The Time Dependent Traveling Salesman Problem: Polyhedra and Branch-Cut-and-Price Algorithm

Hernán Abeledo[1], Ricardo Fukasawa[2], Artur Pessoa[3], and Eduardo Uchoa[3]

[1] Department of Engineering Management and Systems Engineering, The George Washington University, Washington, DC, USA
[2] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada
[3] Departamento de Engenharia de Produção, Universidade Federal Fluminense, Niterói, RJ, Brazil

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 60 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. In particular, we also show that some facet defining cuts for the usual Asymmetric TSP formulation define low dimensional faces of the TDTSP formulation and give a way to lift them. We obtain good computational results with a branch-cut-and-price algorithm using the new cuts, solving several instances of reasonable size at the root node.*

Key words: Traveling salesman problem, integer programming, branch-cut-and-price

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 [1], used earlier in [2] 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. These include routing problems like the Traveling Deliveryman Problem (TDP), known also as the minimum latency problem, and scheduling problems such as the $1|s_{ij}|\sum C_j$.

- The formulation in [1], 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 [3] (including "nasty" cases, like the heterogenous fleet VRP [4]) and also complex single and multi-machine scheduling problems [5]. 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 [6] 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, [7] derived new Symmetric TSP (STSP) facets from known Asymmetric TSP (ATSP) facets. Similarly, [8] 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 [9] and for the ATSP in [6]. Formulations for the TDTSP have been proposed or studied in [1, 10–13]. Exact algorithms for the TDTSP are presented in [10, 14, 15] and, for the special case of the TDP, in [16–18]. Different heuristic methods for the TDTSP have been proposed in [13, 19]. The study of the TDP polytope was initiated in [18]. As far as we know, ours is the first investigation of the TDTSP polytope.

This paper is organized as follows. The TDTSP polytope is defined in Section 2, where its dimension is also established. 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 inequalities were already introduced in the VRP context [3], but now we show that some, and perhaps all, define facets of the TDTSP polytope. Finally, Section 6 presents a branch-cut-and-price algorithm for the TDTSP, separating the newly proposed inequalities. Due to lack of space, all proofs are included in the Appendix.

2 Preliminaries

Let $N = \{1, 2, \dots n\}$ 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 $(v_t : t \in N, v_i \neq v_j$ for $i \neq j)$, corresponds to a permutation of the elements of N, will be called an *s-path* . A $0-T$ path of (V, A) will be 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 $K(N_0)$.

Picard and Queyranne [1] formulated the TDTSP over (V, A) as a linear integer program with the following set of constraints, where variable $x^t_{i,j}$ indicates if arc (i, j, t) is used and N_i denotes $N \setminus \{i\}$.

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

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

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

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

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

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

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

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 $G(n)$.

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

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

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

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

Lemma 2 follows from Lemma 1. Also, the removal of any single equation from (2), such as (2a), yields a full rank system of equations.

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

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.

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

3 Admissible Flow Constraints

Let $p = (0, v_1, v_2, \dots, v_n, T)$ be a $0 - T$ path in (V, A). We define a r-cycle in p as a subpath $(v_i, \dots, v_{i+r})$ 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.

$$x^t_{i,j} \leq \sum_{k \in N \setminus \{i,j\}} x^{t+1}_{j,k}, \qquad (i, j, t) \in A, 1 \leq t \leq n-2. \tag{3}$$

Theorem 2. *If $n \geq 6$, then each constraint of (3) defines a facet of $P(n)$.*

The following lemma relies on a characterization of feasible network flow problems obtained by Gale [20] and Hoffman [21] which, if applied to balanced transportation problems on incomplete bipartite graphs, yields a generalization of Hall's marriage theorem [22].

Lemma 3. *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 $|R| = 2$. Then the transportation problem is feasible if and only if $b(v) \leq b(N(v))$ for each supply node $v \in S$.*

Theorem 3. *Let $x \in R_+^A$ satisfy constraints (1a, 1b, 1c, 1d) and (3). Then x can be decomposed into flows along $0 - T$ paths such that none of them contains a 2-cycle.*

Inequalities (3) may be aptly called *2-cycle elimination constraints*. The 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 2. *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):*

$$\sum_{e \in E} x_e \leq \sum_{f \in C(X,E)} x_f \tag{4}$$

Definition 3. *Let $((i, t), (u_1, t+1), \dots, (u_{r-1}, t+r-1), (i, t+r))$ be a minimal r-cycle in G. The AFCs where $X = \{(u_1, t+1), \dots, (u_{r-1}, t+r-1)\}$ and $E = \{(i, u_1, t)\}$ are called* r-cycle elimination constraints.

Computational experiments and partial results not stated here support the conjecture that all r-cycle elimination constraints are facet-defining. The more general AFCs are usually not facet-defining, but are still interesting because: (i) there are AFCs that are not dominated by r-cycle elimination constraints; (ii) for a fixed set X they can be separated (finding the best set E) in polynomial time as a min-cut problem; and (iii) they proved to be very useful in practice.

4 Lifted Subtour Elimination Constraints

The classical Subtour Elimination Constraints (SECs) [23] are known to define facets of the STSP polytope [24] and also of the ATSP polytope [9]. SEC inequalities can be expressed in terms of the TDTSP variables as follows:

$$\sum_{j \in S} x_{0,j}^0 + \sum_{t=1}^{n-1} \sum_{i \notin S} \sum_{j \in S} x_{i,j}^t \geq 1, \qquad S \subset N, |S| > 1. \tag{5}$$

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

$$\sum_{i \in S} \sum_{j \in N_j} x_{i,j}^1 + \sum_{t=1}^{n-1} \sum_{i \notin S} \sum_{j \in S} x_{i,j}^t \geq 1, \qquad S \subset N, |S| > 1. \tag{6}$$

SECs may define quite low-dimensional faces of the TDTSP polytope. Lifting them provides a much stronger family of valid TDTSP inequalities that we call *Lifted Subtour Elimination Constraints* (LSECs):

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

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.

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

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 (8) can be viewed as a constraint of type (7) for the s-paths of the graph $\bar{G}(n)$, using the same set S in both inequalities. We can conclude that inequality (7) for a fixed set S defines a facet of P_n if and only if inequality (8), for the same set S, defines a facet of P_n.

Lemma 4. *Inequality* (7) *defines a facet of* P_n *if and only if inequality* (8) *also does.*

Our main result for this section establishes that lifted subtour elimination constraints define facets. Its proof relies on a double induction.

Theorem 4. *If* $n \geq 6$ *and* $3 \leq |S| \leq n - 3$, *then constraint* (7) *defines a facet of* $P(n)$.

5 Triangle Clique Constraints

A well-known way of deriving strong cuts for binary integer programs is by analyzing their variable incompatibility graph. 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 [25], 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 $\{(i, j, t), (j, k, t+1)\}$. An independent set $\mathcal{I} \subset \mathcal{V}$ is a maximal set of vertices in $\mathcal{G}$ which are all pairwise incompatible. It is clear that the following inequality is valid:

$$\sum_{(i,j,t)\in\mathcal{I}} x^t_{i,j} \leq 1 \tag{9}$$

These constraints were proposed in [3] for a more general setting and were named *Triangle Clique Constraints.* In particular, [3] describes an efficient pseudo-polynomial separation procedure (which is polynomial when restricted to the TDTSP) and demonstrates the usefulness of these constraints for solving heterogeneous vehicle routing problems with a branch-cut-and-price algorithm. We prove here that constraints (9) 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 $G(n)$, where each subgraph is isomorphic to $(S, A(S))$. Let $A(S, t) = \{(i, j, t) : (i, j) \in A(S)\}$, we call the following four cases of independence sets *alternating.*

1. For n even, $\mathcal{I} = \bigcup_{k=1}^{n/2} A(S, 2k - 1)$.
2. For n even, $\mathcal{I} = \bigcup_{k=1}^{(n/2)-1} A(S, 2k)$.
3. For n odd, $\mathcal{I} = \bigcup_{k=1}^{(n-1)/2} A(S, 2k - 1)$.
4. For n odd, $\mathcal{I} = \bigcup_{k=1}^{(n-1)/2} A(S, 2k)$.

Lemma 5. *Let $n \geq 7$. Let $S \subset N$ and $|S| = 3$. Let $\mathcal{I}$ be an alternating independence set corresponding to S. 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 $\mathcal{I}$.*

Theorem 5. *If $n \geq 7$ and $\mathcal{I} \subset \mathcal{V}$ is an alternating independence set, then (9) defines a facet of $P(n)$.*

6 Branch-cut-and-price Algorithm

The main drawback of working directly with the PQ formulation is its large size, $O(n^3)$ variables and $O(n^2)$ constraints. However, an equivalent reformulation in terms of $O-T$ paths in (V, A) can be handled in an effective way. Number all possible $O-T$ paths from 1 to p. Define $q_{i,j}^{t,l}$ as a binary coefficient indicating whether arc (i, j, t) appears in l-th $O-T$ path, and λ_l as the positive variable associated to that path.

$$\text{Minimize} \sum_{l=1}^{p} \Big(\sum_{(i,j,t)\in A} q_{i,j}^{t,l} c_{i,j}^{t} \Big) \lambda_l \tag{10a}$$

$$\text{S.t.}$$

$$\sum_{l=1}^{p} \Big(\sum_{(i,j,t)\in A} q_{i,j}^{t,l} \Big) \lambda_l = 1 \; j = 1, \ldots, n \tag{10b}$$

$$\lambda \geq 0 \text{ and integer} \tag{10c}$$

The linear relaxation of this formulation can be efficiently solved by column generation, since the pricing subproblem consists in finding shortest $O-T$ paths in (V, A). This can be done in $O(n^3)$ time by dynamic programming. 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. The best known algorithm has a complexity of $O(r!r^2n^3)$ [26]. Usually, this is only practical for $r \leq 4$.

A fractional solution of (10) can be translated into a fractional solution of (1). 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 (10). 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 [14] recently implemented a BCP algorithm for the TDTSP that also uses formulation (10), pricing paths without 4-cycles. The main difference between their BCP and the one presented here lies in the cutting part. They separate families of TSP cuts (using procedures from Concorde [27]) 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 [28]). In contrast, our BCP separates only the specific TDTSP cuts presented in the previous section as follows:

– The proposed AFC separation is based on the flow decomposition of a fractional solution into $O-T$ paths. In a BCP context this decomposition comes directly from the fractional solution of (10). For each path in the decomposition, all minimal r-cycles are identified. For an r-cycle $((i, t), (u_1, t+1), \ldots, (u_{r-1}, t+r-1), (i, t+r))$, we try to separate AFCs in three different ways:

1. We check if the r-cycle elimination AFC for $X = \{(u_1, t+1), \ldots, (u_{r-1}, t+r-1)\}$ and $E = \{(i, u_1, t)\}$ is violated.
2. As mentioned in Section 3, for a fixed X we can find the set $E \subseteq \delta^-(X)$ leading to the most violated AFC 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. It can checked that a violated AFC over X exists only and only if the max $s-t$ flow in that network has value strictly lesser than one. The second AFC separation applies this procedure to set $X = \{(u_1, t+1), \ldots, (u_{r-1}, t+r-1)\}$.
3. The third AFC separation applies the above procedure to set $X = \{(v, t) : v \in \{u_1, \ldots, u_{r-1}\}, t = 1, \ldots, n\}$.

– We do not know if LSECs can be separated in polynomial time. Our current separation is based on a Mixed-Integer Program model that is reasonably effective in practice.
– Triangle cliques are separated in $O(n^3)$ time by the dynamic programming procedure proposed in [3].

An additional element of the proposed BCP is the use of reduced cost fixing to eliminate arcs from formulation (1). The default parameter is pricing paths without 4-cycles. The code is written in C++ and was implemented over the Coin-Bcp framework, version 1.2.2, and used the Coin-LP solver, version 1.10.0 [29]. The experiments were conducted on a machine with an Intel Core 2 Duo 3.06Ghz processor.

Even tough the proposed algorithm is devised for general TDTSPs, all our tests were performed in TDP instances taken from the TSPLIB [30]. In those instances, the cost of an arc (i, j, t) is defined as $(n-t) \cdot d(i, j)$, where $d(i, j)$ is taken from a 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 - the only algorithm able to obtain optimal solutions for instances with $n > 50$ is the combinatorial branch-and-bound proposed in 1993 by Fischetti, Laporte and Martello [16]. Comparisons with the results published in that paper would be meaningless, due to the disparity between machines after almost two decades of computer hardware development. 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 reports the results of those comparisons. Columns Root LB, Nodes and Time represent the lower bound at the root node (bold values are optimal), the number of nodes and the total time to solve the instance for both our code and Fischetti et al.[16]'s (column FLM93). The fast bound computations of Fischetti et al. are advantageous in smaller instances. However, as instances get larger, our stronger lower bounds give our code an advantage (in fact all instances were solved at the root). We include in our table the results of our code for instance brazil58, for which FLM93 was unable to finish solving within a time limit of 21,000 seconds. Solving that harder instance within that time limit required a special parameter setting (that is why this instance is marked with a star), pricing routes without 5-cycles. We also compare our proposed BCP with the best LP based method in the literature, the BCP described in [14].

Instance	OPT	Our BCP			FLM93		
		Root LB	Nodes	Time	Root LB	Nodes	Time
bayg29	22230	**22230**	1	85	20374	5535	0.5
bays29	26862	**26862**	1	3	24268	5194	0.5
berlin52	143721	**143721**	1	85	130292	585347	327.7
eil51	10178	**10178**	1	15	9658	526111	75.4
eil76	17976	**17976**	1	145	16894	$\approx$ 4M	3515.7
brazil58*	512361	**512361**	1	15655	435016	-	-

Table 1. Comparison with the branch-and-bound from [16].

Instance	OPT	BGS08 BCP		Our BCP	
		Time	Nodes	Time	Nodes
gr17	12994	3	1	0.5	1
gr21	24345	10	1	1	1
gr24	13795	15	1	1.5	1
bays29	22230	76	16	4	1
bayg29	26862	191	51	85	1

Table 2. Comparison with the results in [14].

Table 2 reports the results for all TSPLIB instances for which [14] ran their experiments. Their times were obtained in an Intel Pentium 4 3.4 GHz machine. It is worth noting that [14] also eliminate 4-cycles in the pricing, we are basically comparing cut efficacy. We also performed experiments to analyze the effect of using the proposed TDTSP cuts versus the effect of forbidding cycles of higher cardinality in the column generation phase. Notice that, as mentioned before, total elimination of r-cycles by means of cuts seems hard to do, even for r = 3. In contrast, if such a restriction is done in the pricing phase, r-cycles are eliminated in advance. In spite of this, our experiments show that, at least in terms of bounds, our cuts can achieve the same (or better) bounds than r-cycle elimination within a more reasonable time for larger values of r. Table 3

Instance	$r = 4$, no cuts		$r = 4$, all cuts		$r = 5$, no cuts		$r = 6$, no cuts	
	Time	LB	Time	LB	Time	LB	Time	LB
bayg29	6	21824	85	**22230**	34	**22230**	1916	**22230**
bays29	3	**26862**	3	**26862**	35	**26862**	2215	**26862**
berlin52	48	141257	85	**143721**	353	142192	*	*
eil51	15	**10178**	15	**10178**	138	**10178**	5989	**10178**
eil76	93	17949	145	**17976**	452	17956.62	*	*
brazil58	178	468513	*	*	3197	490153	*	*

Table 3. LB obtained using cuts versus eliminating more cycles in the column generation.

illustrates this effect (a star in the table represents that the run did not finish running after 2h). In those instances, it is clear that the best times are achieved by using 4-cycle elimination in the pricing phase combined with cuts.

The TDTSP families of cuts proposed in this article, strong from a polyhedral point of view, appear to be also strong in practice. They were able to completely close the integrality gap for all the TDP instances tested, with n up to 76. However, making an effective use of those cuts to solve larger instances is still a challenge.

References

1. 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)
2. Vajda, S.: Mathematical Programming, Addison-Wesley, (1961)
3. 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)
4. 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)
5. Pessoa, A., Uchoa, E. , Poggi de Aragão, M., Freitas, R.: Algorithms over Arc-time Indexed Formulations for Single and Parallel Machine Scheduling Problems. Technical report RPEP Vol.8 no.8, Universidade Federal Fluminense, Engenharia de Produção Niterói, Brazil (2008)
6. 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)
7. 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)
8. 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)

9. Groetschel, M., Padberg, M.W.: 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)
10. Miranda Bront, J.J., Méndez-Díaz, I., Zabala, P.: An Integer Programming Approach for the Time Dependent Traveling Saleman Problem, In: 23rd European Conference on Operational Research (2009)
11. Fox, K., Gavish, B., Graves, S.: An N-Constraint Formulation of the (Time Dependent) Traveling Salesman Problem. Operations Research 28, 1018–102 (1980)
12. Gouveia, L., Voss, S.: A Classification of formulations for the (time-dependent) traveling salesman problem. European Journal of Operations Research 83, 69–82 (1995)
13. Vander Wiel, R.J., Sahinidis, N.V.: Heuristics Bounds and Test Problem Generation for the time-dependent traveling salesman problem. Transportation Science 29, 167–183 (1995)
14. 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)
15. 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)
16. Fischetti, M., Laporte, G., Martello, S.: The Delivery Man Problem and Cumulative Matroids. Operations Research 41, 1055–1064 (1993)
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. Bentner, J., Bauer, G., Obermair, G.M., Morgensten, I., Schneider, J.: Optimization of the time-dependent traveling salesman problem with Monte Carlo methods. Physical Review E 64, 36701-1 – 36701-8 (2001)
20. Gale, D.: A theorem of flows in networks. Pacific Journal of Mathematics 7, 1073–1082 (1957)
21. 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)
22. Hall, P.,: On representatives of subsets. Journal of London Mathematical Society 10, 26–30 (1935)
23. Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale Traveling Salesman Problem. Operations Research 2, 393–410 (1954)
24. Grotschel, M., Padberg, M.: On the Symmetric Traveling Salesman Problem II: Lifing Theorems and Facets. Mathematical Programming 16, 281–302 (1979)
25. Padberg, M.: On the facial structure of set packing polyhedra. Mathematical Programming 5, 199–215 (1973)
26. 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)
27. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On The Solution Of Traveling Salesman Problems. Documenta Mathematica, Extra Volume ICM 3, 645-646 (1998)
28. Niskanen S., Ostergard, P.R.J.: Cliquer users guide. Helsinki University of Technology, Communications Laboratory, Technical report 48 (2003)
29. Ralphs, T.K., Ladányi, L.: COIN/BCP User's Manual. Available at www.coin-or.org/Presentations/bcp-man.pdf (2001)
30. Reinelt, G.: TSPLIB - A traveling salesman problem library. ORSA Journal on Computing 3:4, 376–384 (1991)

Appendix

Proof of Lemma 1

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:

$$\sum_{j\in N} x^0_{0,j} + \sum_{t=1}^{n-1}\sum_{i\in N}\sum_{j\in N\setminus i} x^t_{i,j} = n \tag{11}$$

Thus, we can eliminate from (1e) the equation corresponding to an arbitrary single index $k \in N$ and (11) 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. $\square$

Proof of Theorem 1

Proof: Lemma 2 implies that $\dim P(n) \le n(n-1)(n-2)$. 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 $n \ge 5$, we need to show that $P_{n+1} = (n+1)n(n-1)$. Consider $G(n+1)$ and its subgraph $G(n)$. The induction hypothesis asserts that $G(n)$ contains $\dim P(n)+1 = n(n-1)(n-2)+1$ affine independent (AI) 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 $\dim P(n) + 1 = n(n-1)(n-2) + 1$ AI s-paths in $G(n+1)$. To complete the required set of AI 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 AI 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 AI 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 $(n-1)(n-2)$ AI 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 $n^2 - 1 = (n-1)(n-2) + (n-1) + 2(n-2) + 2$ AI 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 $2n-1$ AI 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 $2n-1$ AI s-paths containing node $(n+1, t)$. Repeating this procedure for each $t = 2, \ldots n$ yields $(n-1)(2n-1)$ AI s-paths.

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

Proof of Theorem 2

Proof: Without loss of generality, we may assume $i = 1$ 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 (3) 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 (3) for arc $(1, n, t)$ as strict inequality.

Next, since $\dim P_n = n(n - 1)(n - 2)$, we need to show that $n(n - 1)(n - 2)$ affinely independent vectors in P_n satisfy constraint (3) associated with arc $(1, n, t)$ as equality. The construction in the proof of Theorem 1 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 (3) 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 1, $|\overline{B}| = 2n - 3$. We replace $\overline{B}$ by a set B' composed of $2n - 4$ linearly independent s-paths, all of which satisfy constraint (3) for arc $(1, n, t)$ as equality.

The set B' is constructed as follows. For each $k \in N(1, n)$ we combine incoming arc (k, n, t) with outgoing arc $(n, 1, t+1)$ to generate $n-2$ linearly independent s-paths which satisfy (3) 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 (3) as equation with value 1.

Note $B \cup B'$ is linearly independent by construction and $|B \cup B'| = \dim P_n$. □

Proof of Theorem 3

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 $\{x^{t-1}_{i,j} : i \in N_j\}$ and outgoing arcs $\{x^t_{j,k} : k \in N_j\}$. 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 $x^{t-1}_{i,j}$ and each node $k \in D$ has its demand set equal to $x^t_{j,k}$. For each pair $i, k \in N_j$ 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 3. 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 $x^t_{i,j}$, 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. □

Proof of Theorem 4

Proof: We prove by induction on n and on $|S|$ that there are $n(n-1)(n-2)$ linearly independent (LI) s-paths that satisfy (8) as equality. We first establish that (8) 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 (8) as equality. We show below how to construct $(n+1)n(n-1)$ LI paths in $G(n+1)$ that satisfy (8) 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 1. 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 (8) 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 (8). 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 (8) 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 (8). Hence, all these LI paths in $G(n+1)$ satisfy (8) as equality. Like in the proof of Theorem 1, 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 1, 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 (8) 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 (8) 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 1.

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 1. 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 (8) 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 $1,\dots,|S|-1$, 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 (8) 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 $S=\{n-|S|+1,\dots,n\}$. Now, by Lemma 4 it is equivalent to consider instead constraint (7).

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 (7) as equality. We show how to construct $(n+1)n(n-1)$ LI paths in $G(n+1)$ that satisfy (7) 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 1. This is true because the coefficients of all arcs of $G(n)$ in (7) 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 (7).

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 (7) as equality. Hence, the pairs of arcs can be chosen as in proof of Theorem 1. 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 (7) 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 1, 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 of Lemma 5

Proof: 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 $\mathcal{I}$. Thus, at least one of these layers has an index r satisfying $|r - s| > 1$ and $|r - t| > 1$. Choose $c = (p, q, r) \in \mathcal{I}$ such that $p, q \in S \setminus \{i, j\}$. 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 $a \in \mathcal{I}$, we are done. Alternatively, $a \notin \mathcal{I}$, which implies that $\mathcal{I}$ does not contain arcs from layer 1. Then, since $\mathcal{I}$ is an alternating independent set, it must contain the set of arcs $A(S, 2)$ from layer 2. Let p be the unique element in $S \setminus \{i, j\}$ and let $c = (j, p, r) \in \mathcal{I}$. Again, one can find an s-path containing arcs a, b and c. $\square$

Proof of Theorem 5

Proof: We need to find $n(n-1)(n-2)$ linearly independent s-paths that satisfy (9) 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 1. 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 (9) 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 (9) as equality. We show 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 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 (9) 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 1. By Lemma 5, these s-paths can be constructed to satisfy (9) as equality. $\square$