{4,3}^{4}$ is zero in that facet-defining inequality. Inequalities (13) can be separated in polynomial time: for a given arc (i, j, t) the best set X can be determined by finding a minimum cut separating $(j, t+1)$ from T in a graph that does not contain vertices (i, t') and (j, t') for $t > t + 1$. The arc capacities are set as equal to the fractional value of the corresponding variables. After the best X for a certain $E = \{(i, j, t)\}$ is found, inequality (5) is checked.

- **General AFCs.** For a fixed X we can find the set $E \subseteq \delta^{-}(X)$ leading to the most violated inequality (5) or show that no AFC is violated by solving a max-flow min-cut problem. This is done by setting a bipartite network where one side has one vertex for each arc in $\delta^{-}(X)$ and the other side has one vertex for each arc in $\delta^{+}(X)$. There is an arc joining each $e \in \delta^{-}(X)$ to each arc in $C(X, E)$. All those arcs receive infinity capacity. An additional source vertex s is linked to vertices $e \in \delta^{-}(X)$ by arcs with capacity equal to the fractional value of e. In a similar way, arcs $f \in \delta^{+}(X)$ are linked to a target vertex t by arcs with capacity equal to the fractional value of f. A violated AFC over X exists if and only if the max $s - t$ flow in that network has value strictly less than one.

  The heuristic separation of general AFCs applies that procedure on a number of rectangular candidate sets X. A rectangular set is defined by a set $S \subset N$ as $X(S) = \{(i, t) : i \in S, 1 \leq t \leq n\}$. We found that rectangular sets containing a number of r-cycles (for $r \leq 10$) in the flow decomposition of a fractional solution into $0-T$ paths are good candidates. The heuristic is also applied on the non-retangular sets X corresponding to the r-cycles found in the procedure that separated of r-cycle constraints.

- **Lifted Subtour Elimination Constraints.** For a fixed cardinality s, the set S with $|S| = s$ leading to the most violated LSEC (8) can be found by solving a Minimum Cut with Fixed Cardinality Problem, which is NP-hard. Nevertheless, we were able to build a quite practical separation of LSECS by solving a sequence of MIP models, one for each cardinality, for that problem. In order to avoid spending an excessive amount of time on those MIPs, a node limit is set. Therefore, the separation is not always exact.

- **Triangle Clique Constraints.**. An exact separation procedure with $O(n^4)$ time complexity (much faster in practice) was already proposed in [24].

Each round of cuts (a call to all the five separation procedures) introduces at most 350 violated cuts into (11). The first rounds are usually quite effective in improving the lower bounds, subsequent rounds being less effective. Therefore, we decided to perform at most 9 rounds of cuts in the root node and a single cut round in the remaining nodes.

6.2 Pricing Procedures

As already mentioned in [4], there is a significant improvement in the lower bounds provided by the relaxation of the basic formulation (11) when one only prices paths without r-cycles; the larger the r, the better. However, the best known algorithm for such a pricing has a worst case complexity of $\Omega(r!r^2n^3)$ [16]. The explosive increase in the complexity is directly related to the fact that the dynamic programming algorithm must keep up to $r!$ alternative suboptimal subpaths in its states, that may be used when a possible extension would create an r-cycle with better subpaths. As a BCP algorithm may need to perform thousands of calls to its pricing procedure in each node of the enumeration tree, taking $r > 4$ can make the code unbearably slow.

In order to efficiently use 5-cycle elimination in our pricing, a pricing heuristic was used. It consists in limiting the number of suboptimal subpaths kept in each state, so some valid extensions may be missed. The exact pricing is only called when the heuristic fails to find a path without 5-cycle with negative reduced cost or when one wants to determine a valid Lagrangean lower bound. The heuristic is quite effective. In most of the calls (90% being typical) it actually finds the same optimal path without 5-cycle that would be found by the exact pricing, but only taking a small fraction of time (5% being typical).

6.3 Fixing by Reduced Costs

We use the same dynamic programming procedure described in [26] to fix $x^t_{i,j}$ variables by reduced costs. The procedure is very effective. Even on the hardest instances it is typical to have more than 80% of the variables fixed to zero in the end of the root node. The fixing impacts directly in the complexity of the pricing, which is actually $O(r!r^2m)$, where m is the number of non-fixed arcs. The fixing may also yield some improvement on the lower bounds.

6.4 Dual Stabilization

The solution of the relaxation of formulation (11) by column generation is very prone to convergence problems. Some stabilization mechanism is absolutely necessary to mitigate that problem. We used the same dual stabilization described in [26]. The pricing is not performed with the current values of the dual

variables directly provided by the LPs, since they may oscillate wildly for many iterations before settling on meaningful values. The pricing is performed with the stabilized values that come from a linear combination of the current dual variables with the dual solution that provided the best Lagrangean lower bound so far. It is proved in [26] that this mechanism is sound: if a column with the most negative reduced cost with respect to the stabilized dual does not have a negative reduced cost with respect to the current dual, then the stabilized solution must yield a significantly better new Lagrangean bound.

In order to hot-start that stabilization, we perform a number of iterations of the Volume algorithm before performing the column generation in the root node. The Volume algorithm is calibrated in order to converge quickly to a reasonable dual solution, good enough for guiding the first iterations of the column generation, until it begins to "self-stabilize".

6.5 Branch Rule

Branching over individual variables $x^t_{i,j}$ would lead to highly unbalanced search trees. We preferred to branch over aggregated variables $x_{i,j} = \sum_t x^t_{i,j}$, choosing fractional values close to 0.6.

6.6 Primal Heuristic

We start the BCP by calling an iterated local search primal heuristic. The search performs moves (remove a vertex from the route and insert it in other place), swaps (swap pairs of vertices) and reversals (reverses the vertices in a subroute) until a local minimum is found. Then the current solution is perturbed and another search begins. This is repeated by n^2 iterations. This is not a state-of-the-art heuristic, but it is enough to find a good solution (often an optimal one) for helping the fixing by reduced costs. A few iterations of the local search are also performed along the BCP, starting from solutions obtained by rounding the current fractional solution at the end of each column generation.

7 Computational Results

The BCP algorithm was coded in C++ over the Coin-Bcp framework, version 1.2.3 [27], CPLEX 12.0 was the LP solver. All experiments were conducted on a single core of an Intel i5 CPU M430@2.27GHz. The machine has 8GB of RAM and uses a 64-bit version of the Linux OS.

Even though the proposed algorithm is devised for general TDTSPs, all our tests were performed in TDP instances derived from original TSP instances. In those instances, the cost of an arc (i, j, t) is defined as $(n - t + 1) \cdot d(i, j)$, where $d(i, j)$ is taken from the original distance matrix. This allows direct comparisons with a larger literature, as there are relatively few articles providing computational results for non-TDP instances. Those TDP instances are much harder

than their TSP counterparts - as far as we know, the only algorithm that was reported as solving instances with $n > 50$ is the combinatorial branch-and-bound proposed in 1993 by Fischetti, Laporte and Martello [6]. Comparisons with the results published in that paper would be meaningless, due to the disparity between machines after almost two decades. Happily, those authors kindly provided us with that code, so we could compare its performance with that of the proposed BCP on the same machine and on the same instances.

Table 1 presents computational results on the 22 instances from the TSPLIB ranging from 42 to 107 vertices. The columns **Gap CG** show the percent integrality gaps for only performing column generation with 3-cycle, 4-cycle and 5-cycle elimination, while columns **Gap +cuts** are the gaps after up to 9 rounds of cuts. It can be seen that the cuts are indeed effective, with 5-cycle elimination the average gap is reduced from 2.26% to 0.60%. As already mentioned, we decided to use 5-cycle elimination as the default option in the BCP. Column **Rem Arcs** gives the number of arcs that were not fixed by reduced costs in the end of the root node. Column **BCP Time** is the total time in seconds spent by the complete BCP algorithm, including the time took by primal heuristics. We set a time limit of 172,800 seconds (= 48 hours) of computing time; instances *rat99*, *kroD100* and *eil101* could not be solved by the BCP within that limit. Columns **Nds** and **Max Dep** are the total number of nodes explored and the maximum depth reached in the enumeration tree. Column **BB Time** is the total time in seconds spent by the branch-and-bound code by Fischetti et al. That code crashed on instances *pr76*, *gr96* and *pr107* due to numerical overflow. Instance *brazil58* could not be finished in 48 hours. For instances larger than 96 vertices we set a time limit of 24 hours, no such instance could be solved. Column **UB** gives the best known upper bounds, values in bold are proved to be optimal. The best known upper bounds for instances *rat99* and *eil101*, marked with a star, were obtained by other authors [19]. The upper bounds found by our BCP were 58,288 and 27,519, respectivelly.

Table 1: Results over all TSPLIB instances ranging from 42 to 107 vertices and comparison with the B&B by Fischetti et al.

	3-cycle el.		4-cycle el.		BCP with 5-cycle el.						B&B [6]		
Instance	Gap CG	Gap +cuts	Gap CG	Gap +cuts	Gap CG	Gap +cuts	Rem Arcs	BCP Time	Nds	Max Dep	BB Time	Nds	UB
dantzig42	2.47	0.18	1.37	0.00	0.49	0.00	0	28	1	0	3.6	20325	**12528**
swiss42	1.33	0.00	0.00	0.40	0.00	0.00	0	20	1	0	1.2	3945	**22327**
att48	2.61	0.00	1.99	0.00	1.62	0.00	0	103	1	0	4062	101M	**209320**
gr48	4.04	1.62	1.78	0.27	0.97	0.00	0	76	1	0	13.6	57249	**102378**
hk48	1.03	0.45	1.05	0.40	1.06	0.39	1535	124	3	0	16.5	91009	**247926**
eil51	1.19	0.00	0.28	0.00	0.00	0.00	0	33	1	0	92	526111	**10178**
berlin52	2.11	0.73	1.71	0.18	1.19	0.00	0	104	1	0	411	585347	**143721**
brazil58	11.07	0.95	8.17	0.36	5.32	0.00	0	633	1	0	>48hs	>537M	**512361**
st70	3.81	1.11	2.82	0.72	1.99	0.32	8339	2895	3	1	153	616223	**20557**
eil76	0.61	0.07	0.16	0.00	0.00	0.00	0	223	1	0	4327	11M	**17976**
pr76	3.92	2.47	2.66	1.85	2.00	1.51	48K	58045	221	10	-	-	**3455242**
gr96	6.87	2.78	4.36	1.76	2.64	1.03	70K	160250	392	11	-	-	**2097170**
rat99	2.35	1.71	1.69	1.33	1.60	1.31	163K	>48hs	>369	$\geq$ 18	>24hs	>156M	57986*
kroA100	6.56	3.15	3.49	1.55	1.58	0.97	94K	106336	411	10	>24hs	>82M	**983128**
kroB100	3.44	0.59	2.24	0.24	2.01	0.10	15K	7684	5	2	>24hs	>106M	**986008**
kroC100	7.83	2.63	4.09	1.10	3.15	0.66	63K	39559	73	6	>24hs	>181M	**961324**
kroD100	8.54	4.41	7.82	3.71	7.82	3.75	402K	>48hs	>172	$\geq$ 18	>24hs	>33M	976965
kroE100	5.94	2.56	2.70	1.09	2.30	0.84	69K	117965	217	9	>24hs	>138M	**971266**
rd100	5.18	1.51	5.18	0.45	1.00	0.35	0	18582	19	4	>24hs	>324M	**340047**
eil101	3.43	2.56	2.94	2.16	2.80	2.00	267K	>48hs	>76	$\geq$ 21	>24hs	>137M	27513*
lin105	5.11	0.48	3.35	0.00	2.84	0.00	0	6317	1	0	>24hs	>125M	**603910**
pr107	7.92	0.00	7.33	0.00	7.33	0.00	0	9947	1	0	-	-	**2026626**
Avg.	4.42	1.36	3.05	0.80	2.26	0.60							

Some remarks about those results:

- The fast computation of reasonably good lower bounds (gaps of 10% are typical in the root node) used in the branch-and-bound of Fischetti et al. makes it very effective on smaller instances. However, as instances get larger, the much slower computation of stronger lower bounds by cut and column generation pays, and gives our code a clear advantage.
- While the aggressive 5-cycle elimination used in the pricing is usually helpful for the BCP performance, cut separation is essential. For example, instance *brazil58* can be solved by a BCP with only 3-cycle elimination in a few hours. However, a code with 5-cycle elimination but no cuts (so the BCP is reduced to a branch-and-price algorithm) can not solve that instance in reasonable time, the enumeration tree goes below depth 23.
- Cut separation accounts for less than 10% of the computation time in all instances, the bulk of the time is spent by the pricing and by LP solving.

Méndez-Díaz et al. [18], Bigras et al. [4], Miranda Bront et al. [20], Godinho et al. [9] are recent works providing good computational results on TDP instances. This last work presents interesting extended formulations with $\Theta(n^4)$ variables and $\Theta(n^3)$ constraints that obtained very small gaps (usually zero) on several TDP instances with up to 50 vertices. However, the large size of the formulations makes the approach too time-consuming. We compare our proposed BCP with the BCP described in [4], that still has the best published results for an LP based method. Table 2 reports the comparison for all the TDP instances used in [4]. Their times were obtained in an Intel Pentium 4 3.4 GHz machine. It can be seen that our BCP takes much less nodes than the BCP by Bigras et al. This is a clear indication that the new facet-defining TDTSP cuts perform significantly better than cuts borrowed from the TSP or from the independent set problem. We also run the code by Fischetti et al., which solved almost all those instances in a few seconds but could not solve the small instance *rbg031a* within 24 hours. This peculiar behavior can be explained by the fact that the *rbg* instances are very different from the other instances: their distance matrices are asymmetric, do not satisfy the triangle inequality and have many zero entries. In those cases, the combinatorial bounds are very poor and the branch-and-bound approach fails.

The TDTSP families of cuts proposed in this article, strong from a polyhedral point of view, appear to be also strong in practice. They are able to reduce significantly the integrality gaps, allowing the solution of several TDP instances with up to 107 vertices. However, solving some harder instances by a BCP approach may require further work, perhaps by finding new families of cuts, perhaps by improving the separation of some known families of cuts. Anyway, we believe that the polyhedral theory can play an important role in improving algorithms for the TDTSP, as already happened on the classical TSP.

Table 2: Results over the TDP instances used by Bigras et al.

	BCP with 5-cycle el.				BCP [4]		B&B [6]		
Instance	Gap CG	Gap +cuts	BCP Time	Nds	BCP Time	Nds	BB Time	Nds	UB
gr17	0.00	0.00	2.1	1	3	1	0.03	446	**12994**
gr21	0.00	0.00	4.3	1	10	1	0.03	394	**24435**
gr24	0.00	0.00	4.6	1	15	1	0.05	873	**13795**
bayg29	0.87	0.00	15	1	76	16	0.62	5535	**22230**
bays29	0.36	0.00	7.7	1	191	51	0.64	5194	**26862**
rbg016a	0.00	0.00	2.8	1	15	1	1.0	43210	**744**
rbg031a	0.00	0.00	23	1	90	1	>24hs	>1121M	**2476**
rbg050b	0.35	0.22	2068	61	37342	390	>24hs	>399M	**5497**
dTSP40.0	2.66	0.82	670	13	6473	377	3.9	27978	**10311**
dTSP40.1	2.71	0.00	82	1	1452	50	0.96	3696	**9807**
dTSP40.2	1.53	0.00	75	1	1068	28	5.2	22947	**9525**
dTSP40.3	0.00	0.00	36	1	629	24	6.8	18378	**9156**
dTSP40.4	0.00	0.00	31	1	299	4	0.71	3396	**9079**
dTSP50.0	0.00	0.00	61	1	1364	5	88	276630	**12680**
dTSP50.1	1.03	0.00	275	1	56240	571	16	52571	**12853**
dTSP50.2	0.17	0.00	78	1	3668	8	11	32906	**12357**
dTSP50.3	1.61	0.00	138	1	47771	541	24	48744	**12722**
dTSP50.4	2.30	0.00	819	1	109586	1514	22	40094	**13289**
Avg.	0.75	0.06	244.0	5.0	14794	199.1			

References

[1] Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On The Solution Of Traveling Salesman Problems. Documenta Mathematica, Extra Volume ICM 3, 645-646 (1998)

[2] Balas, E., Carr, R., Fischetti, M., Simonetti, N.: New Facets of the STS Polytope Generated from Known Facets of the ATS Polytope. Discrete Optimization 3, 3–19 (2006)

[3] Balas, E., Fischetti M.: Polyhedral theory for the ATSP. In Gutin, G., Punnen, A. (eds.) The Traveling Salesman Problem and Its Variations, pp. 117–168. Kluwer, (2002)

[4] Bigras, L.-Ph., Gamache, M., Savard, G.: The Time-Dependent Traveling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Time. Discrete Optimization 5, 685–699 (2008)

[5] Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale Traveling Salesman Problem. Operations Research 2, 393–410 (1954)

[6] Fischetti, M., Laporte, G., Martello, S.: The Delivery Man Problem and Cumulative Matroids. Operations Research 41, 1055–1064 (1993)

[7] Fox, K., Gavish, B., Graves, S.: An n-Constraint Formulation of the (Time Dependent) Traveling Salesman Problem. Operations Research 28, 1018–102 (1980)

[8] Gale, D.: A theorem of flows in networks. Pacific Journal of Mathematics 7, 1073–1082 (1957)

[9] Godinho, M.T., Gouveia, L., Pesneau, P.: Natural and Extended formulations for the Time- Dependent Travelling Salesman Problem, CIO Report8/2010, Lisbon (2010)

[10] Gouveia, L., Voss, S.: A Classification of formulations for the (time-dependent) traveling salesman problem. European Journal of Operations Research 83, 69–82 (1995)

[11] Gouveia, L., Simonetti, L., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. Mathematical Programming, Online first (2009)

[12] Groetschel, M., Padberg, M.: On the Symmetric Traveling Salesman Problem II: Lifing Theorems and Facets. Mathematical Programming 16, 281–302 (1979)

[13] Groetschel, M., Padberg, M.: Polyhedral theory. In Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.) The Traveling Salesman Problem, pp. 251–305. Wiley (1985)

[14] Hall, P.,: On representatives of subsets. Journal of London Mathematical Society 10, 26–30 (1935)

[15] Hoffman, A.: Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. Proceedings of Symposium in Applied Mathematics 10, 113–128 (1960)

[16] Irnich, S., Villeneuve, D.: The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. INFORMS Journal on Computing 18, 391–406 (2006)

[17] Lucena, A.: Time-Dependent Traveling Salesman Problem - The Deliveryman Case. Networks 20, 753–763 (1990)

[18] Méndez-Díaz, I., Zabala, P., Lucena, A.: A New Formulation for the Traveling Deliveryman Problem. Discrete Applied Mathematics 156, 3233–3237 (2008)

[19] Melo, M., Subramanian, A,: Personal communication (2010)

[20] Miranda Bront, J.J., Méndez-Díaz, I., Zabala, P.: An Integer Programming Approach for the Time Dependent Traveling Saleman Problem, Electronic Notes in Discrete Mathematics 36, 351–358 (2010).

[21] Niskanen, S., Ostergard, P.R.J.: Cliquer users guide. Helsinki University of Technology, Communications Laboratory, Technical report 48 (2003)

[22] Padberg, M.: On the facial structure of set packing polyhedra. Mathematical Programming 5, 199–215 (1973)

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

[24] Pessoa, A., Poggi de Aragão, M., Uchoa, E.: Robust Branch-Cut-and-Price Algorithms for Vehicle Routing Problems. In Golden, B., Raghavan, S., Wasil, E. (eds.) The Vehicle Routing Problem: Latest Advances and New Challenges, pp. 297–326. Springer, New York (2008)

[25] Pessoa, A., Uchoa, E., Poggi de Aragão, M.: A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. Networks 54, 167-177 (2009)

[26] Pessoa, A., Uchoa, E. , Poggi de Aragão, M., Freitas, R.: Exact algorithm over an arc-time indexed formulations for parallel machine scheduling problems. Mathematical Programming Computation 2, 259–290 (2010)

[27] Ralphs, T.K., Ladányi, L.: COIN/BCP User's Manual. Available at www.coin-or.org/Presentations/bcp-man.pdf (2001)

[28] Vajda, S.: Mathematical Programming, Addison-Wesley, (1961)

[29] Vander Wiel, R.J., Sahinidis, N.V.: An Exact Solution Approach for the time-dependent traveling salesman problem. Naval Research Logistics 43, 797–820 (1996)

Appendix

Proof of Theorem 9

Proof that (9) **defines a facet of** $P(n)$

We prove by induction on n and on $|S|$ that there are $n(n-1)(n-2)$ linearly independent (LI) s-paths that satisfy (9) as equality. We first establish that (9) defines a facet of $P(n)$ for $|S| = 3$ and $n \geq 6$. We can assume without loss of generality that $S = \{1, 2, 3\}$. The induction basis is obtained computationally for $n = 6$ and $|S| = 3$.

The induction asserts there are $n(n-1)(n-2)$ LI s-paths in $G(n)$ that satisfy (9) as equality. We show below how to construct $(n+1)n(n-1)$ LI paths in $G(n+1)$ that satisfy (9) as equality. We divide the set of LI paths to be constructed into 5 cases.

1. s-paths contained in $G(n)$ except for the last arc incident to node $(n+1, n+1)$;

2. s-paths that begin at node $(n+1, 1)$ and whose last arc (i, j, n) is not of the form $i \notin S$ and $j \in S$;

3. s-paths that contain node $(n+1, t)$, where $1 < t \leq |S|$, and whose last arc (i, j, n) does not satisfy $i \notin S$ and $j \in S$;

4. s-paths that contain node $(n+1, t)$, where $|S| < t \leq n$, and whose last arc (i, j, n) does not satisfy $i \notin S$ and $j \in S$;

5. s-paths whose last arc (i, j, n) satisfies $i \notin S$ and $j \in S$.

For case 1, we note that the $n(n-1)(n-2)$ LI s-paths in $G(n)$ can be extended in the same way as in the proof of Theorem 3. If an s-path in $G(n)$ ends at an arc $(i, j, n-1)$, with $j \in S$, then this arc contributes to the left-hand side of 9 for $G(n)$ but not for $G(n+1)$. In this case, the arc $(j, n+1, n)$ appended to the s-path replaces (i, j, n) in the left-hand side of (9). On the other hand, if an s-path in $G(n)$ ends at an arc $(i, j, n-1)$, with $j \notin S$, then this arc does not contribute to the left-hand side of (9) for $G(n)$. In this case, the arc $(j, n+1, n)$ appended to the s-path also does not contribute to the left-hand side of (9). Hence, all these LI paths in $G(n+1)$ satisfy (9) as equality. Like in the proof of Theorem 3, we need to find an additional set of $3n^2 - 3n$ LI paths and there are still $3n^2 - 2n$ unused new arcs (so we have n "spare" arcs left).

For case 2, there are $n^2 - (n - |S|)|S|$ new arcs that can be used. We will construct $n^2 - (n - |S|)|S| - 1$ s-paths, each one using one arc not used before, thus consuming one "spare" arc. To proceed, we choose $n^2 - (n - |S|)|S| - 1$ pairs of arcs, each pair containing one arc in layer 1 and one arc in layer n. These arc pairs are chosen in the same way as in the proof of Theorem 3, but leaving out the pairs that contain a forbidden arc (i, j, n) with $i \notin S$ and $j \in S$. Next, for each selected pair of arcs, we build an s-path that contains these two arcs and satisfies (9) with equality. We do as follows: if $k \notin S$ and $i \in S$ (and the arcs $(n+1, k, 1)$ and (i, j, n) are used), then complete the s-path in such a way that it enters the set S only once (either in layer $n-2$ or $n-3$ depending whether $j \in S$ or not). If $i, j, k \notin S$, then complete the s-path in such a way that it enters and leaves the set S only once, between the layers 1 and n. If $k \in S$ and $i \in S$, then complete the s-path in such a way that it leaves the set S in the layer 2 and enters it only once more (either in layer $n-1$ or $n-2$ depending whether $j \in S$ or not). If $k \in S$ and $i, j \notin S$, then complete the s-path in such a way that it leaves the set S only in the layer 4, after all node indices of S have been used.

For case 3, for any pair of node indices i, j, with $i \neq j$, one can always find an s-path that uses the arcs $(i, n+1, t-1)$ and $(n+1, j, t)$ and satisfies (9) with equality as follows. Before the layer t, complete the s-path in any valid way. After the layer t, complete the s-path in such a way that it enters the set S only once and before the layer n. For that, it must leave the set S only after all its node indices have already been visited. Then, one can choose the pairs of arcs $(i, n+1, t-1)$ and $(n+1, j, t)$ in the same way as in the proof of Theorem 3.

For case 4, we choose arc pairs $(i, n+1, t-1)$ and $(n+1, j, t)$ in a similar way as in the proof of Theorem 3. We first fix the incoming arc $(n, n+1, t-1)$. This arc can be combined with an outgoing arc of the form $(n+1, j, t)$ for $j = 1, \ldots, n-1$ to be part of an s-path of $G(n+1)$. Similarly, outgoing arc $(n+1, n-1, t)$ can be combined with incoming arcs of the form $(i, n+1, t-1)$ for $i = 1, \ldots, n-2$ to produce $n-2$ s-paths. Pairing arc $(1, n+1, t-1)$ with $(n+1, n, t)$ yields one more path. Finally, we combine arcs $(n-1, n+1, t-1)$ and $(n+1, 1, t)$ to obtain the last s-path. Note that we never combine a pair of arcs where both endpoints $i, j \in S$. For each chosen pair of arcs, we build an LI s-path that use both arcs $(i, n+1, t-1)$ and $(n+1, j, t)$ and satisfy (9) with equality as follows. If $j \notin S$ and either $i \in S$ or $t = n$, then complete the s-path before the layer t in such a way that all node indices of S are visited. Then, after the layer t, complete the s-path in any valid way (e.g. without entering S again). If $j \in S$ or both $i \notin S$ and $t < n$, then complete the s-path before the layer t in such a way that exactly $|S|-1$ node indices of S are visited in the layers $1, \ldots, |S|-1$. In this case, leave the set S in the layer $|S|-1$ and enter it only after visiting the node $(n+1, t)$, to visit the only remaining node index in S.

For case 5, we build the remaining $(n-|S|)|S|$ LI s-paths by visiting exactly $|S|-1$ node indices of S in the layers $1, \ldots, |S|-1$, leaving the set S in the layer $|S|-1$, and entering it only in the arc layer n, through each chosen arc $(i, j, n+1)$, to visit the only remaining node index in S.

Having established that (9) defines a facet for $|S| = 3$ and $n \geq 6$, we now prove by induction that the result also holds for $|S| \geq 3$. Assume without loss of generality that $S = \{n-|S|+1, \ldots, n\}$. Now, by Lemma 8 it is equivalent to consider instead constraint (8).

Let $n \geq 6$ and $|S| \geq 3$. By induction, there are $n(n-1)(n-2)$ LI s-paths in $G(n)$ that satisfy (8) as equality. We show how to construct $(n+1)n(n-1)$ LI paths in $G(n+1)$ that satisfy (8) as equality, where S is replaced by $S' = S \cup \{n+1\}$.

We divide the set of LI paths we construct into 4 cases.

1. s-paths contained in $G(n)$ except for the last arc incident to node $(n+1, n+1)$;
2. s-paths that begin at node $(n+1, 1)$;
3. s-paths that contain node $(n+1, t)$, where $1 < t \leq n-|S|$;
4. s-paths that contain node $(n+1, t)$, where $n-|S| < t \leq n$.

For case 1, we note that the $n(n-1)(n-2)$ LI s-paths in $G(n)$ can be extended in the same way as in the proof of Theorem 3. This is true because the coefficients of all arcs of $G(n)$ in (8) do not change when extending the graph to $G(n+1)$ and replacing the set S by S'. Moreover, all arcs incident to node $(n+1, n+1)$ have null coefficients in (8).

For case 2, we note that for each pair of arcs $(n+1, j, 1)$ and (k, l, n), we can always find an s-path that does not enter the set S before the arc layer $n-1$.

For that, one must complete the s-path that starts with the arc $(n+1, j, 1)$ by visiting all the vertices in S (except k and l) before leaving this set. Such s-path satisfies (8) as equality. Hence, the pairs of arcs can be chosen as in proof of Theorem 3. This generates $n^2 - 1$ s-paths.

For cases 3 and 4, we note that for each pair of arcs $(i, n+1, t-1)$ and $(n+1, j, t)$, we can always find an s-path that satisfies (8) as equality. For case 3, the s-path can visit as many vertices out of S as possible before the node layer t. After the node layer t, it must leave S as soon as possible, and enter S again, if necessary, only after the arc layer $n - |S|$. For case 4, we have two subcases: $i \in S$ and $i \notin S$. In both subcases, the s-path can leave the set S, after the node layer t as soon as possible. Before the node layer t, If $i \in S$, the s-path can enter the set S only once, exactly $|S|$ arc layers before it leaves this set. If $i \notin S$, the s-path can visit in the first layers all vertices of S not visited after the node layer t. Then, it can leave the set S and enter it again exactly in the arc layer $t-1$. In both cases 3 and 4, The pairs of arcs are chosen as in proof of Theorem 3, which generates $2n-1$ s-paths for each t.

This gives the necessary $(n+1)n(n-1)$ LI s-paths, completing this proof. ∎

Proof that the facets defined by (8) **and** (9) **are different**

We start by showing that each facet defined by (8) is different. Let $S \neq S'$. Assume without loss of generality that $|S'| \leq |S|$. Notice that this implies that $S \not\subseteq S'$.

If $|S' \cap S| \geq 2$ then, pick an s-path that goes through vertex $(i, 1)$ for $i \in S' \cap S$ then goes to vertex $(j, 2)$ for $j \in S \setminus S'$ and then goes to vertex $(k, 3)$ for $k \in (S' \cap S) \setminus \{i\}$ and finally goes through all the remaining vertices in S at positions $4, \ldots, |S|$. Such an s-path satisfies (8) for S at equality. Moreover it enters S' once at position 1 and once at position 3. Since $|S'| \leq n-3$, then $n - |S'| \geq 3$ and so this s-path does not satisfy (8) for S' at equality.

So we may assume that $|S' \cap S| \leq 1$. Then consider an s-path that goes through vertex $(k, 1)$ for $k \in S'$ (if there is a vertex in $S' \cap S$, then let k be that vertex), then goes to vertex $(j, 2)$ for $j \in S \setminus S'$ and then goes through all vertices in $S' \setminus \{k\}$ from positions 3 to $3 + |S'| - |S \cap S'| - 1$. Notice that at this point we have visited $|S' \cap S| + 1$ vertices in S and $|S'| - |S \cap S'|$ vertices outside S. Therefore there are $n - |S'| - 1$ vertices remaining to be visited, with $|S| - (|S \cap S'| + 1)$ of them in S. So then let the s-path go through all remaining vertices in $N \setminus S$. Note that there are a total of $t' = n - |S'| - 1 - (|S| - (|S \cap S'| + 1)) = n - |S'| - |S| + |S \cap S'|$ vertices not visited in $N \setminus S$. Therefore, these vertices will be visited from positions $3 + |S'| - |S \cap S'|$ to $3 + |S'| - |S \cap S'| + t' - 1 = 3 + |S'| - |S \cap S'| + n - |S'| - |S| + |S \cap S'| - 1 = n - |S| + 2$. Finally, the s-path will visit all remaining vertices in S.

Notice that this s-path enters S' at least twice before position 3, and since $|S'| \leq n-3$, we have $n - |S'| \geq 3$ so (8) is not satisfied at equality for S'. Also, this s-path enters S exactly once from position 1 until position $n - |S| + 2$. Therefore, the next time it enters S it does so by using an arc in a layer $t > n - |S|$. So this s-path satisfies (8) at equality.

The fact that each constraint (9) defines a distinct facet of $P(n)$ follows trivially. Now all that is left is to prove that (8) and (9) define distinct facets.

Consider (8) for a set S and (9) for a set S'.

Assume without loss of generality that $\{1,2,3\} \subseteq S$. We may also assume that $4 \notin S$.

But then, consider an s-path that goes through node (1,1), (4,2), (2,3) and then through all remaining nodes in $S' \setminus \{1,2,4\}$ at positions 4 to $3+|S' \setminus \{1,2,4\}|$. This s-path does not satisfy (8) at equality. However, it leaves S' only once at positions $t \geq |S'|$ since $3 + |S' \setminus \{1,2,4\}| \geq 3 + |S'| - 3 = |S'|$ and hence satisfies (9) at equality. ∎

Proof of Lemma 11

Case $|\{i,j\} \cap S| \leq 1$. We can assume without loss of generality that $s < t$, thus either $t = s+1$ or $s = 1$ and $t = n-1$. Since $n \geq 7$ there are at least 3 layers of $G(n)$ with arcs in $\mathcal{I}(S,p)$. Thus, at least one of these layers has an index r satisfying $|r-s| > 1$ and $|r-t| > 1$. Choose $c = (q_1, q_2, r) \in \mathcal{I}(S,p)$ such that $q_1, q_2 \in S \setminus \{i,j\}$. Note that one can find an s-path containing arcs a, b and c.

Case $i,j \in S$. Note that in this case it must be that $s = 1$ and $t = n-1$, since otherwise either $k \in S$ or $l \in S$, contradicting the hypothesis of this lemma. If $a \in \mathcal{I}(S,p)$, we are done. Alternatively, $a \notin \mathcal{I}(S,p)$, which implies that $p = 1$, and hence $\mathcal{I}(S,p)$ does not contain arcs from layer 1. Then, since $\mathcal{I}(S,p)$ is an alternating independent set, it must contain the set of arcs $A(S,2)$ from layer 2. Let k be the unique element in $S \setminus \{i,j\}$ and let $c = (j,k,2) \in \mathcal{I}(S,p)$. Again, one can find an s-path containing arcs a, b and c. ∎

Proof of Theorem 12

We need to find $n(n-1)(n-2)$ linearly independent s-paths that satisfy (10) as equality. Without loss of generality, we may assume $S = \{4,5,6\}$. Our proof follows by induction and follows closely the proof of Theorem 3. For $n = 7$, the two types of alternating independent sets to consider are cases 1 and 2, which by symmetry are equivalent. To establish the basis for the induction, one can prove computationally that inequality (10) for case 1 defines a facet by generating the incidence vectors of all s-paths that satisfy it as equation and using Gaussian elimination to verify that 210 LI paths can be chosen from this set.

Let $n \geq 7$. By induction, there are $n(n-1)(n-2)$ LI s-paths in $G(n)$ that satisfy (10) as equality. We show how to construct $(n+1)n(n-1)$ LI paths in $G(n+1)$ that satisfy (10) as equality. We divide the set of LI paths to be generated into 3 cases.

1. s-paths contained in $G(n)$ except for the last arc incident to node $(n+1, n+1)$;
2. s-paths that begin at node $(n+1, 1)$;
3. s-paths that contain node $(n+1, t)$, for $2 \leq t \leq n$.

Case 1. These paths are obtained by extending the $n(n-1)(n-2)$ paths in $G(n)$ that satisfy (10) as equality by appending an arc incident to node $(n+1, n+1)$. Cases 2 and 3 are constructed in the same fashion as in Theorem 3. By Lemma 11, these s-paths can be constructed to satisfy (10) as equality. ∎