Discrete Applied Mathematics 156 (2008) 3223–3237

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

A new formulation for the Traveling Deliveryman Problem☆

Isabel Méndez-Díaz [a,*], Paula Zabala [a], Abilio Lucena [b]

[a] Depto. de Computación, FCEyN, Universidad de Buenos Aires, Argentina

[b] Depto. de Administração, Universidade Federal do Rio de Janeiro, Brazil

ARTICLE INFO

Article history:
Received 5 October 2005
Received in revised form 1 December 2006
Accepted 21 May 2008
Available online 27 June 2008

Keywords:
Traveling deliveryman problem
Integer programming
Branch-and-cut algorithms

ABSTRACT

The Traveling Deliveryman Problem is a generalization of the Minimum Cost Hamiltonian Path Problem where the starting vertex of the path, i.e. a depot vertex, is fixed in advance and the cost associated with a Hamiltonian path equals the sum of the costs for the layers of paths (along the Hamiltonian path) going from the depot vertex to each of the remaining vertices. In this paper, we propose a new Integer Programming formulation for the problem and computationally evaluate the strength of its Linear Programming relaxation. Computational results are also presented for a cutting plane algorithm that uses a number of valid inequalities associated with the proposed formulation. Some of these inequalities are shown to be facet defining for the convex hull of feasible solutions to that formulation. These inequalities proved very effective when used to reinforce Linear Programming relaxation bounds, at the nodes of a Branch and Bound enumeration tree.

1. Introduction

Given a vehicle depot and a number of geographically dispersed customers, assume that direct travel times are known for every pair of locations involved. The Traveling Deliveryman Problem (TDP) is to find a path, initiated at the depot and visiting every customer exactly once, such that the sum of the times required to reach every customer, along the path, is a minimum. As such, an optimal TDP solution also minimizes, under the constraints imposed, the average time to reach a customer. Therefore, applications of TDP frequently arise in delivery situations where some kind of fairness criteria (for the visiting of clients) must be enforced.

An example of a practical application of TDP is the home delivery of pizzas [10]. Typically, various delivery orders are put together and one wishes to minimize the average arrival time at a customer. Another application is found in the area of computer networks where one wishes to find information stored somewhere in the network [4,15]. Assume that network *locations* have equal probability of containing the desired information. One thus wishes to design a visiting path for the locations such that the average time to reach each of them is minimized. Additional applications of the problem could be found, among others, in disk-head scheduling [5] and in the routing of automated guided vehicles through cells in a flexible manufacturing system [20]. In the literature, TDP is also known as The Traveling Repairman Problem (TRP) [1,4,11] and the Minimum Latency Problem (MLP) [2,3,5,12,21].

Various Combinatorial Optimization problems seek permutations of given sets of items to minimize associated objective functions. Many of these could be cast in terms of a complete digraph where vertices represent items and arcs indicate the possibilities of having two items (i.e. the endpoints of a given arc) placed in immediately subsequent *permutation positions*. In what follows, let us assume that the item occupying the first permutation position is always known in advance. This is

☆ Research partially supported by grants CAPG-BA 008/02 and CNPq PROSUL 4900333/04-4.

* Corresponding author. Tel.: +54 11 45763359; fax: +54 11 45763359.
E-mail addresses: imendez@dc.uba.ar (I. Méndez-Díaz), pzabala@dc.uba.ar (P. Zabala), abiliolucena@globo.com (A. Lucena).

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.05.009

done without any loss of generality since the introduction of an *artificial* item, together with some conveniently defined associated costs, suffices to redefine a general TDP, where the starting vertex is not fixed in advance.

Depending on the objective function involved, some well known problems could easily be identified within the framework suggested above. The most well known of these arise when the objective function only involves terms associated with the cost of placing pairs of items in immediately subsequent positions of the permutation, say a cost c_{ij} is incurred whenever item j immediately follows item i in the permutation. The Hamiltonian Path Problem (HPP) [6] thus results. A generalization of HPP arises when the cost of placing items i and j, say in positions k and $k+1$ of a permutation, is given by $(n-k)c_{ij}$, for c_{ij} as defined above. Such a problem is precisely TDP. It should be noticed that the HPP objective function is biased towards the vehicle (say, one wishes to minimize the total time required to visit all customers) while the TDP objective function is biased towards customers (one wishes to minimize the sum of arrival times at each customer).

TDP could be formally described as follows. Let $D = (V, A)$ be a complete digraph with a set $V = \{v_0, v_1, \ldots, v_n\}$ of vertices and a set A of arcs. Assume that *costs* $\{c_{ij} \geq 0 : (v_i, v_j) \in A\}$ are associated with the arcs of D and that v_0 is a special vertex representing a *depot*. Feasible TDP solutions are given by Hamiltonian paths of D starting at the depot and are denoted *TDP paths*. Given a TDP path, an arc pointing outwards of v_0 must occupy the first of the n available *path positions*. The arcs that follow, along that path, sequentially occupy positions 2 to n. Each arc (v_i, v_j) in a TDP path contributes $(n-k+1)\ c_{ij}$ to the *cost* of the path, where k is the corresponding path position of the arc. Alternatively, one may interpret the cost of a TDP path as the sum of the costs for the layers of paths (along the underlying Hamiltonian path) going from the depot vertex to each of the remaining vertices in the problem. TDP is defined as the problem of finding a least cost TDP path.

TDP is $\mathcal{NP}$-hard for general metrics [19] and remains so even when the metric space is induced by a tree [21] (note that HPP is trivial, in this case). If the metric is defined over points (vertices) placed on a line segment, a $O(n^2)$ Dynamic Programming exact solution algorithm exists for the problem [1]. For that particular case, a linear time algorithm was later suggested in [11].

In spite of the close links between HPP and TDP, the two problems are, structurally speaking, quite different from each other. For instance, a small variation on the cost of a given arc tends to have a marked global impact on TDP optimal solutions. Contrary to that, the corresponding impact on HPP is typically restricted to a small neighborhood of the solution. However, the two problems could be seen as special cases of the *Time-Dependent Traveling Salesman Problem* (TDTSP) [18]. For TDTSP, a general cost c_{ijk} is incurred whenever items i and j respectively appear in positions k and $k+1$ of a Hamiltonian tour. As pointed out before, for that situation, $c_{ijk} = c_{ij}$ for HPP while $c_{ijk} = (n-k+1)c_{ij}$ for TDP.

Exact solution algorithms for TDP could be found, among others, in Fischetti et al. [10], Lucena [17], and van Eijl [8]. In the literature, the largest TDP instance solved to proven optimality does not exceed 60 customers [10]. Approximation algorithms for the problem are found in [2,3,5,12,13]. Various heuristic [22,24] and online [9,16] algorithms for the problem have also been proposed.

In this paper, a new linear Integer Programming (IP) formulation is introduced for TDP and its Linear Programming (LP) relaxation is computationally compared with its counterparts in Fischetti, Laporte and Martello [10] and van Eijl [8]. Additionally, a number of inequalities that are valid for our formulation are also described. Some of these are shown to be facet defining for the polytope given by the convex hull of feasible solutions to our formulation. Computational results are presented for a cutting plane algorithm based on our reinforced formulation. That algorithm is then used to reinforce bounds on the nodes of a Branch and Bound enumeration tree.

This paper differs from TDP papers in the literature in that it concentrates on polyhedral solution approaches to the problem. To the best of our knowledge, this is the very first time such an approach is applied to TDP.

This paper is organized as follows. In Section 2, a new TDP formulation is proposed. In Section 3, that formulation is computationally compared with two TDP formulations found in the literature. In Section 4, some polyhedral results are described for the formulation introduced here. In Section 5, a cutting plane algorithm, based on these inequalities, is computationally tested and is embedded into the nodes of a Branch and Bound enumeration tree. Finally, the paper is closed in Section 6, with a few remarks and suggestions for future work.

2. A new formulation to TDP

The formulation that follows, exploits connections between TDP and the Linear Ordering Problem (LOP) [14].

A *linear ordering* (or a permutation) of the $n+1$ elements of a finite set N is a bijective mapping $\pi : \{0, 1, \ldots, n\} \to N$. For an element $i \in N$ and an ordering π, the *position* of i in π is given by $\pi^{-1}(i)$. The ordering cost $d(\pi)$ is computed from a set of *precedence* costs, $\{c_{ij} : i, j \in N, i \neq j\}$. Every pair of distinct elements $i, j \in N$ must contribute with either c_{ij} or else c_{ji} to $d(\pi)$. If $\pi^{-1}(i) < \pi^{-1}(j)$ then c_{ij} is incurred. Otherwise, c_{ji} is incurred. The Linear Ordering Problem (LOP) is to find an ordering π of N with as large a $d(\pi)$ as possible.

LOP can be cast in graph theoretical terms by assuming the elements of N to be the vertices of a complete digraph $D = (V, A)$. Accordingly, $A = \{(v_i, v_j) : i, j \in N, i \neq j\}$ is defined as the arc set and the precedence costs introduced above are thus taken to be arc costs. A subgraph $D' = (V, A')$, $A' \subseteq A$, denotes a tournament if, for every pair of distinct nodes $v_i, v_j \in V$, exactly one of the arcs, (v_i, v_j) or (v_j, v_i), is contained in A'. An acyclic tournament D' with the largest sum of arc costs implies an optimal LOP solution to N.

Let x_{ij} be a binary 0–1 variable that takes a value of 1 if vertex v_i appears prior to vertex v_j in the TDP path, taking a value of 0, otherwise. Additionally, for every vertex $v_k \in V$, a continuous variable f_{ij}^k assumes a value of 1 if arc (v_i, v_j) is used to

go from v_0 to v_k in the solution path. One should notice that variables f_{0j}^k may be conveniently replaced with variables f_{0j}^j. That applies since the very first arc in the TDP path must be used in any subpath going from v_0 to v_k, where $v_k \in V\backslash\{v_0\}$. A TDP formulation, *MR*, is thus given by

$$\min \quad \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} c_{ij} \sum_{k=1,k\neq i}^{n} f_{ij}^k + n\sum_{i=1}^{n} c_{0i} f_{0i}^i$$

$$\text{s.t.} \quad x_{ij} + x_{ji} = 1 \quad i,j = 1,\ldots,n, i<j \tag{1}$$

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \quad i,j,k = 1,\ldots,n, i<j<k \tag{2}$$

$$x_{ik} + x_{kj} + x_{ji} \leq 2 \quad i,j,k = 1,\ldots,n, i<j<k$$

$$x_{jk} = \sum_{i=1,i\neq j,k}^{n} f_{ij}^k + f_{0j}^j \quad j,k = 1,\ldots,n, j\neq k \tag{3}$$

$$x_{jk} = \sum_{i=1,i\neq j}^{n} f_{ji}^k j, k = 1,\ldots,n, j\neq k \tag{4}$$

$$\sum_{i=1}^{n} f_{0i}^i = 1 \tag{5}$$

$$\sum_{k=1\,k\neq j}^{n} x_{jk} - \sum_{k=1\,k\neq j,i}^{n} x_{ik} \leq f_{ij}^j - n x_{ij} + n - 1 \quad i,j = 1,\ldots,n, i\neq j \tag{6}$$

$$f_{ki}^i + x_{ij} - f_{ki}^j \leq 1 \quad i,j,k = 1,\ldots,n, i\neq j\neq k \tag{7}$$

$$x_{ij} \in \{0,1\} \quad i,j = 1,\ldots,n, i\neq j$$

$$0 \leq f_{ij}^k \leq 1 \quad i,j,k = 1,\ldots,n, i\neq j,k$$

$$0 \leq f_{0i}^i \leq 1 \quad i = 1,\ldots,n.$$

The above formulation contains $n(n^2-n+1)$ variables and $\frac{n(n-1)(8n+5)}{6}+1$ constraints. Constraints (1) and (2) define the set of linear orderings. Constraint (5) ensures that only one arc pointing outwards of v_0 exists. If vertex v_i is visited in the TDP path immediately before vertex v_j, then constraint (6) ensures that arc (v_i, v_j) must be used. Inequalities (7) impose that the arc pointing inwards of v_i must be used to reach v_j if v_i is visited prior to v_j. Finally, if v_j is not visited prior to v_k, constraints (3) and (4) impose that none of the arcs (v_i, v_j) and (v_j, v_i) is used to reach v_k.

It should be pointed out that integrality of variables x_{ij} allows the integrality of variables $f_{ij}^j, f_{ij}^k, f_{0i}^i$ to be relaxed. Constraints (6) ensure that $f_{ij}^j = 1$ if $x_{ij} = 1$ and vertex v_i is the predecessor of vertex v_j (since $\sum_{k=1,k\neq j}^{n} x_{jk} - \sum_{k=1,k\neq i,j}^{n} x_{ik} = 0$). Otherwise, $f_{ij}^j = 0$, from constraint (4). Similar arguments apply for f_{ij}^k and f_{0i}^i in relation, respectively, with constraints (3) and (7). Finally, given that one is faced with a minimization problem, if constraints (3) and (4) are omitted from the formulation, they will be implicitly satisfied at an optimal solution. However, in our computational experiments, these constraints proved useful in cutting CPU times down and are therefore kept in the formulation.

3. Linear Programming relaxation bounds

The most recent TDP formulations to appear in the literature are found in Fischetti, Laporte and Martello [10] and in van Eijl [8]. These formulations address the oriented (or Hamiltonian circuit) version of the problem and are computationally compared here with our own formulation, which is also oriented.

Randomly generated test instances, grouped in sets named *A*, *D* and *S*, respectively, are used in the experiments. Instances in set *D* are generated by drawing points in a 100 × 100 square from the uniform distribution. These points are then interpreted as vertices of a graph where arc costs are given by Euclidian distances (rounded to the nearest integer) between corresponding end points. Instances in set S are generated by drawing arc costs $c_{ij} = c_{ji}$ from uniform distribution in the range [1, 100], for every pair of distinct vertices $i, j \in V$. Instances in sets *D* and *S* therefore involve symmetric arc costs. Contrary to that, instances in set *A* involve asymmetric arc costs drawn from uniform distribution in the range [1, 100]. For either set, *A*, *D* and *S*, five instances are generated for every value of *n* considered, with $20 \leq n \leq 40$.

For our TDP formulation, the use of constraints (3) and (4) allow variables *x* to be dropped. However, the resulting formulation proved far too dense, thus implying a loss of efficiency. Another variable elimination possibility results from the use of constraints (1) and implies a decrease from $n(n-1)$ to $n(n-1)/2$ in the number of variables *x*. In this case, mixed results were obtained. The strategy proved better than the previous one. However, in spite of the polynomial number of constraints involved, CPU time demands (for solving the resulting LP relaxations) proved excessive even for medium size instances. For that reason, we ended up settling for an initial TDP relaxation that involves only the equality constraints in (1)–(7). In doing

Table 1
Comparison of LP relaxation's gaps

n	*MR*		*FLM*		*Eijl*	
	Time	%Gap	Time	%Gap	Time	%Gap
D-20	0.09	17.58	0.03	26.20	0.03	90.36
D-22	0.11	22.75	0.03	35.57	0.04	91.43
D-24	0.17	22.97	0.05	33.64	0.05	92.47
D-26	0.22	22.12	0.05	34.90	0.06	93.02
D-28	0.40	23.29	0.06	35.79	0.07	93.62
D-30	0.48	21.78	0.07	32.53	0.09	93.78
D-35	0.91	24.58	0.12	38.87	0.14	94.63
D-40	2.01	26.20	0.21	41.02	0.18	95.38
Average	0.55	22.66	0.08	34.81	0.08	93.09
S-20	0.08	35.84	0.02	59.14	0.03	88.56
S-22	0.13	28.03	0.03	41.86	0.03	93.35
S-24	0.19	43.67	0.05	68.40	0.05	91.08
S-26	0.27	36.52	0.06	56.97	0.06	80.86
S-28	0.42	38.77	0.07	59.34	0.09	92.16
S-30	0.58	38.76	0.09	59.59	0.12	93.42
S-35	1.04	46.40	0.13	69.78	0.16	93.84
S-40	2.71	44.84	0.19	66.71	0.23	94.97
Average	0.68	39.10	0.08	60.22	0.10	91.03
A-20	0.14	10.33	0.03	24.47	0.03	87.98
A-22	0.20	11.20	0.03	24.56	0.04	89.15
A-24	0.25	13.48	0.05	28.99	0.05	89.91
A-26	0.44	11.78	0.05	25.53	0.07	90.85
A-28	0.74	11.89	0.07	25.41	0.08	91.55
A-30	0.79	14.43	0.10	30.25	0.10	92.01
A-35	1.56	16.36	0.12	25.77	0.15	92.86
A-40	5.78	17.37	0.15	37.68	0.22	93.68
Average	1.24	13.35	0.07	27.83	0.09	91.00

so, overall number of constraints dropped to $2n(n-1)+1$. However, LP relaxation bound quality suffered considerably. In any case, after balancing gains and losses, overall, relaxation proved attractive and was selected to be used in our experiments. Inequalities initially dropped from the formulation, were later considered for inclusion in a cutting plane algorithm.

Computational experiments were carried out on a SUN UltraSparc III workstation with 2GB of RAM and running at 1Ghz. Corresponding LP relaxation solutions were obtained with the solver CPLEX 8.1 [7].

Table 1 compares LP relaxation bounds for the formulations in [8,10] with LP relaxation bounds for the compact version of our formulation, as suggested above. For each formulation tested and for every value n and set of instances considered, table entries give the average percentage gaps between optimal values and corresponding LP relaxation values. These gaps, measured as $(\frac{(\text{Opt}-\text{LP Relaxation value})*100}{\text{Opt}})$ are followed by corresponding average CPU times. As quoted before, we report on the average results obtained over 5 instances for each size n considered.

Results in Table indicate that our TDP formulation appears to be quite promising. In spite of its higher CPU time demands, the LP relaxation bounds it generates are much stronger than those returned by their counterparts in [10,8]. One should therefore expect that, within an LP relaxation based enumeration scheme to solve the problem, our stronger LP relaxation bounds should eventually pay off.

From our computational experience, a Branch and Bound algorithm directly based on van Eijl's formulation appears very uncompetitive with similar algorithms based on the other two formulations. Specifically, more than one hour of CPU time was taken by a Branch and Bound algorithm based on van Eijl's formulation, to solve TDP instances with as few as 16 clients. These instances, however, were solved in a few seconds of CPU times by the other two algorithms. As a result, algorithms based on van Eijl's formulation are not quoted on the tables that follow.

On Table 2, we compare the performance of Branch and Bound algorithms for larger TDP instances. For each algorithm tested and for every value of n and set of instances considered, table entries respectively give corresponding average CPU times and average tree sizes. As before, results quoted here are for the averages over 5 instances tested. For the cases where not all instances were solved within the time limit imposed, i.e. 2 h of CPU time, results quoted are averages for those instances eventually solved. For each such case, the number of instances solved appears in brackets. Asterisks are used for those cases where no instances were solved within the time limit imposed.

As it can be appreciated from Table 2, the Branch and Bound algorithm based on the Fischetti, Laport and Martello (*FLM*) formulation was clearly dominated by the Branch and Bound algorithm based on our TDP formulation. This conclusion applies despite the fact that LP relaxation bounds are computationally cheaper to compute for the *FLM* formulation. However, as indicated by the results on Table 2, the stronger LP relaxation bounds returned by our formulation, albeit at a higher CPU cost, proved to be a more effective alternative for Branch and Bound enumeration.

Table 2
Branch-and-Bound algorithm on *MR* and *FLM* models

n	*MR*		*FLM*	
	Time	Nodes	Time	Nodes
D-20	87.30	23.00	*****	*****
S-20	104.50	29.20	*****	*****
A-20	8.22	2.00	92.55(4)	32 875
D-26	4608.42	429.25	*****	*****
S-26	2686.28	286.40	*****	*****
A-26	34.60	4.40	*****	*****

The results above encouraged us to move one step forward and investigate the use of valid inequalities to reinforce the LP relaxation bounds returned by our formulation.

4. Polyhedral investigations

The aim of this section is to analyze polyhedral properties of the TDP formulation introduced in Section 2. Accordingly, let $\mathcal{CR}$ be the polytope associated with the convex hull of the feasible integral solutions for that formulation.

As the first step in our polyhedral investigations, we would like to find a minimal equation system for $\mathcal{CR}$ and determine its dimension.

Proposition 1. *The dimension of* $\mathcal{CR}$ *is* $n(n-1)(n-\frac{5}{2})+n-1$ *and a minimal equation system for that polytope is defined as*

$$x_{ij} + x_{ji} = 1 \quad \forall i, j = 1, \ldots, n, i < j \tag{8}$$

$$x_{jk} = \sum_{i=1, i\neq j,k}^{n} f_{ij}^{k} + f_{0j}^{j} \quad \forall j, k = 1, \ldots, n, j \neq k \tag{9}$$

$$x_{jk} = \sum_{i=1, i\neq j}^{n} f_{ji}^{k} \quad \forall j, k = 1, \ldots, n, j \neq k \tag{10}$$

$$\sum_{i=1}^{n} f_{0i}^{i} = 1. \tag{11}$$

Proof. Consider, as a row ordering for the constraint matrix,

- Eq. (8) in lexicographic order on ij ($i < j$).
- Eq. (9) for $j = 1, k = 2$.
- Eq. (9) for $j = 2, \ldots, n-1, k = 1$.
- Eq. (11).
- Eq. (9) for $j = 1$ and $k = 3, \ldots, n$.
- Eq. (9) for $j = 2, \ldots, n$ and $k = 2, \ldots, n$ ($j \neq k$).
- Eq. (9) for $j = n$ and $k = 1$.
- Eq. (10) in lexicographic order on jk ($j \neq k$).

Consider as well the following constraint matrix column ordering:

- Variables x_{ij}, $i < j$ in lexicographic order on ij.
- Variables f_{0j}^{j}, $j = 1, \ldots, n$.
- Variables f_{21}^{k}, $k = 3, \ldots, n$.
- Variables f_{1j}^{k}, $j = 2, \ldots, n, k = 2, \ldots, n, j \neq k$.
- Variable f_{2n}^{1}.
- Variables f_{ij}^{j}, $i = 1, \ldots, n, j = 1, \ldots, n, i \neq j$.
- Any order for the rest of the variables.

As a result of the ordering suggested above, it is evident that the constraint matrix contains an $\frac{n(n-1)}{2} + 2n(n-1) + 1$ lower triangular block having non-zero coefficients along the diagonal. It is then full rank.

Suppose now that there exists an equality

$$\alpha X + \beta F_{ini} + \gamma F_{sig} + \delta F_{cam} = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \alpha_{ij} x_{ij} + \sum_{i=1}^{n} \beta_i f_{0i}^{i} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \gamma_{ij} f_{ij}^{j} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \delta_{ijk} f_{ij}^{k} = \pi_0 \tag{12}$$

valid with respect to $\mathcal{CR}$. To prove this proposition, we have to show that $(\alpha, \beta, \gamma, \delta)$ can be written as a combination of the minimal equation system. We now proceed to proving different cases which allow us to gather the necessary information on the coefficients in α, β, γ and δ.

Consider scalars a_{ij} $i, j = 1, \ldots, n$ $i < j$ (corresponding to (8)), b_{ij} $i, j = 1, \ldots, n$ $i \neq j$ (corresponding to (9)), c_{ij} $i, j = 1, \ldots, n$ $i \neq j$ (corresponding to (10)) and d (corresponding (11)) such that:

- $\alpha_{ij} = a_{ij} + b_{ij} + c_{ij}$ $\forall i < j$
- $\alpha_{ij} = a_{ji} + b_{ij} + c_{ij}$ $\forall i > j$
- $\beta_i = d - \sum_{j=1\ i\neq j}^{n} b_{ij}$ $\forall i$
- $\gamma_{ij} = -c_{ij}$ $\forall i \neq j$
- $\delta_{ijk} = -c_{ik} - b_{jk}$ $\forall i \neq j \neq k$.

We then define

- $c_{ij} = -\gamma_{ij}$ $\forall i \neq j$
- $b_{jk} = \gamma_{ik} - \delta_{ijk}$ $\forall j \neq k$ for a fixed $i \neq j, k$
- $d = \beta_i + \sum_{r=1\ r\neq i}^{n} b_{ir}$ for a fixed i
- $a_{ij} = \alpha_{ij} - \gamma_{ki} + \delta_{kij} + \gamma_{ij}$ $\forall i < j$ for a fixed $k \neq i, j$.

To establish validity of our choice above, we have to prove that

- *Case* 1: The definition of b_{jk} does not depend on i, i.e.

$$\gamma_{ik} - \delta_{ijk} - \gamma_{i'k} + \delta_{i'jk} = 0 \quad \forall i \neq i'.$$

- *Case* 2: The definition of d does not depend on i and

$$\beta_i + \sum_{r=1\ r\neq i}^{n} b_{ir} = \beta_{i'} + \sum_{r=1\ r\neq i'}^{n} b_{i'r} \quad \forall i \neq i'.$$

Then,

$$\beta_i + \gamma_{ki'} - \delta_{jik} - \sum_{r=1, r\neq i,k}^{n} \delta_{kir} - \beta_{i'} - \gamma_{ki} + \delta_{ji'k} + \sum_{r=1, r\neq i',k}^{n} \delta_{ki'r} = 0 \quad \forall i \neq i'.$$

- *Case* 3: Since $\alpha_{ij} = a_{ji} + b_{ij} + c_{ij}$ for $i < j$, it must be checked that

$$\alpha_{ji} + \delta_{kji} + \gamma_{ji} + \gamma_{kj} - \alpha_{ij} - \delta_{kij} - \gamma_{ij} - \gamma_{ki} = 0 \quad \forall i < j, \text{ for any } k.$$

We will now generate multipliers λ and feasible solutions of $\mathcal{CR}$ which differ in their components and allow us to conveniently concentrate on each of the cases above.

- *Proof Case* 1: Consider feasible solutions $FS^1, \ldots, FS^{12}$ of $\mathcal{CR}$, associated with the following paths:

1: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_i\ v_j\ v_{i'}\ v_k$
2: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_i\ v_{i'}\ v_j\ v_k$
3: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_k\ v_i\ v_j\ v_{i'}$
4: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_k\ v_i\ v_{i'}\ v_j$
5: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_k\ v_{i'}\ v_i\ v_j$
6: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i'}\ v_i\ v_j\ v_k$
7: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_j\ v_i\ v_{i'}\ v_k$
8: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_j\ v_{i'}\ v_i\ v_k$
9: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i'}\ v_j\ v_i\ v_k$
10: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_k\ v_j\ v_i\ v_{i'}$
11: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_k\ v_j\ v_{i'}\ v_i$
12: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_k\ v_{i'}\ v_j\ v_i$

where $\{v_{j_1}, \ldots, v_{j_{n-4}}\} = V \backslash \{v_0, v_i, v_{i'}, v_j, v_k\}$.

Additionally, consider the linear combination

$$\sum_{i=1}^{12} \lambda_i (\alpha X^i + \beta F^i_{ini} + \gamma F^i_{sig} + \delta F^i_{cam}) = \sum_{i=1}^{12} \lambda_i \pi_0$$

where $\lambda = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$.

It is easy to check that $\gamma_{ik} - \delta_{ijk} - \gamma_{i'k} + \delta_{i'jk} = 0$ follows from the linear combination above.

- *Proof Case* 2: Consider feasible solutions $FS^1, \ldots, FS^8$ of $\mathcal{CR}$, associated with the following paths:

1: $v_0\ v_i\ v_{i'}\ v_k\ v_{j_1} \ldots v_{j_{n-4}}\ v_j$
2: $v_0\ v_k\ v_i\ v_{i'}\ v_{j_1} \ldots v_{j_{n-4}}\ v_j$
3: $v_0\ v_k\ v_j\ v_i\ v_{i'}\ v_{j_1} \ldots v_{j_{n-4}}$
4: $v_0\ v_j\ v_i\ v_{i'}\ v_k\ v_{j_1} \ldots v_{j_{n-4}}$
5: $v_0\ v_{i'}\ v_k\ v_{j_1} \ldots v_{j_{n-4}}\ v_i\ v_j$
6: $v_0\ v_k\ v_{i'}\ v_{j_1} \ldots v_{j_{n-4}}\ v_i\ v_j$
7: $v_0\ v_k\ v_j\ v_{i'}\ v_{j_1} \ldots v_{j_{n-4}}\ v_i$
8: $v_0\ v_j\ v_{i'}\ v_k\ v_{j_1} \ldots v_{j_{n-4}}\ v_i$

where $\{v_{j_1}, \ldots, v_{j_{n-4}}\} = V \backslash \{v_0, v_i, v_{i'}, v_j, v_k\}$. Taking $\lambda = (1, -1, 1, -1, -1, 1, -1, 1)$ we conclude that

$$\beta_i + \gamma_{ki'} - \delta_{jik} - \sum_{r=1, r\neq i,k}^{n} \delta_{kir} - \beta_{i'} - \gamma_{ki} + \delta_{ji'k} + \sum_{r=1, r\neq i',k}^{n} \delta_{ki'r} = 0 \quad \forall i \neq i'.$$

- *Proof Case* 3: Consider feasible solutions FS^1 and FS^2 of $\mathcal{CR}$ associated with the following paths:
  1: $v_0\ v_{j_1} \ldots v_{j_{n-3}}\ v_k\ v_i\ v_j$
  2: $v_0\ v_{j_1} \ldots v_{j_{n-3}}\ v_k\ v_j\ v_i$
  where $\{v_{j_1}, \ldots, v_{j_{n-3}}\} = V \backslash \{v_0, v_i, v_j, v_k\}$. Defining $\lambda = (1, -1)$ it then follows that:

$$\alpha_{ji} + \delta_{kji} + \gamma_{ji} + \gamma_{kj} - \alpha_{ij} - \delta_{kij} - \gamma_{ij} - \gamma_{ki} = 0 \quad \forall i < j, \text{ for any } k. \quad \square$$

We present next some valid inequalities that are facet defining for $\mathcal{CR}$.

4.1. Facet-defining inequalities

Proposition 2. ***Family* 1** *inequalities*

$$f_{ij}^k \leq f_{ij}^j \quad i, j, k = 1, \ldots, n, i \neq j, k, j \neq k.$$

are facet-defining for $\mathcal{CR}$.

Proof. Let F be the face of $\mathcal{CR}$ defined by ***Family* 1** inequality $f_{i_0j_0}^{k_0} \leq f_{i_0j_0}^{j_0}$. Any path where v_{i_0} and v_{j_0} are the first vertices in the path, lie on F. Then F is a proper face of $\mathcal{CR}$.

Suppose that there is an inequality $\alpha X + \beta F_{ini} + \gamma F_{sig} + \delta F_{cam} = \pi_0$ valid with respect to $\mathcal{CR}$ such that $F \subseteq \mathcal{CR} \cap \{(X, F_{ini}, F_{sig}, F_{cam}) : \alpha X + \beta F_{ini} + \gamma F_{sig} + \delta F_{cam} = \pi_0\}$. To prove the proposition, we have to show that $(\alpha, \beta, \gamma, \delta)$ can be written as a combination of the minimal equation system and the ***Family* 1** inequality. We proceed to prove some different cases involved which allow us to gather sufficient information on the coefficients of α, β, γ and δ to prove the proposition.

Consider scalars a_{ij} $i, j = 1, \ldots, n$ $i < j$ (corresponding to (8)), b_{ij} $i, j = 1, \ldots, n$ $i \neq j$ (Eq. (9)), c_{ij} $i, j = 1, \ldots, n$ $i \neq j$ (Eq. (10)), d (Eq. (11)) and e (corresponding to *Family* 1) such that

- $\alpha_{ij} = a_{ij} + b_{ij} + c_{ij}\ \forall i < j$
- $\alpha_{ij} = a_{ji} + b_{ij} + c_{ij}\ \forall i > j$
- $\beta_i = d - \sum_{r=1\ r\neq i}^{n} b_{ir}\ \forall i$
- $\gamma_{ij} = -c_{ij}\ \forall i \neq j, (i, j) \neq (i_0, j_0)$
- $\delta_{ijk} = -c_{ik} - b_{jk}\ \forall i \neq j \neq k, (i, j, k) \neq (i_0, j_0, k_0)$
- $\gamma_{i_0j_0} = -c_{i_0j_0} - e$
- $\delta_{i_0j_0k_0} = -c_{i_0k_0} - b_{j_0k_0} + e$.

We then define:

- $e = \delta_{i_0j_0k_0} - \gamma_{i_0k_0} + \gamma_{ik_0} - \delta_{ij_0k_0}$ for some $i \neq i_0, j_0, k_0$
- $c_{ij} = -\gamma_{ij}\ \forall (i, j) \neq (i_0, j_0)$
- $c_{i_0j_0} = -\gamma_{i_0j_0} + e$
- $b_{jk} = \gamma_{ik} - \delta_{ijk}$ for some i, $(i, k) \neq (i_0, j_0)$, $(i, j, k) \neq (i_0, j_0, k_0)$
- $d = \beta_i + \sum_{j=1\ i\neq j}^{n} b_{ij}$ for some i
- $a_{ij} = \alpha_{ij} - b_{ij} - c_{ij}\ \forall i < j$.

To establish validity of the above, we must prove that

- *Case* 1: b_{jk} does not depend on i, i.e.

$$\gamma_{ik} - \delta_{ijk} - \gamma_{i'k} + \delta_{i'jk} = 0$$

  $\forall i, j, k, i' \in \{1, \ldots, n\}$ such that $(i, j, k) \neq (i_0, j_0, k_0)$, $(i, k) \neq (i_0, j_0)$ and $(i', k) \neq (i_0, j_0)$.
- *Case* 2: For $(i, k) = (i_0, j_0)$, we have $b_{jj_0} = \gamma_{i_0j_0} - \delta_{i_0jj_0} + e$ and $b_{jj_0} = \gamma_{ij_0} - \delta_{ijj_0}\ \forall i \neq i_0$. Then,

$$e = -\gamma_{i_0j_0} + \delta_{i_0jj_0} + \gamma_{ij_0} - \delta_{ijj_0}$$

  for $i, j \neq i_0, j_0$. Taking into account the definition of e, we must check that

$$\gamma_{ij_0} + \delta_{i_0jj_0} - \gamma_{i_0j_0} - \delta_{ijj_0} + \gamma_{i_0k_0} + \delta_{ij_0k_0} - \gamma_{ik_0} - \delta_{i_0j_0k_0} = 0$$

  $\forall j \neq i_0, j_0$ and any $i \neq k_0$
- *Case* 3: d does not depend on i:

$$\beta_{i_0} + \gamma_{ki'} - \sum_{r=1\ r\neq i_0,k}^{n} \delta_{ki_0r} - \delta_{ji_0k} - \beta_{i'} - \gamma_{ki_0} + \sum_{r=1\ r\neq i',k}^{n} \delta_{ki'r} + \delta_{ji'k} = 0$$

  $\forall i' \neq i_0, k, j \notin \{i_0, j_0, k_0\}$ must be checked.

- *Case* 4: For $i < j$, $(i, j), (j, i) \neq (i_0, j_0)$, we must first verify that

  $$a_{ij} = \alpha_{ji} - b_{ji} - c_{ji} = \alpha_{ij} - b_{ij} - c_{ij}$$

  and then that $\alpha_{ij} + \delta_{kij} - \gamma_{kj} + \gamma_{ij} = \alpha_{ji} + \delta_{kji} - \gamma_{ki} + \gamma_{ji}$ follows.
  Thus, if (i, j) or $(j, i) = (i_0, j_0)$, then

  $$\alpha_{i_0j_0} + \delta_{ii_0j_0} - \gamma_{ij_0} + \gamma_{i_0j_0} + e = \alpha_{j_0i_0} + \delta_{ij_0i_0} - \gamma_{ii_0} + \gamma_{j_0i_0}.$$

- *Proof Case* 1: We analyze 5 possibilities. For each of them, we identify multipliers λ and feasible solutions of $\mathcal{CR}$ which differ in their components and enable the proof.
  1. For $i, j, k, i' \notin \{i_0, j_0, k_0\}$, consider the paths

     1: $v_0\ v_{j_1} \dots j_{n-7}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_j\ v_i\ v_{i'}$
     2: $v_0\ v_{j_1} \dots j_{n-7}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_j\ v_{i'}\ v_i$
     3: $v_0\ v_{j_1} \dots j_{n-7}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_i\ v_j\ v_{i'}$
     4: $v_0\ v_{j_1} \dots j_{n-7}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_i\ v_{i'}\ v_j$
     5: $v_0\ v_{j_1} \dots j_{n-7}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_{i'}\ v_j\ v_i$
     6: $v_0\ v_{j_1} \dots j_{n-7}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_{i'}\ v_i\ v_j$
     7: $v_0\ v_{j_1} \dots v_{j_{n-7}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_j\ v_i\ v_{i'}\ v_k$
     8: $v_0\ v_{j_1} \dots v_{j_{n-7}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_j\ v_{i'}\ v_i\ v_k$
     9: $v_0\ v_{j_1} \dots v_{j_{n-7}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_i\ v_j\ v_{i'}\ v_k$
     10: $v_0\ v_{j_1} \dots v_{j_{n-7}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_i\ v_{i'}\ v_j\ v_k$
     11: $v_0\ v_{j_1} \dots v_{j_{n-7}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_{i'}\ v_j\ v_i\ v_k$
     12: $v_0\ v_{j_1} \dots v_{j_{n-7}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_{i'}\ v_i\ v_j\ v_k$

     where $\{v_{j_1}, \dots, v_{j_{n-7}}\} = V \backslash \{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_i, v_{i'}, v_j, v_k\}$.
     The condition thus follows from the linear combination given by
     $\lambda = (\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2})$.
  2. $\{i, j, k\} \cap \{i_0, j_0, k_0\} = \{p\}$.
     Take the paths and multipliers in the first case and delete vertex v_p from subpath $v_{i_0}\ v_{j_0}\ v_{k_0}$.
  3. $\{i, j, k\} \cap \{i_0, k_0\} = \{v_{p_1}, v_{p_2}\}$ or $\{i, j, k\} \cap \{j_0, k_0\} = \{v_{p_1}, v_{p_2}\}$.
     Similar to the previous case, deleting however vertices v_{p_1} and v_{p_2} from subpath $v_{i_0}\ v_{j_0}\ v_{k_0}$.
  4. $|\{i, j, k, i'\} \cap \{i_0, j_0\}| = 2$.
     Six possibilities are involved, respectively given by
     (a) $i = i_0$ y $j = j_0$

     1: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_{i'}\ v_k$
     2: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{i'}\ v_{i_0}\ v_{j_0}\ v_{k_0}$
     3: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k$
     4: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_{i'}$
     5: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{i'}\ v_{j_0}\ v_{i_0}\ v_{k_0}$
     6: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_k$
     7: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{j_0}\ v_{k_0}\ v_{i'}\ v_{i_0}$
     8: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{j_0}\ v_{k_0}\ v_{i'}\ v_{i_0}\ v_k$

     with $\{v_{j_1}, \dots, v_{j_{n-5}}\} = V \backslash \{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_{i'}, v_k\}$.
     Take $\lambda = (-1, 1, -1, 1, -1, 1, -1, 1)$.
     (b) $i = i_0$ y $i' = j_0$

     1: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_j\ v_k$
     2: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_j\ v_{j_0}\ v_k$
     3: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_j$
     4: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{i_0}\ v_{k_0}\ v_j\ v_{j_0}$
     5: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{k_0}\ v_{j_0}\ v_{i_0}\ v_j\ v_k$
     6: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{k_0}\ v_{j_0}\ v_{i_0}\ v_j$
     7: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{k_0}\ v_j\ v_{j_0}\ v_{i_0}\ v_k$
     8: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{k_0}\ v_j\ v_{j_0}\ v_{i_0}$

     with $\{v_{j_1}, \dots, v_{j_{n-5}}\} = V \backslash \{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_j, v_k\}$.
     Take $\lambda = (1, -1, -1, 1, -1, 1, 1, -1)$.
     (c) $i = j_0$ y $j = i_0$

     1: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_{i'}\ v_{j_0}\ v_k$
     2: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_k\ v_{k_0}\ v_{i'}\ v_{j_0}$
     3: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{i'}\ v_{i_0}\ v_{k_0}\ v_{j_0}$
     4: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{i_0}\ v_k\ v_{k_0}\ v_{j_0}$
     5: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_{i'}\ v_k$
     6: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{j_0}\ v_{i_0}\ v_k\ v_{k_0}\ v_{i'}$
     7: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_k\ v_{i'}\ v_{j_0}\ v_{i_0}\ v_{k_0}$
     8: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{j_0}\ v_{i_0}\ v_k\ v_{k_0}$

     with $\{v_{j_1}, \dots, v_{j_{n-5}}\} = V \backslash \{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_{i'}, v_k\}$.
     Take $\lambda = (1, -1, -1, 1, -1, 1, 1, -1)$.
     (d) $i = j_0$ y $k = i_0$

     1: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_{i'}\ v_j$
     2: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_j\ v_{j_0}\ v_{i'}$
     3: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_j$
     4: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{i_0}\ v_{k_0}\ v_j\ v_{j_0}$
     5: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{k_0}\ v_{j_0}\ v_{i'}\ v_j\ v_{i_0}$
     6: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{k_0}\ v_j\ v_{j_0}\ v_{i'}\ v_{i_0}$
     7: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{k_0}\ v_{j_0}\ v_j\ v_{i_0}$
     8: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{k_0}\ v_j\ v_{j_0}\ v_{i_0}$

     with $\{v_{j_1}, \dots, v_{j_{n-5}}\} = V \backslash \{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_{i'}, v_j\}$.
     Take $\lambda = (-1, 1, 1, -1, 1, -1, -1, 1)$.
     (e) $j = i_0$ y $k = j_0$

     1: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_i\ v_{i'}$
     2: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_i\ v_{i'}\ v_{j_0}$
     3: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_i\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_{i'}$
     4: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_i\ v_{i'}\ v_{i_0}\ v_{j_0}\ v_{k_0}$
     5: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_i$
     6: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{i'}\ v_{i_0}\ v_{k_0}\ v_i\ v_{j_0}$
     7: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{j_0}\ v_i\ v_{i_0}\ v_{k_0}\ v_{i'}$
     8: $v_0\ v_{j_1} \dots v_{j_{n-5}}\ v_{j_0}\ v_i\ v_{i'}\ v_{i_0}\ v_{k_0}$

with $\{v_{j_1}, \ldots, v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_{i'}, v_i\}$.
Take $\lambda = (1, -1, -1, 1, -1, 1, 1, -1)$.

(f) $j = j_0$ y $k = i_0$

1: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_{i'}\ v_i$
2: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{i_0}\ v_{k_0}\ v_{i'}\ v_i\ v_{j_0}$
3: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_i\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_{i'}$
4: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_i\ v_{i_0}\ v_{k_0}\ v_{i'}\ v_{j_0}$
5: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{k_0}\ v_{j_0}\ v_{i'}\ v_i\ v_{i_0}$
6: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{k_0}\ v_{i'}\ v_i\ v_{j_0}\ v_{i_0}$
7: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_i\ v_{k_0}\ v_{j_0}\ v_{i'}\ v_{i_0}$
8: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_i\ v_{k_0}\ v_{i'}\ v_{j_0}\ v_{i_0}$

with $\{v_{j_1}, \ldots, v_{j_{n-5}}\} = V\backslash\{i_0, j_0, k_0, i', i\}$.
Take $\lambda = (-1, 1, 1, -1, 1, -1, -1, 1)$.

5. $\{i, j, i', k\} \cap \{i_0, j_0, k_0\} = 3$
Consider the paths

1: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_l$
2: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_{j_0}\ v_l\ v_{k_0}$
3: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_l$
4: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_{k_0}\ v_l\ v_{j_0}$
5: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_l\ v_{j_0}\ v_{k_0}$
6: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_l\ v_{k_0}\ v_{j_0}$
7: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_l$
8: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{j_0}\ v_{i_0}\ v_l\ v_{k_0}$
9: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{j_0}\ v_{k_0}\ v_{i_0}\ v_l$
10: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{j_0}\ v_{k_0}\ v_l\ v_{i_0}$
11: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{j_0}\ v_l\ v_{i_0}\ v_{k_0}$
12: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{j_0}\ v_l\ v_{k_0}\ v_{i_0}$
13: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{k_0}\ v_{i_0}\ v_l\ v_{j_0}$
14: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{k_0}\ v_{j_0}\ v_{i_0}\ v_l$
15: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{k_0}\ v_{j_0}\ v_l\ v_{i_0}$
16: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{k_0}\ v_l\ v_{j_0}\ v_{i_0}$
17: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_l\ v_{i_0}\ v_{j_0}\ v_{k_0}$
18: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_l\ v_{i_0}\ v_{k_0}\ v_{j_0}$
19: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_l\ v_{j_0}\ v_{i_0}\ v_{k_0}$
20: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_l\ v_{j_0}\ v_{k_0}\ v_{i_0}$
21: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_l\ v_{k_0}\ v_{j_0}\ v_{i_0}$

where $\{v_{j_1}, \ldots, v_{j_{n-4}}\} = V\backslash\{i_0, v_{j_0}, k_0, l\}$.
We now present multipliers for each of the 9 different possibilities involved.

(a) $i = i_0, j = j_0, i' = k_0$ y $l = k$.
$\lambda = (-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -1, 0, 0, \frac{1}{2}, 0, \frac{1}{2}, 0, 1, -\frac{1}{2}, -\frac{1}{2}, 1, -1, 1, -\frac{1}{2}, -\frac{1}{2})$.

(b) $i = i_0, k = j_0, i' = j_0$ y $l = j$.
$\lambda = (-\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, 0, -1, 0, \frac{1}{2}, 0, \frac{1}{2}, 0, 1, -\frac{1}{2}, -\frac{1}{2}, 0, 0, 1, -\frac{1}{2}, -\frac{1}{2})$.

(c) $i = i_0, j = k_0$ y $i' = j_0$ y $l = k$
$\lambda = (\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -1, 0, 0, \frac{1}{2}, 0, \frac{1}{2}, 0, 1, -\frac{1}{2}, -\frac{1}{2}, 0, 0, 1, -\frac{1}{2}, -\frac{1}{2})$.

(d) $i = j_0, j = i_0, k = k_0$ y $l = i'$
$\lambda = (\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, 0, 0, 0, -\frac{1}{2}, 1, -\frac{1}{2}, 0, 0, -\frac{1}{2}, \frac{1}{2}, 0, 0, -1, \frac{1}{2}, \frac{1}{2})$.

(e) $i = j_0, j = i_0, i' = k_0$ y $l = k$
$\lambda = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, 0, 0, 1, -\frac{1}{2}, 0, -\frac{1}{2}, 0, -1, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, -\frac{1}{2}, \frac{1}{2})$.

(f) $i = j_0, k = i_0, i' = k_0$ y $l = j$.
$\lambda = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, 0, 0, 0, \frac{1}{2}, 0, -\frac{1}{2}, 0, 0, -\frac{1}{2}, \frac{1}{2}, 0, 0, 0, -\frac{1}{2}, \frac{1}{2})$.

(g) $i = j_0, j = k_0, k = i_0$ y $l = i'$.
$\lambda = (\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, 0, 0, 0, -\frac{1}{2}, 0, \frac{1}{2}, 0, 0, -\frac{1}{2}, \frac{1}{2}, 0, 0, 0, -\frac{1}{2}, \frac{1}{2})$.

(h) $i = k_0, j = i_0, k = j_0$ y $l = i'$.
$\lambda = (-\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, -1, 1, -\frac{1}{2}, -1, \frac{1}{2}, -1, 0, \frac{1}{2}, \frac{1}{2}, 0, 1, 0, -\frac{1}{2}, -\frac{1}{2})$.

(i) $i = k_0, j = j_0, k = i_0$ y $l = i'$
$\lambda = (\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, 0, 0, 0, -\frac{1}{2}, 0, \frac{1}{2}, 0, 0, -\frac{1}{2}, \frac{1}{2}, 0, 0, 0, \frac{1}{2}, -\frac{1}{2})$.

- *Proof Case* 2: Two cases are involved.

1. For $j = k_0$, consider the following paths

1: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_i\ v_{j_0}\ v_{k_0}$
2: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_{i_0}\ v_i\ v_{k_0}\ v_{j_0}$
3: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_i\ v_{i_0}\ v_{j_0}\ v_{k_0}$
4: $v_0\ v_{j_1} \ldots v_{j_{n-4}}\ v_i\ v_{i_0}\ v_{k_0}\ v_{j_0}$

with $\{v_{j_1}, \ldots, v_{j_{n-4}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_i\}$.
If $\lambda = (-1, 1, 1, -1)$ is taken, the condition follows.

2. For $j \neq k_0$, consider the following paths

1: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_i\ v_{i_0}\ v_j\ v_{k_0}\ v_{j_0}$
2: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_j\ v_{i_0}\ v_i\ v_{j_0}\ v_{k_0}$
3: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_j\ v_{i_0}\ v_i\ v_{k_0}\ v_{j_0}$
4: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_j\ v_i\ v_{i_0}\ v_{j_0}\ v_{k_0}$
5: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{j_0}\ v_i\ v_{i_0}\ v_j\ v_{k_0}$
6: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{j_0}\ v_j\ v_i\ v_{i_0}\ v_{k_0}$
7: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{j_0}\ v_i\ v_j\ v_{k_0}\ v_{i_0}$
8: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_{j_0}\ v_j\ v_i\ v_{k_0}\ v_{i_0}$
9: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_i\ v_j\ v_{k_0}\ v_{j_0}\ v_{i_0}$
10: $v_0\ v_{j_1} \ldots v_{j_{n-5}}\ v_j\ v_i\ v_{k_0}\ v_{j_0}\ v_{i_0}$

with $\{v_{j_1}, \ldots, v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_i, v_j\}$.
If $\lambda = (1, 1, -1, -1, -1, 1, 1, -1, -1, 1)$ is taken, the condition follows.

- *Proof Case* 3: We consider 3 possibilities
  1. For $i' = j_0$, consider the paths

$$\begin{array}{ll} 1{:}\ v_0\ v_{i_0}\ v_{j_0}\ v_j\ v_{k_0}\ v_k\ v_{j_1}\dots v_{j_{n-5}} & 5{:}\ v_0\ v_{j_0}\ v_j\ v_{k_0}\ v_k\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \\ 2{:}\ v_0\ v_k\ v_{i_0}\ v_{j_0}\ v_j\ v_{k_0}\ v_{j_1}\dots v_{j_{n-5}} & 6{:}\ v_0\ v_k\ v_{j_0}\ v_j\ v_{k_0}\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \\ 3{:}\ v_0\ v_k\ v_j\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_{j_1}\dots v_{j_{n-5}} & 7{:}\ v_0\ v_k\ v_j\ v_{j_0}\ v_{k_0}\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \\ 4{:}\ v_0\ v_j\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_{j_1}\dots v_{j_{n-5}} & 8{:}\ v_0\ v_j\ v_{j_0}\ v_{k_0}\ v_k\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \end{array}$$

     with $\{v_{j_1},\dots,v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_j, v_k\}$.
     Taking $\lambda = (1,-1,1,-1,-1,1,-1,1)$, the condition follows.
  2. For $i' = k_0$, consider the paths

$$\begin{array}{ll} 1{:}\ v_0\ v_{i_0}\ v_{k_0}\ v_j\ v_{j_0}\ v_k\ v_{j_1}\dots v_{j_{n-5}} & 5{:}\ v_0\ v_{k_0}\ v_j\ v_{j_0}\ v_k\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \\ 2{:}\ v_0\ v_k\ v_{i_0}\ v_{k_0}\ v_j\ v_{j_0}\ v_{j_1}\dots v_{j_{n-5}} & 6{:}\ v_0\ v_k\ v_{k_0}\ v_j\ v_{j_0}\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \\ 3{:}\ v_0\ v_k\ v_j\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_{j_1}\dots v_{j_{n-5}} & 7{:}\ v_0\ v_k\ v_j\ v_{k_0}\ v_{j_0}\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \\ 4{:}\ v_0\ v_j\ v_{i_0}\ v_{k_0}\ v_{j_0}\ v_k\ v_{j_1}\dots v_{j_{n-5}} & 8{:}\ v_0\ v_j\ v_{k_0}\ v_{j_0}\ v_k\ v_{i_0}\ v_{j_1}\dots v_{j_{n-5}} \end{array}$$

     with $\{v_{j_1},\dots,v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_j, v_k\}$.
     Taking $\lambda = (1,-1,1,-1,-1,1,-1,1)$, the condition follows.
  3. For $i' \notin \{j_0, k_0\}$, consider the paths

$$\begin{array}{ll} 1{:}\ v_0\ v_{i_0}\ v_{i'}\ v_j\ v_{j_0}\ v_k\ v_{k_0}\ v_{j_1}\dots v_{j_{n-6}} & 5{:}\ v_0\ v_{i'}\ v_j\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_k\ v_{j_1}\dots v_{j_{n-6}} \\ 2{:}\ v_0\ v_k\ v_{i_0}\ v_{i'}\ v_j\ v_{j_0}\ v_{k_0}\ v_{j_1}\dots v_{j_{n-6}} & 6{:}\ v_0\ v_k\ v_{i'}\ v_j\ v_{j_0}\ v_{k_0}\ v_{i_0}\ v_{j_1}\dots v_{j_{n-6}} \\ 3{:}\ v_0\ v_k\ v_j\ v_{i_0}\ v_{i'}\ v_{j_0}\ v_{k_0}\ v_{j_1}\dots v_{j_{n-6}} & 7{:}\ v_0\ v_k\ v_j\ v_{i'}\ v_{j_0}\ v_{k_0}\ v_{i_0}\ v_{j_1}\dots v_{j_{n-6}} \\ 4{:}\ v_0\ v_j\ v_{i_0}\ v_{i'}\ v_{j_0}\ v_k\ v_{k_0}\ v_{j_1}\dots v_{j_{n-6}} & 8{:}\ v_0\ v_j\ v_{i'}\ v_{j_0}\ v_{i_0}\ v_{k_0}\ v_k\ v_{j_1}\dots v_{j_{n-6}} \end{array}$$

     with $\{v_{j_1},\dots,v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_{i'}, v_j, v_k\}$.
     Taking $\lambda = (1,-1,1,-1,-1,1,-1,1)$, the condition follows.
- *Proof Case* 4: Possibilities involved are:
  1. $i, j \notin \{i_0, j_0, k_0\}$: We must prove that

$$\alpha_{ij} + \delta_{kij} - \gamma_{kj} + \gamma_{ij} = \alpha_{ji} + \delta_{kji} - \gamma_{ki} + \gamma_{ji}.$$

     In order to do so, consider the paths

$$1{:}\ v_0\ v_{j_1}\dots v_{j_{n-6}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_i\ v_j \quad 2{:}\ v_0\ v_{j_1}\dots v_{j_{n-6}}\ v_{i_0}\ v_{j_0}\ v_{k_0}\ v_k\ v_j\ v_i$$

     with $\{v_{j_1},\dots,v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_i, v_j, v_k\}$ and take $\lambda = (1,-1)$.
  2. $\{i,j\} \cap \{i_0, j_0, k_0\} = \{p\}$.
     Consider the paths and multipliers from the previous case and delete vertex v_p from subpath $v_{i_0}, v_{j_0}, v_{k_0}$.
  3. $\{i,j\} = \{i_0, k_0\}$ or $\{i,j\} = \{j_0, k_0\}$.
     Consider the paths and multipliers from the previous case and delete vertices v_i and v_j from subpath $v_{i_0}, v_{j_0}, v_{k_0}$.
  4. $\{i,j\} = \{i_0, j_0\}$.
     We must prove that

$$\alpha_{i_0j_0} + \delta_{i_{i_0}j_0} - \gamma_{ij_0} + \gamma_{i_0j_0} + e = \alpha_{j_0i_0} + \delta_{ij_0i_0} - \gamma_{ii_0} + \gamma_{j_0i_0}.$$

     In order to do so, consider the paths:

$$\begin{array}{ll} 1{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_{k_0}\ v_{i_0}\ v_i\ v_j\ v_{j_0} & 5{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_{j_0}\ v_j\ v_{k_0}\ v_{i_0}\ v_i \\ 2{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_{k_0}\ v_i\ v_{i_0}\ v_j\ v_{j_0} & 6{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_{j_0}\ v_{k_0}\ v_i\ v_{i_0}\ v_j \\ 3{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_j\ v_{k_0}\ v_{i_0}\ v_i\ v_{j_0} & 7{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_{j_0}\ v_j\ v_{k_0}\ v_i\ v_{i_0} \\ 4{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_{j_0}\ v_{k_0}\ v_{i_0}\ v_i\ v_j & 8{:}\ v_0\ v_{j_1}\dots v_{j_{n-5}}\ v_j\ v_{k_0}\ v_i\ v_{j_0}\ v_{i_0} \end{array}$$

     with $\{v_{j_1},\dots,v_{j_{n-5}}\} = V\backslash\{v_0, v_{i_0}, v_{j_0}, v_{k_0}, v_i, v_j\}$.
     Taking $\lambda = (-1,1,1,1,-1,-1,1,-1)$, the condition follows. □

Additional valid inequalities are presented next. They are facet-defining for $\mathcal{CR}$. However, to simplify the presentation, we omit corresponding proofs since they run along the same lines of the proofs previously presented in this paper.

Proposition 3. *For $i, j, k, l = 1,\dots,n$, $i \neq j, k, l$, $j \neq k, l$, $k \neq l$, inequalities*

$$f_{ij}^j + f_{il}^j + f_{kj}^l \leq x_{ij} + f_{kj}^i$$
$$f_{ij}^j + f_{il}^j + f_{kl}^j \leq x_{ij} + f_{kl}^i$$
$$f_{0i}^i + f_{ij}^j \leq x_{ij} + f_{ij}^k$$
$$f_{ik}^j + f_{ki}^j + f_{il}^k + f_{ij}^k + f_{jl}^k + f_{jk}^k \leq x_{ij} + x_{jk}$$

are facet-defining for $\mathcal{CR}$.

Proposition 4. *For $i, j, k = 1, \ldots, n$, $i \neq j, k$, $j \neq k$, inequalities*

$$f_{ki}^{j} + f_{ik}^{j} + f_{jk}^{k} \leq x_{ij} + f_{jk}^{i}$$
$$f_{ij}^{k} + f_{ki}^{j} + f_{ik}^{j} \leq x_{ij}$$
$$f_{jk}^{k} + f_{ik}^{j} + f_{ij}^{j} \leq x_{ij} + f_{ij}^{k} + f_{jk}^{i}$$

are facet-defining for $\mathcal{CR}$.

4.2. Inequalities derived from LOP

Two families of facet defining inequalities for the LOP polytope, denoted respectively 3-Cycle and k-Fence, were introduced in Grötschel, Jünger and Reinelt [14]. Valid inequalities for $\mathcal{CR}$, obtained after lifting 3-Cycle and k-Fence inequalities, are presented next.

4.2.1. 3-cycle inequalities

The 3-Cycle inequalities

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \quad i, j, k = 1, \ldots, n, i \neq j, k, j \neq k$$

are valid for the $\mathcal{CR}$. They are not, however, facet-defining for that polytope. This holds true since $x_{ij} + x_{jk} + x_{ki} = 2$ implies $f_{ji}^{l} = f_{kj}^{l} = f_{ik}^{l} = 0$ for all $l = 1, \ldots, n$. However, a lifting procedure may be applied to 3-Cycle inequalities (considering different orderings of variables for sequential lifting) to obtain families of inequalities that are facet defining for $\mathcal{CR}$. This applies to the two families of inequalities that follow:

Proposition 5. *Inequality*

$$x_{ij} + x_{jk} + x_{ki} + f_{ik}^{j} + f_{ji}^{i} \leq 2$$

is facet-defining for $\mathcal{CR}$.

Proposition 6. *Inequality*

$$x_{ij} + x_{jk} + x_{ki} + f_{kj}^{i} + f_{ik}^{j} + f_{ji}^{k} \leq 2.$$

is facet-defining for $\mathcal{CR}$.

Assume now that vertex sets $U = \{u_1, \ldots, u_k\}$ and $W = \{w_1, \ldots, w_k\}$, of cardinality $k \geq 3$, are given. In association with U and W, define an arc set $A = \cup_{i=1}^{k}(\{(u_i, w_i)\} \cup \{(w_i, v) | v \in U \backslash \{u_i\}\})$. A k-Fence inequality is then given by

$$\sum_{(ij)\in A} x_{ij} \leq k^2 - k + 1.$$

The above inequalities are valid for our formulation. However they are not facet defining for $\mathcal{CR}$. If $\sum_{(ij)\in A} x_{ij} = k^2 - k + 1$ implies, for instance, $f_{0u_i}^{u_i} = 0, \forall i = 1, \ldots, k$. Applying, once again, sequential lifting over these variables and also over distinct variables $f_{w_i u_i}^{u_i} \forall i = 1, \ldots, k$, a stronger valid inequality

$$(k-2)\sum_{i=1}^{k} f_{0u_i}^{u_i} + \sum_{i=1}^{k} f_{w_i u_i}^{u_i} + \sum_{i=1}^{k} x_{u_i w_i} + \sum_{i=1}^{k}\sum_{j=1}^{k}{}_{j\neq i} x_{w_i u_j} \leq k^2 - k + 1$$

results. Although stronger, this inequality is still not facet-defining for $\mathcal{CR}$.

5. Computational experiments

Valid inequalities introduced in the previous section for $\mathcal{CR}$, are used here to reinforce TDP LP relaxation bounds. The stronger bounds thus obtained are then embedded into a Branch and Bound algorithm for TDP.

5.1. A cutting plane algorithm

Assume that one intends to implement a cutting plane algorithm to TDP. Assume as well that such an algorithm should be based on the families of valid TDP inequalities described in previous sections. Thus, given a fractional LP relaxation solution, a corresponding *separation problem* must be solved for each different family of inequalities involved. In doing so, as a solution

Table 3
Gap improvement by cutting plane algorithm

Inst.	Relax.		Set 0		Set 0 + Set 1		Set 0 + Set 2		Set 0 + Set 1 + Set 2	
	%Gap	Time	%Gap	Time	%Gap	Time	%Gap	Time	%Gap	Time
D-20	17.58	0.09	13.84	6.36	4.46	7.48	9.07	8.91	4.46	6.35
D-22	22.75	0.11	14.98	6.46	0.54	27.84	8.27	9.2	0.5	21.08
D-24	22.97	0.17	16.14	10.21	0.65	70.21	9.18	14.35	0.61	50.57
D-26	22.12	0.22	15.76	14.96	0.32	125.77	8.16	23.62	0.25	100
D-28	23.29	0.4	17.32	22.68	0.74	230.62	10.02	33.6	0.68	166.94
D-30	21.78	0.48	15.67	32.45	0.71	280.36	9.03	52.59	0.64	221.39
D-35	24.58	0.91	20.24	118.08	1.36	1233.72	12.35	159.94	1.29	694.5
D-40	26.2	2.01	20.9	318.25	1.66	3481.09	11.97	402.21	1.54	2090.91
Average	22.66	0.55	16.86	66.18	1.3	682.14	9.76	88.05	1.25	418.97
S-20	35.84	0.08	17.64	6.23	0	8.55	8.53	9.67	0	8.69
S-22	28.03	0.13	17.06	6.32	0	20.94	10.33	9.57	0	21.42
S-24	43.67	0.19	26.6	12.38	0.25	61.92	16.91	19.37	0.11	63.17
S-26	36.52	0.27	20.58	17.51	0.3	50.12	12.2	28.2	0.23	52.99
S-28	38.77	0.42	23.58	30.17	0.12	117.54	14.43	50.45	0.05	124.01
S-30	38.76	0.58	24.63	47.35	0.36	216.72	15.42	77.67	0.22	221.22
S-35	46.4	1.04	29.96	154.17	0.2	729.99	18.3	245.79	0.07	704.96
S-40	44.84	2.71	31.28	511.89	1.69	2428.75	20.19	676.99	1.35	2566.65
Average	39.1	0.68	23.92	98.25	0.37	454.32	14.54	139.71	0.25	470.39
A-20	10.33	0.14	2.8	3.94	0	3.27	0.8	5.58	0	3.28
A-22	11.2	0.2	3.4	4.69	0	4.4	1.22	5.54	0	4.46
A-24	13.48	0.25	5.7	6.91	0.18	16.55	3.41	9.59	0.15	16.1
A-26	11.78	0.44	3.27	13.3	0	14.05	1.06	18.69	0	14.17
A-28	11.89	0.74	5.01	20.75	0.52	63.74	3	27.92	0.47	64.06
A-30	14.43	0.79	6.53	32.53	0.61	66.09	3.1	48.06	0.58	69.46
A-35	16.36	1.56	3.27	92.29	0.11	183.92	1.75	133.13	0.1	176.04
A-40	17.37	5.78	9.43	252.13	1.97	993.85	6.66	390.66	1.9	1043.7
Average	13.35	1.24	4.93	53.32	0.42	168.23	2.62	79.9	0.4	173.91

to these problems, one should either exhibit an inequality that is violated at the given fractional solution or else obtain a certificate that no such inequality exists.

For any family of valid TDP inequalities in Section 4, only a polynomial number of inequalities exist. As a result, corresponding separation problems could be efficiently tackled through direct enumeration. Additionally, for the case of lifted k-Fence inequalities, after some computational testing, we will only use inequalities with $k = 3$ in the experiments that follow. A greedy heuristic is used for the separation of 3-Fence inequality. Given the current LP relaxation solution (x^*, f^*), we search for three arcs (u_1, w_2), (u_2, w_2) and (u_3, w_3), having no endnode in common such that $0.4 \leq f^{u_i}_{w_i u_i}, x_{u_i w_i} \leq 0.6$. Then, we check if the 3-Fence inequality associated with these arcs is violated.

In order to evaluate the strength of the inequalities to be investigated, TDP lower bounds returned by the cutting plane algorithm are compared with the original LP relaxation bounds. Furthermore, we now use, as cutting planes, those inequalities that were previously dropped from the LP relaxation tested in Section 3. Two stopping criteria are used in the experiments. In accordance with these, the cutting plane algorithm either stops if no violated inequality is identified or else if half an hour of CPU time is spent.

For the results in Table 3, valid inequalities are divided in three large sets. The first one, denoted Set 0, only contains the inequalities dropped from the original LP relaxation (see Section 3). The second set, denoted Set 1, only contains those inequalities in Section 4 that directly originate from the TDP formulation in Section 2. The third set, denoted Set 2, only contains those inequalities that originate from LOP. Four types of experiments were conducted. The first one restricts the cutting plane algorithm to using only additional inequalities from Set 0. Likewise, the second experiment is restricted to only using additional inequalities from Set 0 and Set 1. For the third experiment, only additional inequalities from Set 0 and Set 2 are used. Finally, for the fourth experiment, inequalities from any of the three sets defined above may be used.

Results on Table 3 indicate that substantial TDP lower bound improvements are attained through the use of the valid inequalities suggested in this study. That proved particularly true for those inequalities in Set 1. However, for some of the instances tested, inequalities in Set 2 succeeded in further improving Set 1 lower bounds. Furthermore, in most of the instances that happened with lower CPU time demand, and only a few cases incurred into negligible additional time. As a result, we concluded that the combined use of inequalities from Set 0, Set 1 and Set 2 is likely to attain the best computational results in an implicit enumeration scheme to solve TDP.

5.2. Branch-and-Bound vs ***BC-R***

Some additional experiments were conducted to evaluate the potential benefits of using, within an implicit enumeration scheme, the cutting plane algorithm described above. As part of that experiment, a Branch and Bound algorithm where

Table 4
BC-R vs Branch-and-Bound in random instances

n	***BC-R***		*B-and-B*	
	Time	Nodes	Time	Nodes
D-20	7.72	0.00	87.30	23.00
D-22	43.18	1.20	478.85	115.60
D-24	108.66	1.60	3276.89	430.00
D-26	155.30	2.00	4608.42	429.25
Average	78.72	1.20	2112.87	249.46
S-20	8.68	0.00	104.50	29.20
S-22	20.80	0.00	309.43	79.6
S-24	68.18	0.80	3866.27	479.20
S-26	60.08	0.40	2686.28	286.40
Average	39.43	0.30	1741.62	218.60
A-20	3.35	0.00	8.22	2.00
A-22	4.52	0.00	11.79	3.60
A-24	17.05	0.40	97.12	23.00
A-26	14.48	0.00	34.60	4.40
Average	9.85	0.10	37.93	8.25

cutting planes are used at the nodes of the enumeration tree, was implemented. The resulting algorithm, denoted ***BC-R***, was compared with a pure Branch and Bound algorithm, where no cutting planes are used. For convenience, we simply denote by Branch and Bound the comparison algorithm. In practical terms, we have compared ***BC-R*** with itself, after switching off its cutting plane subroutine. The search strategy used was Depth-First while the pseudo reduced cost branching variable selection was applied.

Comparisons were carried out over 5 instances, for each value of n considered, where $n \leq 26$. In order to solve instances with $n > 26$, the use of cutting planes appears essential, if one is to comply with the 2 h CPU time deadline limit imposed. Results on Table 4, clearly indicate that ***BC-R*** attains a better performance than Branch and Bound.

5.3. CPLEX vs ***BC-R***

After implementing an ad-hoc Branch and Bound algorithm such as ***BC-R***, it appears quite natural to compare it with a general purpose Mixed Integer Programming solver such as CPLEX [7]. In doing so, it should be stressed that CPLEX benefits from features such as preprocessing, clique cuts and cover cuts.

Instances with n up to 40 were used in the experiments. Table 5 presents the average results obtained for every value of n considered (over the 5 instances tested in each case). Specifically, CPU times and number of tree nodes explored appear on that table, where symbol ***** is used to indicate that the corresponding instance could not be solved within the 2.5 h CPU time limit imposed.

As it can be appreciated from the results on Table 5, within the time limit imposed, instances up to 26 clients could be solved to optimality by both algorithms. But, CPLEX CPU time requirements are significantly higher. CPLEX could only solve instances with 28 clients in the assymetric case. Once again, the CPU time costs were higher than ***BC-R*** time costs. A much better performance was attained by ***BC-R*** which managed to solve to proven optimality all instances tested within the CPU time limit imposed.

5.4. Non randomly generated test instances

So far, TDP solution algorithms have basically been tested on randomly generated instances such as the ones used in this study. The only exception to that is found in Wu, Huang, and Zhan [23] where 4 non randomly generated TSPLIB instances (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/) are used in computational experiments. In line with [23], we now present computational results for 13 non randomly generated TSPLIB instances. It should be stressed that, contrary to most TDP solution algorithms, ours do not require test instances to be geometric. It is thus more general than their counterparts. However, generality comes at a price and, specifically for geometric instances, our algorithm looses strength for not being geared to that type of instance.

Computational results are presented in Table 6 and involve instances with up to 40 vertices. The first 9 instances in that table are symmetric while the 3 remaining ones are asymmetric. Comparing results in Table 6 with those we previouly obtained, it becomes evident that the non randomly generated instances appear much harder for our algorithm to solve than randomly generated ones. Reasons for that are probably associated with the fact that, in addition to being geometric, these *TSPLIB* instances are much more structured. As one may recall, our algorithm is not geared to this type of instance.

Table 5
BC-R vs CPLEX in random instances

	BC-R		Cplex	
	Time	Nodes	Time	Nodes
D-20	7.72	0.00	108.10	30.20
D-22	43.18	1.20	472.38	115.60
D-24	108.66	1.60	3216.22	438.00
D-26	155.30	2.00	4629.42	429.25
D-28	424.32	4.80	****	****
D-30	448.59	3.20	****	****
D-35	1790.53	5.20	****	****
D-40	8097.56	14.00	****	****
Average	1384.48	4.00	****	****
S-20	8.68	0.00	103.71	28.40
S-22	20.80	0.00	296.20	79.60
S-24	68.18	0.80	3617.51	479.20
S-26	60.08	0.40	2704.52	286.40
S-28	120.79	0.40	****	****
S-30	248.59	0.80	****	****
S-35	682.79	0.40	****	****
S-40	6011.65	11.2	****	****
Average	902.69	1.75	****	****
A-20	3.35	0.00	8.29	2.00
A-22	4.52	0.00	11.73	3.60
A-24	17.05	0.40	94.64	23.00
A-26	14.48	0.00	36.07	4.40
A-28	81.09	1.60	490.21	54.2
A-30	132.54	2.40	****	****
A-35	198.69	0.80	****	****
A-40	5358.86	16.20	****	****
Average	726.32	2.68	****	****

Table 6
BC-R vs CPLEX vs Branch-and-Bound in real data

	BC-R		Cplex		*B-and-B*	
	Time	Nodes	Time	Nodes	Time	Nodes
bayg29	5334.55	56	****	****	****	****
bays29	1440.34	18	****	****	****	****
burma14	0.61	0	1.54	2	1.63	2
fri26	293.74	6	3580.84	346	3542.64	346
gr17	22.44	2	60.04	32	64.28	42
gr21	22.57	0	317.50	78	355.88	80
gr24	18.06	0	482.57	96	484.24	96
ulysses16	64.11	8	177.49	126	267.63	164
ulysses22	1190.91	34	****	****	****	****
br17	3979.86	196	****	****	****	****
ftv33	5029.67	28	****	****	****	****
ftv35	8585.87	38	****	****	****	****
ftv38	3231.61	8	****	****	****	****

6. Conclusions and future work

A new TDP formulation and some associated valid inequalities were introduced in this paper. LP relaxation bounds for that formulation proved much stronger than two other TDP lower bounds proposed in the literature. Furthermore, considerable additional lower bound gains are attained through the use of the valid inequalities proposed here.

TDP is clearly a very challenging problem to solve to proven optimality. Having that in mind, the results obtained in this paper indicate that a cutting plane algorithm, like the one suggested here, is an attractive proposition for generating good quality lower bounds for that problem. Moreover, the proposed cutting plane algorithm, operating at the nodes of a Branch and Bound tree, has shown to be particularly useful in substantially reducing the number of Branch and Bound nodes, as well as CPU time.

We hope that the results obtained here stimulate further research on polyhedral algorithms to solve TDP. For that problem, this kind of investigation is barely starting and this paper is the very first contribution in that direction.

Acknowledgements

The first and second author's research were partially supported by grants UBACYT Grant X212, PICT Grant 01109112. The third author's research was partially supported by grants CNPq-PROSUL 4900333/04-4, CNPq 300149/94-08, CAPES/SPU n. 008/02, and FAPERJ E-26/170.187/2005.

References

[1] F. Afrati, S. Cosmadakis, C. Papadimitriou, G. Papageorgiou, N. Papakostantinou, The complexity of the travelling repairman problem, RAIRO Informatique Theorique et Applications 20 (1986) 79–87.
[2] A. Archer, D. Williamson, Faster approximation algorithms for the minimum latency problem, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, 2003.
[3] S. Arora, G. Karakostas, Approximation schemes for minimum latency problem, in: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, 1993, pp. 688–693.
[4] G. Ausiello, S. Leonardi, A. Marchetti-Spaccamela, On salesmen, repairmen, spiders and other traveling agents, in: Proc. of the Italian Conference on Algorithms and Complexity, 2000, pp. 1–16.
[5] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, in: Proceedings of 26th ACM Symp. on Theory of Computing, STOC, 1994, pp. 163–171.
[6] V. Chvatal, Hamiltonian cycles, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Eds.), The Traveling Salesman Problem — A Guided Tour of Combinatorial Optimization, Wiley & Sons, Chichester, 1985, pp. 403–429.
[7] CPLEX Linear Optimization 8.1 with Mixed Integer & Barrier Solvers, ILOG, 1997–1998.
[8] C.A. van Eijl, A polyhedral approach to the delivery man problem, Memorandum COSOR 95-19, Eindhoven University of Technology, 1995.
[9] E. Feuerstein, L. Stougie, On-line single-server dial-a-ride problems, Theoretical Computer Science 268 (1) (2001) 91–105.
[10] M. Fischetti, G. Laporte, S. Martello, The delivery man problem and cumulative matroids, Operations Research 41 (6) (1993) 1055–1064.
[11] A. García, P. Jodrá, J. Tejel, A note on the travelling repairman problem, Pre-publications del seminario mathematico 3, University of Zaragoza, Spain, 2001.
[12] M. Goemans, J. Kleinberg, An improved approximation ratio for the minimum latency problem, Mathematical Programming 82 (1998) 111–124.
[13] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, SIAM Journal on Computing 24 (1995) 296–317.
[14] M. Grötschel, M. Jünger, G. Reinelt, Facets of the linear ordering polytope, Mathematical Programming 69 (1985) 43–60.
[15] E. Koutsoupias, C.H. Papadimitriou, M. Yannakakis, Searching a Fixed graph, ICALP, 1996, pp. 280–289.
[16] S.O. Krumke, W.E. de Paepe, D. Poensgen, L. Stougie, News from the online traveling repairman, Theoretical Computer Science 295 (1–3) (2003) 279–294.
[17] A. Lucena, Time-dependent traveling salesman problem — The deliveryman case, Networks 20 (1990) 753–763.
[18] J. Picard, M. Queyranne, The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling, Operations Research 26 (1) (1978) 86–110.
[19] S. Sahni, T. Gonzalez, P-complete approximation problems, Journal of the Association for Computing Machinery 23 (1976) 555–565.
[20] D. Simchi-Levi, O. Berman, Minimizing the total flow time of n jobs on a network, IIE Transactions 23 (1991) 236–244.
[21] R. Sitters, The minimum latency problem is NP-hard for weighted trees, in: Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization, 2002, pp. 230–239.
[22] R.J. Vander Wiel, N.V. Sahinidis, Heuristic bounds and test problem generation for the time-dependent traveling salesman problem, Transportation Science 29 (2) (1995) 167–183.
[23] B.Y. Wu, Z. Huang, F. Zhan, Exact algorithms for the minimum latency problem, Information Processing Letters 92 (6) (2004) 303–309.
[24] I.R. Webb, Depth-first solutions for the deliveryman problem on tree-like networks: An evaluation using a permutation, in: Schmidt (Ed.), SWAT 2002, in: LNCS, vol. 2368, 2002, pp. 190–199.