CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Traveling Salesman Problem with Pickup and Delivery: Polyhedral Results and a Branch-and-Cut Algorithm

Irina Dumitrescu
Stefan Røpke
Jean-François Cordeau
Gilbert Laporte

January 2008

CIRRELT-2008-01

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Traveling Salesman Problem with Pickup and Delivery: Polyhedral Results and a Branch-and-Cut Algorithm

Irina Dumitrescu[1], Stefan Røpke[2,3], Jean-François Cordeau[2,4,*], Gilbert Laporte[2,3]

[1] School of Mathematics and Statistics, The University of Sydney, Sydney, NSW 2052

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3] Canada Research Chair in Distribution Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[4] Canada Research Chair in Logistics and Transportation, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. The *Traveling Salesman Problem with Pickup and Delivery* (TSPPD) is defined on a graph containing pickup and delivery vertices between which there exists a one-to-one relationship. The problem consists of determining a minimum cost tour such that each pickup vertex is visited before its corresponding delivery vertex. In this paper, the TSPPD is modeled as an integer linear program and its polyhedral structure is analyzed. In particular, the dimension of the TSPPD polytope is determined and several valid inequalities, some of which are facet defining, are introduced. Separation procedures and a branch-and-cut algorithm are developed. Computational results show that the algorithm is capable of solving to optimality instances involving up to 35 pickup and delivery requests, thus more than doubling the previous record of 15.

Keywords. Traveling salesman problem, pickup and delivery, precedence relationships, polyhedral results, valid inequalities, separation procedures, branch-and-cut algorithm.

Acknowledgements. This work was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 227837-04 and 39682-05. This support is gratefully acknowledged.

---

* Corresponding author: Jean-Francois.Cordeau@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2008

1 Introduction

The purpose of this paper is to present polyhedral results and a branch-and-cut algorithm for the *Traveling Salesman Problem with Pickup and Delivery* (TSPPD) defined as follows. Let $G = (V,E)$ be an undirected graph, where V is the set of vertices and E is the set of edges. The set V consists of pickup and delivery vertices, as well as two vertices corresponding respectively to the start and the end depot. Pickup and delivery vertices are paired to form *requests*. Let n be the number of requests, $P = \{1,\ldots,n\}$ the set of pickup vertices, and $D = \{n+1,\ldots,2n\}$ the set of delivery vertices. We denote the delivery vertex corresponding to a pickup vertex $i \in P$ by $n+i$, where $n+i \in D$. We can write the set of vertices as $V = P \cup D \cup \{0,2n+1\}$, where 0 is the vertex corresponding to the start depot, and $2n+1$ is the vertex corresponding to the end depot. For any two vertices i and j, $i < j$, we represent the edge between i and j as (i,j). A non-negative cost c_{ij} is associated with every edge $(i,j) \in E$. The TSPPD consists of finding a least cost Hamiltonian tour on G, containing edge $(0,2n+1)$, and such that each pickup vertex $i \in P$ is visited before the corresponding delivery vertex $n+i$.

The TSPPD has many applications in courier services and dial-a-ride systems. It is also the single-vehicle version of the multi-vehicle one-to-one pickup and delivery problem (VRPPD) on which a rich literature exists (Cordeau et al. [2007]). It can therefore be used to optimize each VRPPD route individually. Several problems are related to the TSPPD. One is the *Traveling Salesman Problem* (TSP) in which each pickup vertex coincides with its delivery vertex. The TSPPD is a special case of the *Precedence-Constrained TSP* (Balas et al. [1995]) in which each vertex may have several predecessors. Another related problem is the *TSP with backhauls* where all pickup vertices must be visited before any of the delivery vertices (see, e.g., Gendreau et al. [1996]). The TSPPD is NP-hard since any TSP instance can be transformed into a TSPPD instance using a polynomial transformation (Renaud et al. [2002]). It is also a very difficult problem from an empirical point of view. The largest instance size solved so far is only $n = 15$. In this paper we more than double this size.

The TSPPD has received relatively little attention. While some papers have proposed exact algorithms for the TSPPD and some of its variants (Hernández-Pérez and Salazar-González [2004], Kalantari et al. [1985]), most have focused on heuristic solution methods (Healy and Moll [1995], Renaud et al. [2002, 2000], Savelsbergh [1990], Fiala Timlin and Pulleyblank [1992]). As far as we are aware only Ruland and Rodin have looked at the polyhedral structure of the TSPPD (Ruland [1994], Ruland and Rodin [1997]). The TSPPD they studied is exactly the one that we consider in our paper. However, apart from establishing the validity of several classes of constraints, Ruland and Rodin did not present polyhedral results. In our paper, we fill some of the gaps in the literature and derive several polyhedral results. In particular, we determine the dimension of the TSPPD polytope, we introduce new valid inequalities, and we show under which conditions several classes of valid inequalities are facets for the TSPPD polytope. We also propose a branch-and-cut algorithm that uses the inequalities discussed.

The remainder of this paper is organized as follows. A mathematical programming formulation of the problem is presented in Section 2. The dimension of the TSPPD polytope is determined in Section 3. Valid inequalities, some of which are facet defining, are introduced in Section 4, and separation procedures are described in Section 5. Implementation details are provided in Section 6 followed by computational results in Section 7 and by conclusions in Section 8.

2 Mathematical model

In addition to the notation already introduced, we define $\delta(S) = \{(i,j) \in E : i \in S, j \notin S \text{ or } i \notin S, j \in S\}$ for any set of vertices $S \subseteq V$. If $S = \{i\}$ we write $\delta(i)$ instead of $\delta(\{i\})$. The TSPPD was formulated by Ruland [1994] as a binary linear program by associating a binary variable x_{ij} with every edge $(i,j) \in E$. We provide this formulation using the notation $x(E')$ for $\sum_{(i,j)\in E'} x_{ij}$, where $E' \subseteq E$:

$$\text{minimize} \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

subject to

$$x_{0,2n+1} = 1 \tag{2}$$
$$x(\delta(i)) = 2 \quad \forall i \in V \tag{3}$$
$$x(\delta(S)) \geq 2 \quad \forall S \subseteq V, 3 \leq |S| \leq |V|/2 \tag{4}$$
$$x(\delta(S)) \geq 4 \quad \forall S \in \mathscr{U} \tag{5}$$
$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E, \tag{6}$$

where $\mathscr{U}$ is the collection of subsets $S \subset V$ satisfying $3 \leq |S| \leq |V|-2$ with $0 \in S$, $2n+1 \notin S$ and for which there exists $i \in P$ such that $i \notin S$ and $n+i \in S$. Constraints (3) are degree constraints, (4) are subtour elimination constraints (SEC), and (5) are precedence constraints which ensure that vertex i is visited before vertex $n+i$ for every $i \in P$.

3 Dimension of the TSPPD polytope

Assuming the set of edges E is ordered, let B^E be the set of binary vectors with components indexed by E. We associate an incidence vector $x \in B^E$ with every tour t in the graph G. The vector x is defined as follows: $x_{ij} = 1$ if $(i,j) \in t$, and $x_{ij} = 0$ otherwise. For notational convenience we do not distinguish between a tour and its incidence vector. We also perform arithmetic on tours, which will translate into basic operations with vectors in B^E. In the rest of this section we will use tour subtraction. The subtraction of a tour t_1 from a tour t_2 will have an incidence vector obtained from subtracting the incidence vector corresponding to t_1 from the incidence vector corresponding to t_2. In fact, the incidence vector of $t_2 - t_1$ represents the way in which the two tours differ from each other; the edges that appear in both tours will cancel out. For example $(1,2,3)-(2,3,5) = ((1,2),(2,3),(1,3))-((2,3),(3,5),(2,5)) = (1,2)+(1,3)-(3,5)-(2,5)$ meaning that the incidence vector corresponding to $(1,2,3)-(2,3,5)$ will have a 1 on the positions corresponding to $(1,2),(1,3)$, a -1 on the positions corresponding to $(3,5)$ and $(2,5)$, and a 0 on the positions corresponding to every other edge. From the incidence vector of the difference we can tell that the first vector in the subtraction contains the edges $(1,2)$ and $(1,3)$, while the second one does not, and that the second tour contains the edges $(1,2)$ and $(1,3)$, while the first tour does not. We call the *leading edge* the edge corresponding to the first non-zero element of an incidence vector or of a vector obtained after performing arithmetic on tours.

Definition 3.1. *Let $\mathscr{T}$ be the set of all feasible tours of the TSPPD, i.e., the incidence vectors that satisfy (2)-(6). The* TSPPD polytope *is*

$$\mathsf{P}_{TSPPD} = conv(\mathscr{T}).$$

Assumption 1. *We make the following assumptions:*

1. $\delta(0) = \{(0,1),(0,2),\ldots,(0,n),(0,2n+1)\}$ *and* $\delta(2n+1) = \{(0,2n+1),(n+1,2n+1),\ldots,(2n,2n+1)\}$.
2. *The subgraph of G induced by G and $P \cup D$ is a complete graph.*

The first assumption simply means that G is the graph obtained at the end of a preprocessing step. The edges that cannot appear in any tour are eliminated before we even attempt to solve the problem. These edges are of the form $(0,n+i)$ or $(i,2n+1)$, where $i \in P$. The graph G cannot be further reduced. The second assumption is clearly non-restrictive and is needed only for the proofs of the theoretical results presented in this paper. We now define an order on the set of edges.

Definition 3.2. *Define $E^0 = \{(0, 2n+1)\}$ and $E^1 = E \setminus (E^0 \cup E^2)$, where $E^2 = (\delta(0) \cup \delta(2n+1) \cup \{(n, 2n)\}) \setminus E^0$. Let $\prec_{E^1}$ be the lexicographic order on the set E^1 and $\prec_{E^2}$ the lexicographic order on the set E^2. We define a relation of total order $\prec$ on the set of edges E as follows:*

i. *for any $(i, j) \in E \setminus E^0$, $(0, 2n+1) \prec (i, j)$;*

ii. *the restriction of $\prec$ to E^1 is $\prec_{E^1}$;*

iii. *the restriction of $\prec$ to E^2 is $\prec_{E^2}$;*

iv. *for any $(i, j) \in E^1$ and $(k,l) \in E^2$, $(i, j) \prec (k,l)$.*

Remark 3.3. *The number of edges in E is $2n^2 + n + 1$.*

Proposition 3.4 (Ruland [1994])**.** *The dimension of P_{TSPPD} is at most $2n^2 - n - 2$.*

Proof. The rank of the matrix induced by the equality constraints is $2n+3$ (see Ruland [1994]), so by Proposition 2.4 from Chapter I.4 of Nemhauser and Wolsey [1988] the polytope has dimension at most $|E| - (2n+3) = 2n^2 - n - 2$. □

In order to prove the next result we need to introduce further notation. Given a set of pickup vertices $S \subseteq P$ we denote by $\mathscr{P}(S)$ the set of all paths that visit all vertices in S exactly once. We denote by τ^p an element of $\mathscr{P}(S)$. We denote by τ^d the path which visits delivery vertices only, such that the k-th element of τ^d is the delivery vertex corresponding to the pickup vertex on position k in τ^p (i.e. $\tau^d_k = n + \tau^p_k$), for any $k = 1, \ldots, |S|$. If a path in a tour is defined on the empty set, we will read the tour *without* that path.

Theorem 3.5. *The dimension of P_{TSPPD} is $2n^2 - n - 2, \forall n \geq 2$.*

Proof. We will prove that there are $(2n^2 - n - 2) + 1$ linearly independent feasible tours in the TSPPD polytope. Since linear independence implies affine independence, there are $2n^2 - n - 1$ affinely independent elements in the TSPPD polytope. This implies that the dimension of the polytope is at least $2n^2 - n - 2$. But from Proposition 3.4 we know that the dimension of the TSPPD polytope is at most $2n^2 - n - 2$. It will follow that the dimension of the TSPPD polytope is exactly $2n^2 - n - 2$.

To construct $2n^2 - n - 1$ linear combinations of feasible tours in the TSPPD polytope, we take each feasible tour and consider it a row in a matrix, in which every column corresponds to an edge (ordered increasingly with respect to the order introduced in Definition 3.2). By row operations we find $2n^2 - n - 1$ linearly independent vectors, which are linear combinations of rows (feasible tours) in the matrix and form an upper triangular matrix. The rank of the upper triangular matrix is $2n^2 - n - 1$ and so the rank of the initial matrix (the one that has *all* the feasible tours as its rows) will be at least $2n^2 - n - 1$. Therefore there are $2n^2 - n - 1$ linearly independent rows of that matrix. Since any row in that matrix is a feasible tour, there are $2n^2 - n - 1$ linearly independent feasible tours of the TSPPD polytope, which is what we need to show.

We will group the linear combinations of feasible tours into several mutually disjoint sets T_i, $i = 0, \ldots, 7$. The set $T = \cup_{i=0}^{7} T_i$ will contain the linearly independent linear combinations of feasible tours needed. Each vector in a set T_i will have a distinct leading edge, from the first $2n^2 - n - 1$ edges (ordered according to the order introduced in Definition 3.2). Next we describe the sets T_i. If $n = 2$, then $T_3 = T_4 = T_6 = T_7 = \emptyset$ and these cases can be skipped.

0. Leading edge $(\mathbf{0}, \mathbf{2n+1})$: Let $T_0 = \{(0, 1, 2, \ldots, 2n, 2n+1)\}$. $|T_0| = 1$.

1. Leading edges $(\mathbf{1}, \mathbf{i})$, $\mathbf{i} = \mathbf{2}, \ldots, \mathbf{n}$:
We construct the vectors a_i as linear combinations of feasible tours, such that their leading edges are $(1, i)$.

- For any $i = 2,\ldots,n-1$, let $\tau^p \in \mathscr{P}(P \setminus \{1,i,n\})$.

$$\begin{aligned} a_i &= (0,1,i,n,n+1,n+i,2n,\tau^p,\tau^d,2n,2n+1) - (0,1,n,i,n+1,n+i,2n,\tau^p,\tau^d,2n,2n+1) \\ &= (1,i) - (1,n) + (n,n+1) - (i,n+1). \end{aligned}$$

- For $i = n$, let $\tau^p \in \mathscr{P}(P \setminus \{1,n\})$.

$$\begin{aligned} a_n &= (0,1,n,n+1,2n,\tau^p,\tau^d,2n+1) - (0,1,n+1,n,2n,\tau^p,\tau^d,2n+1) \\ &= (1,n) - (1,n+1) + (n+1,2n) - (n,2n). \end{aligned}$$

Let $T_1 = \{a_i : i = 2,\ldots,n\}$. Clearly, $|T_1| = n-1$.

2. Leading edges $(\mathbf{1},\mathbf{n}+\mathbf{i})$, $\mathbf{i}=\mathbf{1},\ldots,\mathbf{n}$:
We construct the vectors b_i as linear combinations of feasible tours, such that their leading edges are $(1,n+i)$.

- For $i = 1$ let $\tau^p \in \mathscr{P}(P \setminus \{1,n\})$.

$$\begin{aligned} b_1 &= (0,\tau^p,\tau^d,n,1,n+1,2n,2n+1) - (0,\tau^p,\tau^d,n,1,2n,n+1,2n+1) \\ &= (1,n+1) - (1,2n) + (2n,2n+1) - (n+1,2n+1). \end{aligned}$$

- For any $i \geq 2$, let $\tau^p \in \mathscr{P}(P \setminus \{1,i\})$.

$$\begin{aligned} b_i &= (0,i,\tau^p,\tau^d,n+i,1,n+1,2n+1) - (0,1,n+1,i,\tau^p,\tau^d,n+i,2n+1) \\ &= (1,n+i) - (i,n+1) + (n+1,2n+1) - (n+i,2n+1) + (0,i) - (0,1). \end{aligned}$$

Let $T_2 = \{b_i : i = 1,\ldots,n\}$. $|T_2| = n$.

3. Leading edges $(\mathbf{i},\mathbf{j})$, $\mathbf{i}=\mathbf{2},\ldots,\mathbf{n}-\mathbf{1}$, $\mathbf{j}=\mathbf{i}+\mathbf{1},\ldots,\mathbf{n}$:
We construct the vectors c_{ij} as linear combinations of feasible tours, such that their leading edges are (i,j). Let $\tau^p \in \mathscr{P}(P \setminus \{i,j\})$.

$$\begin{aligned} c_{ij} &= (0,\tau^p,\tau^d,i,j,n+i,n+j,2n+1) - (0,\tau^p,\tau^d,i,n+i,j,n+j,2n+1) \\ &= (i,j) - (i,n+i) - (j,n+j) + (n+i,n+j). \end{aligned}$$

Let $T_3 = \{c_{ij} : i = 2,\ldots,n-1,\ j = i+1,\ldots,n\}$. We note that $|T_3| = (n-2)(n-1)/2$.

4. Leading edges $(\mathbf{i},\mathbf{n}+\mathbf{j})$, $\mathbf{i}=\mathbf{2},\ldots,\mathbf{n}-\mathbf{1}$, $\mathbf{j}=\mathbf{1},\ldots,\mathbf{n}$:
We construct the vectors d_{ij} as linear combinations of feasible tours, such that their leading edges are $(i,n+j)$.

- For any $i \neq j$ and $j \neq n$, let $\tau^p \in \mathscr{P}(P \setminus \{i,j,n\})$.

$$\begin{aligned} d_{ij} &= (0,j,n+j,i,n,2n,n+i,\tau^p,\tau^d,2n+1) - (0,j,n+j,n,i,2n,n+i,\tau^p,\tau^d,2n+1) \\ &= (i,n+j) - (i,2n) - (n,n+j) + (n,2n). \end{aligned}$$

- For any $i = j$, let $\tau^p \in \mathscr{P}(P \setminus \{i,n\})$. We note that since $i = j$ and $i = 2,\ldots,n-1$, we are in the situation where $j \neq n$. In this case the leading edge will be $(i,n+i) = (i,n+j)$.

$$\begin{aligned} d_{ii} &= (0,n,i,n+i,2n,\tau^p,\tau^d,2n+1) - (0,i,n,n+i,2n,\tau^p,\tau^d,2n+1) \\ &= (i,n+i) - (n,n+i) + (0,n) - (0,i). \end{aligned}$$

- For $j = n$, let $\tau^p \in \mathscr{P}(P \setminus \{i,n\})$. The leading edge will be $(i,2n) = (i,n+n)$.

$$\begin{aligned} d_{in} &= (0,n,i,2n,n+i,\tau^p,\tau^d,2n+1) - (0,i,n,2n,n+i,\tau^p,\tau^d,2n+1) \\ &= (i,2n) - (n,2n) + (0,n) - (0,i). \end{aligned}$$

Let $T_4 = \{d_{ij} : i = 2,\dots,n-1,\ j = 1,\dots,n\}$. $|T_4| = n(n-2)$.

5. Leading edges $(\mathbf{n},\mathbf{n}+\mathbf{i})$, $\mathbf{i} = \mathbf{1},\dots,\mathbf{n}-\mathbf{1}$:
We construct the vectors e_i as linear combinations of feasible tours. Let $\tau^p \in \mathscr{P}(P \setminus \{i,n\})$.

$$\begin{aligned} e_i &= (0,\tau^p,\tau^d,i,n,n+i,2n,2n+1) - (0,\tau^p,\tau^d,i,n,2n,n+i,2n+1) \\ &= (n,n+i) - (n,2n) + (2n,2n+1) - (n+i,2n+1). \end{aligned}$$

Since $n+i < 2n, \forall i = 1,\dots,n-1$, the leading edge of any e_i is $(n,n+i)$.
Let $T_5 = \{e_i : i = 1,\dots,n-1\}$. $|T_5| = n-1$.

6. Leading edges $(\mathbf{n}+\mathbf{i},\mathbf{n}+\mathbf{j})$, $\mathbf{i} = \mathbf{1},\dots,\mathbf{n}-\mathbf{2}$, $\mathbf{j} = \mathbf{i}+\mathbf{1},\dots,\mathbf{n}-\mathbf{1}$:
We construct the vectors f_{ij} as linear combinations of feasible tours. Let $\tau^p \in \mathscr{P}(P \setminus \{i,j,n\})$.

$$\begin{aligned} f_{ij} &= (0,\tau^p,\tau^d,i,j,n,n+i,n+j,2n,2n+1) - (0,\tau^p,\tau^d,i,j,n,n+i,2n,n+j,2n+1) \\ &= (n+i,n+j) - (n+i,2n) + (2n,2n+1) - (n+j,2n+1). \end{aligned}$$

Since $i < j$, it follows that $n+i < n+j$. Also, since $i < n$, we have $n+i < 2n$. Therefore, the leading edge of any vector f_{ij} is $(n+i,n+j)$.
Let $T_6 = \{f_{ij} : i = 1,\dots,n-2, j = i+1,\dots,n-1\}$. $|T_6| = (n-2)(n-1)/2$.

7. Leading edges $(\mathbf{n}+\mathbf{i},\mathbf{2n})$, $\mathbf{i} = \mathbf{1},\dots,\mathbf{n}-\mathbf{2}$:
We construct the vectors g_i as linear combinations of feasible tours, such that their leading edges are $(n+i,2n)$. Let $\tau^p \in \mathscr{P}(P \setminus \{i,n-1,n\})$.

$$\begin{aligned} g_i &= (0,\tau^p,\tau^d,n-1,n,i,2n,n+i,2n-1,2n+1) - (0,\tau^p,\tau^d,n-1,n,i,2n,2n-1,n+i,2n+1) \\ &= (n+i,2n) - (2n-1,2n) + (2n-1,2n+1) - (n+i,2n+1). \end{aligned}$$

Let $T_7 = \{g_i : i = 1,\dots,n-2\}$. $|T_7| = n-2$.

The size of the union set $T = \cup_{i=0}^{7} T_i$ is given by $|T| = |T_0| + |T_1| + \cdots + |T_7| = 2n^2 - n - 1$. The vectors in T have distinct leading edges, which are the first $2n^2 - n - 1$ edges with respect to the order introduced (Definition 3.2). Modulo row interchanging, they form an upper triangular matrix of rank $2n^2 - n - 1$. Therefore, they are the vectors we need. $\square$

4 Valid Inequalities

In this section we describe the inequalities we have tested in our branch-and-cut algorithm. We will recall the inequalities proposed by other authors whenever we use them in our algorithm or if we prove new results related to them. We also give conditions under which some of the inequalities define facets of the TSPPD polytope. The proofs of these results are rather technical and tedious. For this reason we chose to provide them in Dumitrescu [2005]. In each of the proofs we show that there are $2n^2-n-2$ linearly independent (therefore affinely independent) feasible tours that satisfy those inequalities at equality. This implies that the face the inequality represents has dimension $2n^2-n-3 = \dim(\mathrm{P}_{TSPPD})-1$ and the face is a facet (see for example Nemhauser and Wolsey [1988]). The proofs are very similar to that of Theorem 3.5. In every one we consider every feasible tour that satisfies the inequalities under consideration at equality to be a row in a matrix. This is always possible when the set of edges E is totally ordered. We demonstrate that this matrix has rank $2n^2-n-2$. This is done by using elementary row operations (addition, subtraction and row interchanging) to obtain an upper triangular matrix. From the definition of the rank of a matrix, if the rank is $2n^2-n-2$, it follows that there are $2n^2-n-2$ linearly independent rows. Because the rows were feasible tours satisfying the inequalities at equality, we have the $2n^2-n-2$ linearly independent elements needed.

We first note that any valid inequality given in this section can be transformed into another valid inequality by switching the positions of the pickup and delivery vertices. This is easy to see, due to symmetry, and is described formally in the following proposition.

Proposition 4.1. *If $ax \leq b$ is a valid inequality for the TSPPD, then there exists another valid inequality $a'x \leq b$, where $a'_{ij} = a_{n+i,n+j}$, $a'_{i,n+j} = a_{j,n+i}$, $a'_{n+i,n+j} = a_{ij}$, $a'_{0i} = a_{n+i,2n+1}$, $a'_{n+i,2n+1} = a_{0i}$ and $a'_{0,2n+1} = a_{0,2n+1}$ for all $i, j \in P$. If $ax \leq b$ is a facet of the TSPPD polytope, then $a'x \leq b$ is also a facet.*

For a set of vertices $S \subseteq V$ we introduce the notation $\pi(S)$ for the set of predecessors of the vertices in S, i.e. $\pi(S) = \{i \in P : n+i \in S\}$. Similarly we denote by $\sigma(S)$ the set of successors of the vertices in S, $\sigma(S) = \{n+i \in D : i \in S\}$. Also, for $S \subseteq V$ we define $\bar{S} = V \setminus S$ and $E(S) = \{(i,j) \in E : i \in S, j \in S\}$. For $S_1, S_2 \subseteq V$ we define $(S_1 : S_2) = \{(i,j) \in E : i \in S_1, j \in S_2 \text{ or } i \in S_2, j \in S_1\}$.

4.1 Generalized Order Constraints

Ruland and Rodin [1997] proved the following result.

Proposition 4.2 (Generalized Order Constraints (GOC))**.** *Let $S_1, \ldots, S_m \subset P \cup D$ be mutually disjoint sets such that $m \geq 2$, $S_i \cap \pi(S_{i+1}) \neq \emptyset, \forall i = 1, \ldots, m$, where $S_{m+1} = S_1$. Then the inequality*

$$\sum_{i=1}^{m} x(S_i) \leq \sum_{i=1}^{m} |S_i| - m - 1 \tag{7}$$

is valid.

Example. Consider the subsets $S_1 = \{i, n+k\}, S_2 = \{j, n+i\}, S_3 = \{k, n+j\}$. Clearly $S_i \cap \pi(S_{i+1}) \neq \emptyset, \forall i = 1,2,3$. The GOC for these sets is $x_{i,n+k} + x_{j,n+i} + x_{k,n+j} \leq 2$. This inequality is illustrated in Figure 1.

An equivalent class of inequalities was also proposed by Balas et al. [1995] for the *precedence-constrained asymmetric traveling salesman problem* (PCATSP), under the name of *cycle breaking inequalities*.

Remark 4.3. *For small values of n the generalized order constraints (7) do not define facets for the TSPPD polytope. This remark is easy to check using Porta (Christof and Löbel).*

4.2 Order Matching Constraints

Proposition 4.4 (Order Matching Constraints (OMC))**.** *For any $i_1, \ldots, i_m \in P$ and $H \subseteq (P \cup D) \setminus \{n+i_1, \ldots, n+i_m\}$ such that $\{i_1, \ldots, i_m\} \subseteq H$, the inequality*

$$x(H) + \sum_{j=1}^{m} x_{i_j, n+i_j} \leq |H| \tag{8}$$

is valid.

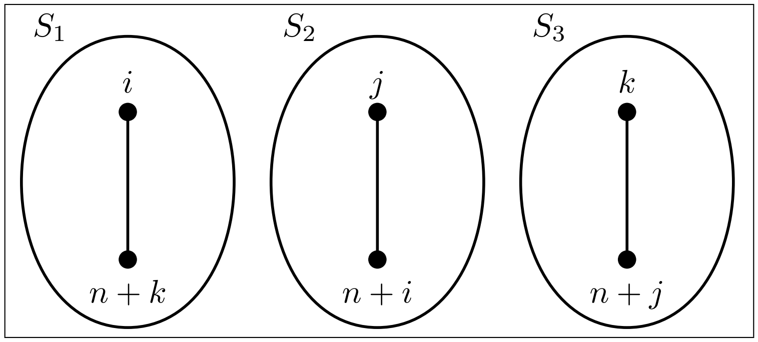

Figure 1: Generalized order constraint with $m = 3$

Proof. The OMC were introduced by Ruland [1994] who stated and proved the above result only for m even. Dumitrescu [2005] has extended this result to m odd by showing that if the proposition is true for $m = 2, \dots, k$ and $2 < k < n$ then it is also true for $m+1$. $\square$

Proposition 4.6 below contains a more general result. In the following proposition we give the characterisation of a subset of the order matching constraints, which are facet defining for the TSPPD polytope.

Proposition 4.5. *For any $H = \{i_1, \dots, i_m\} \subseteq P$, the inequality*

$$x(H) + \sum_{j=1}^{m} x_{i_j, n+i_j} \leq |H|$$

defines a facet of the TSPPD polytope.

Proof. See Dumitrescu [2005]. $\square$

Cordeau [2006] has provided a generalization of the order matching constraints for the asymmetric *dial-a-ride problem*. Because of symmetry this result also holds in our case.

Proposition 4.6 (Generalized order matching constraints (GOMC))**.** *For all $i_1, \dots, i_m \in P$, $H \subseteq P \cup D$ and $T_j \subset P \cup D$, for $j = 1, \dots, m$, such that $\left\{i_j, n+i_j\right\} \subseteq T_j$, $T_i \cap T_j = \emptyset$, $\forall i \neq j$, and $H \cap T_j = \{i_j\}$ for $j = 1, \dots, m$, the inequality*

$$x(H) + \sum_{j=1}^{m} x(T_j) \leq |H| + \sum_{j=1}^{m} |T_j| - 2m$$

is valid for the TSPPD.

This inequality can be further generalized by relaxing the constraints on the sets H and T_j as the following proposition shows.

Proposition 4.7 (Doubly generalized order matching constraints (DGOMC))**.** *For all subsets $\{i_1, \dots, i_m\} \subseteq P$, $H \subset P \cup D$ and $T_j \subset P \cup D$, for $j = 1, \dots, m$, such that $\left\{i_j, n+i_j\right\} \subseteq T_j$ for $j = 1, \dots, m$, $T_i \cap T_j \subseteq H$, $\forall i \neq j$, $\{i_1, \dots, i_m\} \subseteq H$ and $\{n+i_1, \dots, n+i_m\} \cap H = \emptyset$, the inequality*

$$x(H) + \sum_{j=1}^{m} x(T_j) \leq |H| + \sum_{j=1}^{m} |T_j| - 2m \tag{9}$$

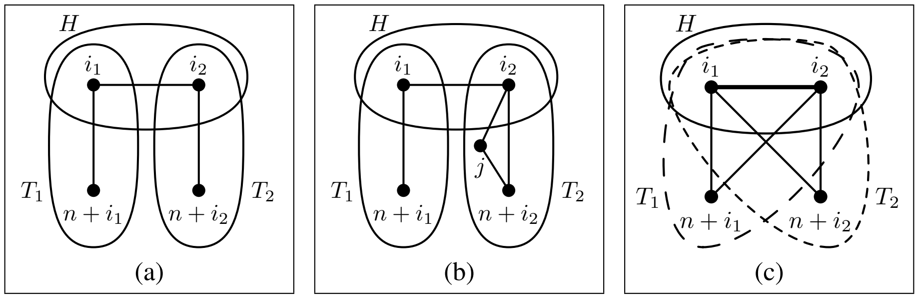

Figure 2: Doubly generalized order matching constraints

is valid for the TSPPD.

Proof. In a feasible TSPPD tour $x(T_j) \leq |T_j| - 1$ for $j = 1, \ldots, m$. If $x(T_j) = |T_j| - 1$ for a subset T_j, then the TSPPD tour enters and leaves T_j once. The tour must visit i_j before $n + i_j$ because of the precedence constraint. Set $\alpha = \left|\left\{ j \in \{1, \ldots, m\} : x(T_j) = |T_j| - 1 \right\}\right|$. Since i_j is in H and $n + i_j$ is outside H for all $j = 1, \ldots, m$ and because $(T_i \cap T_j) \setminus H = \emptyset$ for $i \neq j$ the TSPPD tour must leave H at least α times, and consequently it must enter H at least α times; therefore $x(\delta(H)) \geq 2\alpha$. In a feasible TSPPD solution, $\alpha \geq \sum_{j=1}^{m}(x(T_j) - |T_j| + 2)$ since $x(T_j) - |T_j| + 2 = 1$ if $x(T_j) = |T_j| - 1$ and ≤ 0 otherwise. Using this lower bound on α we obtain $x(\delta(H)) \geq 2\sum_{j=1}^{m}(x(T_j) - |T_j| + 2)$. From the degree constraints, $2x(H) + x(\delta(H)) = 2|H|$ and thus $2|H| - 2x(H) \geq 2\sum_{j=1}^{m}(x(T_j) - |T_j| + 2)$. Rearranging terms yields (9). □

Example. Figure 2 shows three examples of the doubly generalized order matching constraints. The solid lines in each of the figures represent the edges on the left-hand side of (9). The first figure (2.a) shows a simple DGOMC with four vertices, at most two edges can be used. This DGOMC is also an OMC. The second figure (2.b) shows a DGOMC with five vertices. This DGOMC is also a GOMC, but not an OMC as T_2 contains three vertices. At most three of the edges in the figure can be used in a TSPPD solution. The last figure (2.c) shows a DGOMC with $H = \{i_1, i_2\}$, $T_1 = \{i_1, i_2, n+i_1\}$ and $T_2 = \{i_1, i_2, n+i_2\}$. This DGOMC is neither an OMC nor a GOMC as $T_1 \cap T_2 \neq \emptyset$. The edge $\{i_1, i_2\}$ has a coefficient 3 on the left-hand side as it appears in all the terms $x(H)$, $x(T_1)$ and $x(T_2)$. This is illustrated by a thick line in the figure. The left-hand side can be at most 4 in this example.

4.3 Precedence Constraints

The set of inequalities that we discuss next were proved to be valid by Ruland [1994]. They appear as inequalities (5) in the mathematical model provided in Section 2.

Proposition 4.8 (Precedence Constraints (PC))**.** *For any $S \subset V$, $3 \leq |S| \leq |V| - 2$ with $0 \in S, 2n+1 \notin S$, and for which there exists $i \in P$ such that $i \notin S$ and $n + i \in S$, the inequality*

$$x(\delta(S)) \geq 4 \tag{10}$$

is valid for the TSPPD.

The following proposition establishes that a subset of the precedence constraints are facet defining.

Proposition 4.9. *Under the assumption that there exists a unique $i \in P$ such that $i \notin S$ and $n + i \in S$, the inequality (10) is facet defining for the TSPPD polytope.*

Proof. See Dumitrescu [2005]. □

Ruland [1994] showed that the subset of the precedence constraints identified in Proposition 4.9 are sufficient to guarantee a feasible integer solution together with constraints (2)–(4) and (6).

4.4 π-inequalities

We now describe a set of valid inequalities for the TSPPD polytope, which were first introduced by Balas et al. [1995] for the PCATSP.

Proposition 4.10 (π-Inequalities)**.** *For any $S \subseteq V \setminus \{2n+1\}$, the inequality*

$$x((S \setminus \pi(S) : \bar{S} \setminus \pi(S))) \geq 1 \tag{11}$$

is valid for the TSPPD polytope.

Balas et al. [1995] used the term σ*-inequalities* to denote the inequalities obtained by exchanging the roles of pickup and delivery vertices in the π-inequalities.

Remark 4.11. *For small values of n the inequalities (11) do not define facets for the TSPPD polytope.*

This remark is easy to check with Porta (Christof and Löbel). As the depot is split into a start- and end-depot in our formulation it is possible to strengthen the inequality slightly:

Proposition 4.12. *For any $S \subseteq V \setminus \{2n+1\}$, the inequality*

$$x\left(\left(S \setminus \pi(S) : \bar{S} \setminus (\pi(S) \cup \{0\})\right)\right) \geq 1 \tag{12}$$

is valid for the TSPPD polytope.

Proof. Consider a feasible tour for the TSPPD. If we assume that 0 is the first vertex in the tour, let $\hat{s}$ be the last vertex from S on the tour. We note that $\hat{s} \in S \setminus \pi(S)$. The successor of $\hat{s}$ in the tour cannot be vertex 0, it is not in S and cannot be in $\pi(S)$, therefore it must belong to $\bar{S} \setminus (\pi(S) \cup \{0\})$. It follows that the edge between $\hat{s}$ and its successor in the tour links $S \setminus \pi(S)$ and $\bar{S} \setminus (\pi(S) \cup \{0\})$. □

Example. Figure 3 shows an example of a π-inequality for an instance with three requests. The figure shows the set S containing the vertices $\{i, n+i, n+j\}$. The solid lines shown in the figure represent the edges that appear on the left-hand side of (12). At least one of these edges must be used in any feasible tour.

4.5 Lifted Subtour Elimination Constraints

This section presents new valid inequalities that strengthen the classical TSP subtour elimination constraints.

Proposition 4.13 (Lifted subtour elimination constraints (LSEC))**.** *Let $S \subseteq P \cup D$ with the property that there exists $i \in P$ such that $i \in S$ and $n+i \in S$. The inequality*

$$x(S) + \sum_{j \in S \cap P, n+j \notin S} x_{i,n+j} \leq |S| - 1 \tag{13}$$

is valid.

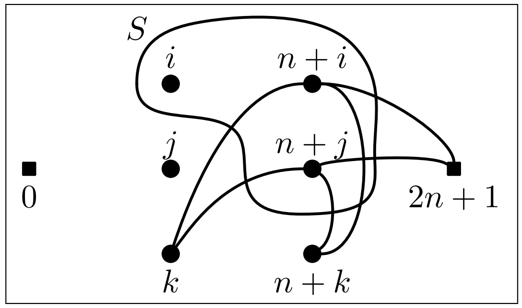

Figure 3: π-inequality example

We skip the proof of Proposition 13 since we will soon introduce a more general version of these inequalities.

Proposition 4.14. *Equation (13) is facet defining under the following assumptions:*

- *there is no $i \in P$ such that $i \notin S$ and $n+i \in S$, and*
- $\{i : i \in P \cap S, n+i \notin S\} \neq \emptyset$.

Proof. See Dumitrescu [2005]. □

We note that the lifted subtour elimination constraint can be generalized as follows.

Proposition 4.15 (Generalized lifted subtour elimination constraints (GLSEC))**.** *Let $S \subset P \cup D$ with the property that there exists $i \in P$ such that $i \in S$ and $n+i \in S$. Let $T_k \subset P \cup D$, for $k = 1, \ldots, m$ such that there exists a $p_k \in P$ such that $p_k \in S$ and $n + p_k \in T_k$ for $k = 1, \ldots, m$. Furthermore we require that $T_k \cap S = \{i\}, \forall k = 1, \ldots, m$ and $T_j \cap T_k = \{i\}, \forall j = 1, \ldots, m, k = 1, \ldots, m, j \neq k$. The inequality*

$$x(S) + \sum_{k=1}^{m} x(T_k) \leq |S| - 1 + \sum_{k=1}^{m} (|T_k| - 2) \tag{14}$$

is valid.

Proof. Assume that the inequality is violated in a valid TSPPD solution. This implies that there exists a vertex i and sets S, T_k satisfying the conditions in the proposition such that

$$x(S) + \sum_{k=1}^{m} x(T_k) \geq |S| + \sum_{k=1}^{m} (|T_k| - 2).$$

First note that in a valid solution at most two of the sets T_k can satisfy the equality $x(T_k) = |T_k| - 1$ because of the degree constraint (3) on vertex i.

Assume $x(T_{k_\alpha}) = |T_{k_\alpha}| - 1$ and $x(T_{k_\beta}) = |T_{k_\beta}| - 1$ for $k_\alpha, k_\beta \in \{1, \ldots, m\}$, $k_\alpha \neq k_\beta$ then we have that $x(T_k) \leq |T_k| - 2$ for all $k \in \{1, \ldots, m\}$, $k \neq k_\alpha$, $k \neq k_\beta$. In order for the inequality to be violated we must have that $x(S) \geq |S| - 2$. But $x(S) = |S| - 1$ is not possible as $x(\delta(i) \cap E(S)) = 0$ because of the degree constraint on vertex i. Consequently $x(S) = |S| - 2$ if a feasible integer solution violates the inequality under the given assumptions. In that case $S' = S \setminus \{i\}$ and T_{k_α} defines a violated generalized order constraint (see section 4.1) as $\{i, n + p_{k_\alpha}\} \subseteq T_{k_\alpha}$, $\{p_{k_\alpha}, n+i\} \subseteq S'$, and $x(S') + x(T_{k_\alpha}) = |S'| + |T_{k_\alpha}| - 2$.

Now assume that $x(T_{k_\alpha}) = |T_{k_\alpha}| - 1$, $k_\alpha \in \{1,\ldots,m\}$ and that $x(T_k) \leq |T_k| - 2$ for all $k \in \{1,\ldots,m\}$, $k \neq k_\alpha$. In that case $x(S) = |S| - 1$ if the GLSEC is violated. This again implies that the sets $S' = S \setminus \{i\}$ and T_{k_α} define a violated generalized order constraint. If $x(T_k) \leq |T_k| - 2$ for all $k \in \{1,\ldots,m\}$, then the inequality cannot be violated as $x(S) \leq |S| - 1$ (due to the subtour elimination constraint (4)). We can conclude that the GLSEC cannot be violated by a feasible integer solution to the TSPPD. □

The inequality is a generalization of the LSEC (13) as a given LSEC defined by a pickup vertex i and a set S can be expressed as a GLSEC as follows. Let $\{p_1,\ldots,p_m\} = \{j \in P \cap S : n+j \notin S\}$ be the set of pickups in S without their corresponding delivery in S. The pickup vertex i and the sets S and $T_k = \{i, n+p_k\}, k = 1,\ldots,m$ define a GLSEC that is identical to the LSEC.

Example. Figure 4 shows two examples of the GLSEC. The first example (a) shows a GLSEC with $S = \{i, n+i, p_1, p_2\}, T_1 = \{i, n+p_1\}, T_2 = \{i, n+p_2\}$ where $\{p_1, p_2\} \subseteq P$. The solid lines in the figure represent the edges on the left-hand side of inequality (14). At most three of the edges can be used in a feasible tour. This GLSEC is also an LSEC.

The second example (b) shows a GLSEC with $T_2 = \{i, n+p_2, k\}$ and S and T_1 as before. This GLSEC is not an LSEC as $|T_2| > 2$. At most four of the edges in the figure can be used in a feasible tour.

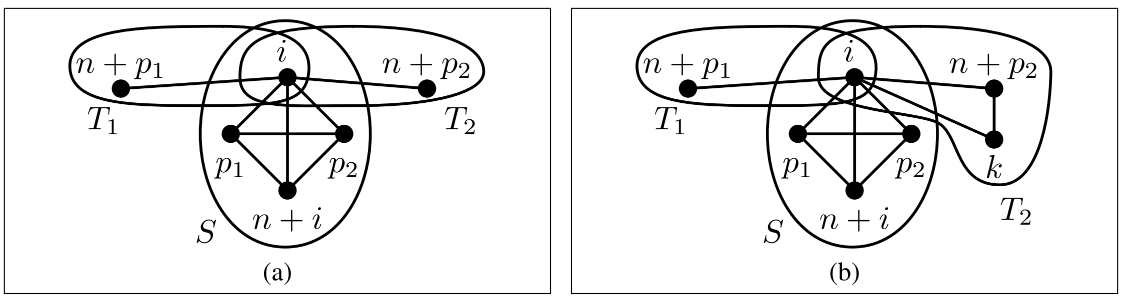

Figure 4: Generalized lifted subtour elimination constraint

4.6 Depot Constraints

Proposition 4.16 (Depot constraints (DC))**.** *For any $S \subset V$ and $T \subset D$ such that $0 \in S$, $2n+1 \notin S$, $S \cap T = \emptyset$ and $\pi(T) \cap S = \emptyset$ the following inequality*

$$2x(S) + x(S:T) \leq 2(|S| - 1) \tag{15}$$

is valid for the TSPPD.

Proof. Let $S' = S \setminus \{0\}$. In an integer solution, $x(0 : S')$ is equal to 0 or 1. If $x(0 : S') = 1$, the number of edges between S and T is at most $x(\delta(S)) - 2$. This is true since one of the edges from the set $\delta(S)$ is used to connect 0 to $2n+1 \notin T$ and at least one other edge from the set cannot reach T as this would imply that there is a path from vertex 0 to a delivery vertex that does not visit the corresponding pickup vertex (because of the definition of S and T). Thus

$$\begin{aligned} x(S:T) &\leq (x(\delta(S)) - 2) \\ \Leftrightarrow x(S:T) &\leq 2(|S| - x(S)) - 2 \\ \Leftrightarrow 2x(S) + x(S:T) &\leq 2(|S| - 1) \end{aligned}$$

as $x(\delta(S)) = 2(|S| - x(S))$. Consequently the inequality is valid when $x(0:S') = 1$. If $x(0:S') = 0$ then we have

$$\begin{aligned} x(S:T) &\leq x\left(\delta\left(S'\right)\right) \\ \Leftrightarrow x(S:T) &\leq 2(|S'| - x(S')) \\ \Leftrightarrow 2x(S') + x(S:T) &\leq 2|S'| \\ \Leftrightarrow 2x(S) + x(S:T) &\leq 2(|S| - 1). \end{aligned}$$

The first inequality is true since $x(0:T) = 0$; the last equivalence is true since $x(S) = x(S')$ because of the assumption $x(0:S') = 0$. $\square$

It is easy to see that the depot inequality is a strengthening of the subtour elimination constraints for the set S when $0 \in S$.

Example. Figure 5 shows an example of a depot constraint. In the figure $S = \{0, i, j\}$ and $T = \{n+k, n+l\}$ where $\{i, j, k, l\} \subseteq P$ and $\{i, j\} \cap \{k, l\} = \emptyset$. The solid lines correspond to the the left-hand side of inequality (15), and the thick lines represent edges with coefficient 2. For this depot constraint the left-hand side has to be less than or equal to 4.

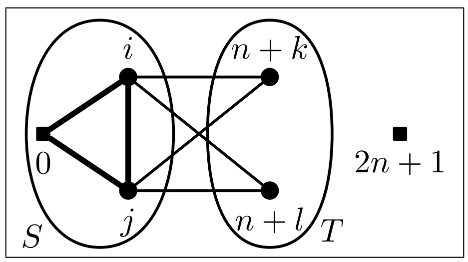

Figure 5: Depot constraint

4.7 Start-End Constraints

This section introduces the *start-end constraint* (StEnC). Before presenting the valid inequality a technical definition is necessary. The definition characterizes the so called *SE-infeasible* sets that lead to start-end constraints.

Definition 4.17. *Let $S \subseteq P \cup D$, $S_1 \subseteq S \cap P$, $S_2 \subseteq S \cap D$ and $E_S \subseteq E(S)$. The quadruple (S, S_1, S_2, E_S) is called* SE-feasible *if a feasible TSPPD tour $(0, v_1, \ldots, v_{2n}, 2n+1)$ exists such that either*

1. *$\{v_1, \ldots, v_{|S|}\} = S$ and $(v_i, v_{i+1}) \in E_S$ for all $i \in \{1, \ldots, |S|-1\}$ and $v_1 \in S_1$ or*
2. *$\{v_{2n-|S|+1}, \ldots, v_{2n}\} = S$ and $(v_i, v_{i+1}) \in E_S$ for all $i \in \{2n-|S|+1, \ldots, 2n-1\}$ and $v_{2n} \in S_2$ or*
3. *there exist integers $p \geq 1$ and $q \geq 1$, $p+q = |S|$ such that $S = \{v_1, \ldots, v_p\} \cup \{v_{2n-q+1}, \ldots, v_{2n}\}$ and $(v_i, v_{i+1}) \in E_S$ for all $i \in \{1, \ldots, p-1\} \cup \{2n-q+1, \ldots, 2n-1\}$ and $v_1 \in S_1$ and $v_{2n} \in S_2$.*

If the quadruple is not SE-feasible it is called SE-infeasible.

Example. Figure 6 shows two examples of SE-feasible quadruples and two examples of SE-infeasible quadruples. In these examples i, j, k and l are assumed to be distinct pickup vertices. Figure (6.a) shows the quadruple $(S = \{i, j, k, n+i\}, S_1 = \{i, j\}, S_2 = \emptyset, E_S = \{(i, j), (i, n+i), (j, n+i), (j, k)\})$, and the solid lines in the figure represent the set E_S. This quadruple is SE-feasible as it fulfills condition 1 in Definition 4.17 with a feasible TSPPD tour being $(0, i, n+i, j, k, \ldots, 2n+1)$.

Figure (6.b) shows the quadruple $(S = \{i, j, n+k, n+l\}, S_1 = \{i, j\}, S_2 = \{n+l\}, E_S = \{(i, j), (n+k, n+l)\})$. This quadruple is SE-feasible as it fulfills condition 3 in Definition 4.17 with a feasible TSPPD tour being $(0, i, j, \ldots, n+k, n+l, 2n+1)$.

Figure (6.c) shows the quadruple $(S = \{i, j, n+i, n+k\}, S_1 = \{i, j\}, S_2 = \{n+i, n+k\}, E_S = \{(i, n+k), (j, n+k), (j, n+i)\})$ this quadruple is SE-infeasible as it is impossible to find a feasible TSPPD tour that satisfies one of the three conditions in Definition 4.17. If (i, j) is added to E_S then the quadruple becomes SE-feasible as $(0, i, j, n+i, \ldots, n+k, 2n+1)$ is a feasible tour satisfying condition 3 with the modified set E_S.

Figure (6.d) shows another SE-infeasible quadruple $(S = \{i, j, k, n+k\}, S_1 = \{i, j\}, S_2 = \{n+k\}, E_S = \{(i, k), (i, n+k), (j, n+k)\})$. If k is added to S_1 then the quadruple becomes SE-feasible as $(0, k, i, n+k, j, \ldots, 2n+1)$ is a feasible tour satisfying condition 1 with the modified set S_1.

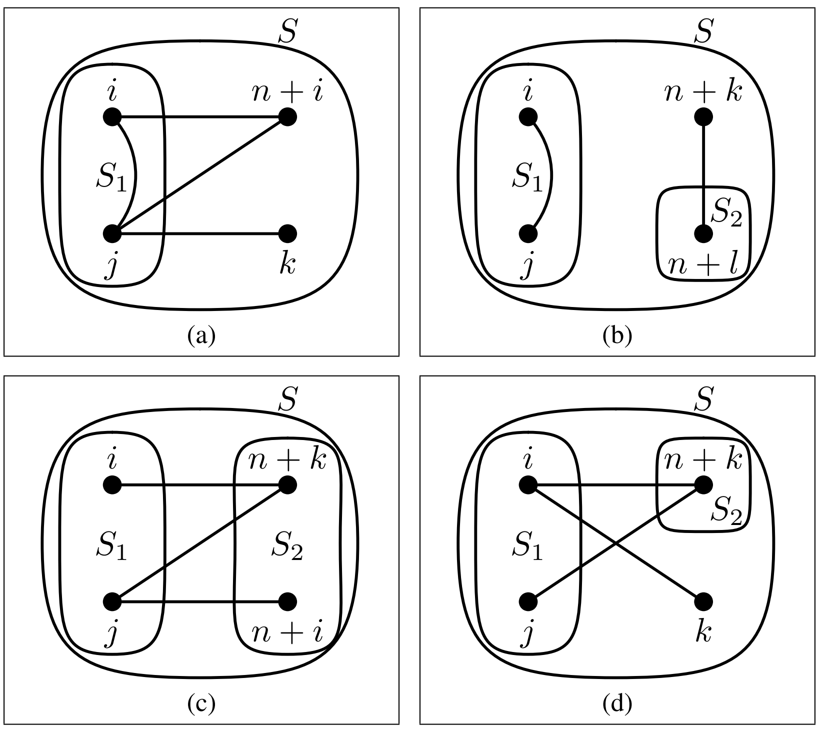

Figure 6: Examples of SE-feasibility (cases (a) and (b)) and SE-infeasibility (cases (c) and (d)).

Proposition 4.18 (Start-End Constraints (StEnC)). *Let $S \subseteq P \cup D$, $S_1 \subseteq S \cap P$, $S_2 \subseteq S \cap D$ and $E_S \subseteq E(S)$. If (S, S_1, S_2, E_S) is SE-infeasible then the following inequality is valid*

$$x(E_S) + x(0 : S_1) + x(2n+1 : S_2) \leq |S| - 1. \tag{16}$$

Proof. Let $E_0 = (0 : S_1) \cup (2n+1 : S_2)$. The proof of Proposition 4.18 is split into 3 parts: $x(E_0) = 0$, $x(E_0) = 1$, and $x(E_0) = 2$. For $x(E_0) = 0$ the inequality is implied by the subtour elimination constraint on S. If $x(E_0) = 1$ then the inequality can only be violated by a feasible integer solution if $x(E_S) = |S| - 1$, which implies that the tour only enters and leaves S once. The case $x(E_0) = 1$ implies that the path visiting S either connects to the start depot using an edge from $(0 : S_1)$ or to the end depot using an edge from $(2n+1 : S_2)$. Both cases are ruled out by the condition that (S, S_1, S_2, E_S) is SE-infeasible: $x(0 : S_1) = 1$ implies that condition 1) in Definition 4.17 is satisfied and $x(2n+1 : S_2) = 1$ implies that condition 2) in Definition 4.17 is satisfied. Last, if $x(E_0) = 2$ then the inequality can only be violated by a feasible integer solution if $x(E_S) \geq |S| - 2$. $x(E_S) = |S| - 1$ is ruled out by using the above arguments. Indeed $x(E_S) = |S| - 2$ and $x(E_0) = 2$ imply that the tour starts in vertex 0, visits a number of vertices in S, then leaves S to visit vertices outside of S, returns to visit vertices in S and finally goes directly from S to $2n+1$. The vertex visited after vertex 0 must be from S_1 and the vertex visited before vertex $2n+1$ must be a vertex from S_2. This is again impossible because (S, S_1, S_2, E_S) is SE-infeasible. Such a solution would imply that condition 3) in Definition 4.17 is satisfied. □

Example. Figure 7 illustrates the start-end constraints obtained from the SE-infeasible sets shown in Figures (6.c) and (6.d). At most three of the edges shown in Figure (7.a) and (7.b) can be used in a feasible TSPPD tour.

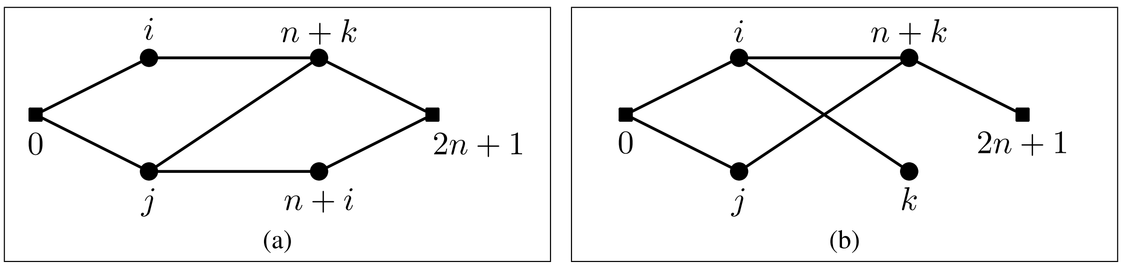

Figure 7: Start-end constraints obtained from the SE-infeasible quadruples shown in Figures (6.c) and (6.d).

It is worth noting that the proposition and its proof only use the notion of feasible tours. This implies that the inequality can be used for other variants of the TSP, for example the *Traveling Salesman Problem with Time Windows*.

5 Separation Procedures

This section describes both exact and heuristic separation procedures for the valid inequalities introduced in Section 4. To separate the subtour elimination constraints (4) and precedence constraints (5) we use the separation procedures suggested by Ruland [1994]. Note that many of the inequalities induce a similar inequality by switching the roles of the pickup and delivery vertices, and the vertices of the start and end depot as stated in Proposition 4.1. In this section we only describe algorithms for separating one of the two forms. Separation algorithms for the sister-inequalities follow easily.

In the rest of this section we use x^* to denote the fractional solution we wish to separate from P_{TSPPD}.

5.1 Separation of generalized order constraints

Ruland [1994] proposed an exact separation procedure for separating the GOC with $m = 2$. We have implemented a similar procedure, shown in Algorithm 1. We assume that x^* does not violate any subtour elimination constraint. The algorithm works by iterating through all pairs of requests. For each request pair $(i, n+i)$ and $(j, n+j)$ the best possible sets S_1 and S_2 are constructed such that $S_1 \cap S_2 = \emptyset$ and $i \in S_1 \cap \pi(S_2)$ and $j \in S_2 \cap \pi(S_1)$. The optimization problems in lines 3 and 4 can be solved using a maximum flow algorithm. The key observation needed to see that this algorithm finds a violated GOCs for $m = 2$ if one exists is that $x^*(S'_2) - |S'_2| = x^*(S_2) - |S_2|$ in line 5. We now prove the validity of this claim. Let $T = S_1 \cap S_2$. Assume that $x^*(T) + x^*(T : S_1 \setminus T) < |T|$. Combining this assumption with the fact that $x^*(S_1) = x^*(S_1 \setminus T) + x^*(T) + x^*(T : S_1 \setminus T)$ leads to the inequality $x^*(S_1 \setminus T) - |S_1 \setminus T| > x^*(S_1) - |S_1|$ which is in contradiction with the maximization performed in line 3 of the algorithm. Therefore $x^*(T) + x^*(T : S_1 \setminus T) \geq |T|$ and analogously $x^*(T) + x^*(T : S_2 \setminus T) \geq |T|$. Adding these two inequalities yields $2x^*(T) + x^*(T : S_1 \setminus T) + x^*(T : S_2 \setminus T) \geq 2|T|$. Since $2x^*(T) + x^*(\delta(T)) = 2|T|$ and $(S_1 \setminus T) \cap (S_2 \setminus T) = \emptyset$ we also have that $2x^*(T) + x^*(T : S_1 \setminus T) + x^*(T : S_2 \setminus T) \leq 2|T|$ so $x^*(T) + x^*(T : S_1 \setminus T) = x^*(T) + x^*(T : S_2 \setminus T) = |T|$. Now $x^*(S_2) - |S_2| = x^*(S_2 \setminus T) + x^*(T) + x^*(T : S_2 \setminus T) - |S_2| = x^*(S_2 \setminus T) + |T| - |S_2| = x^*(S'_2) - |S'_2|$ as claimed.

Algorithm 1 can be extended to find inequalities with $m > 2$ for a constant m. Its time complexity is $O(n^m f(n))$, where $f(n)$ is the time complexity of solving a maximum flow problem on a graph with n vertices. For $m > 2$ this effort does not seem to be worthwhile compared with the effect the inequalities have on the lower bound. Instead we propose a heuristic for separating generalized order constraints for $m \geq 3$.

Algorithm 1 Exact separation of generalized order constraints (for $m = 2$)

1: **for all** $i \in \{1, \ldots, n\}$ **do**
2:     **for all** $j \in \{i+1, \ldots, n\}$ **do**
3:         $S_1 = \arg\max_{S \subset P \cup D}\{x^*(S) - |S| : \{i, n+j\} \subseteq S, \{j, n+i\} \cap S = \emptyset\}$.
4:         $S_2 = \arg\max_{S \subset P \cup D}\{x^*(S) - |S| : \{j, n+i\} \subseteq S, \{i, n+j\} \cap S = \emptyset\}$.
5:         Set $S'_2 = S_2 \setminus (S_1 \cap S_2)$.
6:         **if** $x^*(S_1) + x^*(S'_2) > |S_1| + |S'_2| - 3$ **then**
7:             A violated inequality has been found. Store the inequality.
8:         **end if**
9:     **end for**
10: **end for**

The heuristic needs a set $\mathscr{S}$ of candidates for the sets S_i in the inequality. For this we use all the sets calculated in lines 3 and 5 of Algorithm 1. We only keep the sets S for which $x(S) > |S| - 2$ as sets S with $x(S) \leq |S| - 2$ can never play the role of one of the sets S_i in a violated GOC. A multi-graph G' with n vertices is created. For each set S in $\mathscr{S}$ we generate one or more arcs in the graph. Each arc (i, j), $i, j \in P$ corresponds to a set $S \in \mathscr{S}$ for which $\{j, n+i\} \subseteq S$ and $\{i, n+j\} \cap S = \emptyset$. We associate a cost as well as the vertices of S with the arc. The cost of the arc is set to $|S| - x^*(S) - 1$, so that the cost of all arcs are greater than or equal to zero. To find violated GOCs we look for minimum cost cycles in G', such that the vertex sets associated with the arcs in the cycle are mutually disjoint. Only cycles with cost less than 1 are interesting as these correspond to violated inequalities. Such cycles can be found by means of a labeling algorithm for the resource constrained shortest path problem, as described by Irnich and Desaulniers [2005]. This leads to an algorithm whose running time is not polynomially bounded, but is reasonably fast in practice. To ensure that the branch-and-cut algorithm does not spend too much time separating GOCs with $m \geq 3$ we impose a time limit on the running time of the algorithm for finding cycles in G'.

5.2 Heuristic separation of doubly generalized order matching constraints

In this section we present a heuristic for the DGOMC. Since this family of inequalities generalizes the OMC and GOMC we do not implement separation procedures for them. It is worth noting that Ruland [1994] proposed an exact separation procedure for the OMC for the special case when $m = 2$.

The separation heuristic for the DGOMC is shown in pseudo-code in Algorithm 2. The heuristic considers all pairs of requests. For each pair of requests $(i, n+i)$ and $(j, n+j)$ it attempts to find a violated inequality with $m \geq 2$, where $(i, n+i) \subseteq T_1$ and $(j, n+j) \subseteq T_2$. The candidates for T_1 and T_2 are found in lines 3 and 4. The conditions on the optimization problem in lines 3 and 4 ensure that the conditions in Proposition 4.7 are satisfied. In lines 5 and 6 a set H that matches the sets T_1 and T_2 is constructed. The intermediate set H' contains the vertices that H must contain, while the statement in line 6 constructs H such that $x^*(H) - |H|$ is maximized subject to H satisfying the conditions in Proposition 4.7. The sets T_1, T_2 and H found in lines 3 to 6 could define a violated DGOMC, but before checking whether a violated inequality has been found, the heuristic tries to add more T sets to the inequality (in line 8).

Algorithm 2 Doubly generalized order matching separation

1: **for all** $i \in \{1, \ldots, n\}$ **do**
2:   **for all** $j \in \{i+1, \ldots, n\}$ **do**
3:     $T_1 = \arg\max_{S \subseteq P \cup D}\{x^*(S) - |S| : \{i, n+i\} \subseteq S, n+j \notin S\}$.
4:     $T_2 = \arg\max_{S \subseteq P \cup D}\{x^*(S) - |S| : \{j, n+j\} \subseteq S, n+i \notin S\}$.
5:     $H' = \{i, j\} \cup (T_1 \cap T_2)$.
6:     $H = \arg\max_{S \subseteq P \cup D}\{x^*(S) - |S| : H' \subseteq S, \{n+i, n+j\} \cap S = \emptyset\}$.
7:     set $\Gamma = \{T_1, T_2\}$.
8:     Add more T sets to Γ to improve inequality.
9:     **if** $x^*(H) + \sum_{T \in \Gamma} x^*(T) > |H| + \sum_{T \in \Gamma} |T| - 2|\Gamma|$ **then**
10:       A violated inequality has been found. Store the inequality.
11:     **end if**
12:   **end for**
13: **end for**

We have implemented two strategies for adding T sets. The first strategy keeps the set H fixed and adds T sets that are compatible with H and the existing T sets. That is, the new set T' should contain a request $(k, n+k)$ such that $k \in H$, $n+k \notin H$ and $T' \cap T \subseteq H$ for all $T \in \Gamma$, where Γ is the set of all existing T sets (introduced in line 7). The second strategy keeps the existing T sets fixed, but allows H to change. This strategy generates new T sets that are compatible with the existing T sets. When a promising new T set has been identified a new set H is computed that ensures that the conditions from Proposition 4.7 are satisfied. This set is constructed in a way that resembles how the original H set was constructed in lines 5 and 6. For both strategies we only accept changes to an existing collection of T and H sets if they improve the inequality defined by the original sets (makes the inequality more violated or makes it closer to being violated). The second strategy is more time consuming than the first one, so it is only attempted at the root node of the branch-and-cut tree. In lines 9 to 11 the heuristic checks for violation of a DGOMC. If a violated inequality has been detected, it is stored and the heuristic moves on to a new pair of requests.

The computations in lines 3, 4 and 6 can be performed by using a max-flow algorithm. In order to speed up the heuristic we only use an exact max-flow algorithm at the root node of the branch-and-cut tree and a faster heuristic max-flow algorithm elsewhere. Another way of speeding up the heuristic is to skip lines 3 to 11 if either T_1 or T_2 found in lines 3 and 4 are poor, that is, if $x^*(T_1) - |T_1|$ or $x^*(T_2) - |T_2|$ are close to 2.

5.3 Heuristic separation of π- and σ- inequalities

The heuristic for separating π- and σ-inequalities uses a randomized greedy principle. It creates an initial candidate set S containing only one vertex. Vertices are added to the set iteratively. The set S is used as a candidate for both π-

and σ-inequalities. At each iteration the heuristic selects the vertex to add using the formula

$$\arg \min_{i \in (P \cup D) \setminus S} \left\{ \min \left\{ x^*(S \setminus \pi(S) : \bar{S} \setminus (\pi(S) \cup \{0\})), x^*(S \setminus \sigma(S) : \bar{S} \setminus (\sigma(S) \cup \{2n+1\})) \right\} \right\}.$$

A noise is added to the evaluation of $x^*(\cdot)$ in order to randomize the heuristic. If a set S violates either a π- or a σ-inequality this inequality is stored and the heuristic considers a new vertex to initialize S. All vertices are used as initial vertices several times.

5.4 Exact separation of lifted subtour elimination constraints

This section describes an exact separation algorithm for the LSEC. The running time of the procedure is exponential in n, but for the instance sizes considered in this paper it is reasonably fast. The pseudo-code is shown in Algorithm 3. We assume that the fractional solution does not violated any ordinary subtour elimination constraint. In line 1, the algorithm selects the *connector vertex*, i.e, the vertex corresponding to vertex i in Proposition 4.13. We call it the connector vertex because it provides the connection between the set S and the delivery vertices outside S. In line 2 the set of interesting delivery vertices for the chosen connector vertex is determined, and in line 3 the algorithm iterates through all subsets of the interesting delivery vertices. For each such subset the algorithm computes in line 4 the optimal corresponding set S that together with the connector vertex defines an LSEC. In line 5 a check is performed to determine if the constructed LSEC is violated.

Algorithm 3 Exact separation of lifted subtour elimination constraints

1: **for all** $i \in \{1, \ldots, n\}$ **do**
2:     $D_i = \{j \in D \setminus \{n+i\} : x^*_{ij} > 0\}$.
3:     **for all** $W \subseteq D_i$ **do**
4:         $S_W = \arg\max_{S \subseteq P \cup D} \{x^*(S) - |S| : (\{i, n+i\} \cup \pi(W)) \subseteq S, W \cap S = \emptyset\}$.
5:         **if** $x^*(S_W) + x^*(i : W) > |S_W| - 1$ **then**
6:             A violated LSEC has been found, store the inequality.
7:         **end if**
8:     **end for**
9: **end for**

The computation in line 4 can be carried out using a max-flow computation. The algorithm has a worst case exponential running time as every subset of the set D_i is examined in lines 3 to 8. The set D_i can contain up to n elements, but in practice D_i is rather small, usually $|D_i| \leq 6$.

Proposition 5.1. *Algorithm 3 finds a violated LSEC if one exists.*

Proof. Assume that $S' \subseteq P \cup D$ and the connector vertex $i' \in S' \cap P$ defines a violated LSEC. We show that the algorithm detects a violated LSEC. Since we assumed that S' and i' yielded a violated LSEC and that no SEC was violated by the fractional solution, we have

$$\sum_{j \in (S' \cap P), n+j \notin S'} x^*_{i',n+j} > 0.$$

Let $W' = \{j \in \sigma(S') : j \notin S', x^*_{i'j} > 0\}$. Then $\sum_{j \in (S' \cap P), n+j \notin S'} x^*_{i',n+j} = x^*(i : W')$. At some point during execution of the separation algorithm $i = i'$ and $W = W'$ in line 4, because of the definition of the set D_i. When this occurs the algorithm finds the set S_W maximizing $x^*(S_W) - |S_W|$ such that $(\{i, n+i\} \cup \pi(W)) \subseteq S_W$ and $W \cap S_W = \emptyset$. The set S' satisfies these constraints as well as it defined an LSEC. This implies $x^*(S_W) - |S_W| \geq x^*(S') - |S'|$ and therefore

$$\begin{aligned} x^*(S_W) - |S_W| + x^*(i' : W) &\geq x^*(S') - |S'| + x^*(i' : W') > -1 \\ \Rightarrow x^*(S_W) + x^*(i' : W) &> |S_W| - 1. \end{aligned}$$

The second inequality follows as S' and i defined a violated LSEC. This shows that a violated inequality will be detected in line 5 whenever $i = i'$ and $W = W'$. □

The basic algorithm outlined above can be accelerated using the following observations. In line 3 we can consider the subsets of D_i in increasing order of their size, that is, we first consider singletons. Let S_W be the subset found in line 4, corresponding to a set $W = \{n + j\}$. If $x^*(S_W) + x^*(i : D_i) \leq |S_W| - 1$ then we can remove $n + j$ from D_i as no LSEC with i as connector vertex, $j \in S$ and $n + j \notin S$ can be violated. After removing a vertex from D_i using this rule, it is useful to check whether more vertices can be removed (even the ones already checked). This is because removing a vertex from D_i decreases $x^*(i : D_i)$. The reduction also works when $|W| \geq 2$. In this case we eliminate all supersets of W if $x^*(S_W) + x^*(i : D_i) \leq |S_W| - 1$ where S_W is the set found in line 4 corresponding to W.

5.5 Heuristic separation of generalized lifted subtour elimination constraints

The separation algorithm for the GLSEC is a heuristic. The pseudo-code for separation when the connector vertex is a pickup is shown in Algorithm 4. In line 3 the heuristic loops over all candidates for connector vertices. In lines 4 to 7 the heuristic finds a candidate for the set S given the chosen connector vertex. This is done by finding a path of "highly connected" vertices in the fractional solution (line 4). The set of vertices formed by the path is extended by adding more vertices in lines 6 and 7. We prefer to add pickup vertices as this leaves the algorithm with more opportunities for constructing the T sets. In line 8 we construct the set W, containing the pickups from S that do not have their corresponding delivery in S. In lines 9 to 17 the heuristic goes through the vertices in W, trying to construct good T sets. The purpose of line 11 is to improve the set S given a set T. The optimization problems in lines 10 and 11 can be solved using maximum flow calculations. The sets T and S' constructed in line 10 and 11 may represent a violated GLSEC, but before checking for violation the heuristic tries to add more T sets to improve the inequality (in line 13). This is done similarly as for the first T set in line 10, but one must take care that the constructed T sets only have vertex i in common. In line 14 one checks whether the set S' along with the sets in Γ violates a GLSEC.

5.6 Heuristic separation of depot constraints

The separation routine for the depot constraint is a simple randomized construction heuristic. The heuristic starts with a set $S = \{0, i\}$ such that $x^*_{0i} > 0$. Given a candidate set S the best matching set T is computes as $T = D \setminus (S \cup \sigma(S))$. The heuristic checks whether S and T define a violated inequality. If not, it tries to extend S by adding a vertex j. The vertex is chosen as $\arg\max_{j \in P \cup D \setminus S} f(S \cup \{j\})$, where $f(S) = 2x^*(S) + x^*(S : D \setminus (S \cup \sigma(S))) - 2(|S| + 1)$. That is, $f(S)$ measures by how much S violates the inequality. If a violated inequality is detected, then the extension of the set S stops and the inequality is returned. In order to randomize the heuristic, a random number in the interval $[-0.3, 0.3]$ is added to $f(S)$ when the function is evaluated. The heuristic is applied several times with each possible start vertex, the first time without any randomization.

5.7 Heuristic separation of start-end constraints

A heuristic for the start-end constraints can be constructed by repeatedly solving two subproblems: 1) candidate sets S, S_1, S_2 and E_S have to be selected such that $S \subseteq P \cup D$, $S_1 \subseteq S \cap P$, $S_2 \subseteq S \cap D$ and $E_S \subseteq E(S)$ and such that inequality (16) is violated, and 2) one must determine whether (S, S_1, S_2, E_S) is SE-infeasible.

Algorithm 4 Heuristic separation of GLSEC

```
Input: fractional solution x*
Generate a weighted graph G' = (V,E') with E' = {e ∈ E : x*_e > 0} and weight w(e) = 1 − x*_e, e ∈ E'.
for all i ∈ {1,...,n} do
    Find shortest path p from i to n+i in G'.
    Let S contain the vertices in p.
    Improve S by adding vertices to S as long as x*(δ(S)) decreases.
    Improve S by adding pickup vertices to S as long as x*(δ(S)) does not increase.
    Set W = (S ∩ P) \ π(S).
    for all j ∈ W do
        T = argmax_{T∈P∪D} {x*(T) − |T| : {i, n+j} ⊆ T, T ∩ S = {i}}.
        S' = argmax_{S'∈P∪D} {x*(S') − |S'| : {i, n+i, j} ⊆ S', T ∩ S' = {i}}.
        set Γ = {T}.
        Add more T sets to Γ to improve inequality.
        if x*(S') + ∑_{T∈Γ} x*(T) > |S'| − 1 + ∑_{T∈Γ}(|T| − 2) then
            A violated inequality has been found. Store the inequality.
        end if
    end for
end for
```

We first turn our attention to the second subproblem. We have implemented an algorithm that checks whether (S,S_1,S_2,E_S) is SE-infeasible by enumerating feasible partial paths using only edges from $E_S \cup (0:S_1) \cup (S_2:2n+1)$. This algorithm precisely determines whether (S,S_1,S_2,E_S) is SE-infeasible or not, but requires that $|S|$ and $|E_S|$ be fairly small in order to have a reasonable running time.

A heuristic algorithm for checking whether (S,S_1,S_2,E_S) is SE-infeasible works as follows. First construct the sets $S_D = \{i \in S \cap D : i-n \notin S\}$ and $S_P = \{i \in S \cap P : i+n \notin S\}$. Here, S_D is the set of vertices in S that cannot be reached by a feasible path starting in vertex 0 that only visits vertices from S. The interpretation of S_P is similar, but considers paths ending at vertex $2n+1$. Define graphs $G_1 = (S, E_S \setminus (\delta(S_D) \cup E(S_D)))$ and $G_2 = (S, E_S \setminus (\delta(S_P) \cup E(S_P)))$. For each $i \in S_1$ define $U_i^1 \subset S$ as the set of vertices from S that are unreachable from i in G_1. Similarly, for each $j \in S_2$ define $U_j^2 \subset S$ as the set of vertices from S that are unreachable from j in G_2. If $U_i^1 \cap U_j^2 \neq \emptyset$ for all $i \in S_1, j \in S_2$ then (S,S_1,S_2,E_S) is SE-infeasible. If $U_i^1 \cap U_j^2 = \emptyset$ for some $i \in S_1$ and $j \in S_2$ then we do not know whether (S,S_1,S_2,E_S) is SE-feasible or not. The algorithm is therefore weaker than the enumerative algorithm, but is much faster in most cases. As a result we only use the enumerative algorithm when $|S| \leq 12$ ($|S| \leq 20$ at the root node). For larger sets we resort to the heuristic procedure.

The first subproblem determines the sets S, S_1, S_2 and E_S. We propose two heuristics for performing this selection. Given a fractional solution x^* both heuristics use the sets $A_1 = \{i \in P : x^*_{0,i} > 0\}$ and $A_2 = \{i \in D : x^*_{i,2n+1} > 0\}$ as a starting point. For each set $B_1 \subseteq A_1$ and $B_2 \subseteq A_2$ the first heuristic tries to determine a set S such that $S, S_1 = B_1, S_2 = B_2$ and $E_S = \{e \in E(S) : x^*_e > 0\}$ violates inequality (16). This set is determined as follows. Let $S^+ = B_1 \cup B_2$ and $S^- = (\sigma(B_1) \cup \pi(B_2)) \setminus S^+$. Here, S^+ is the set of vertices we need in S and S^- is the vertices that we do not want in S. The set S^- is chosen to make it more likely that the four sets are going to satisfy the conditions in Proposition 4.18. Now S is determined by $S = \arg\max_{S' \subseteq P \cup D}\{x^*(S') - |S'| : S^+ \subseteq S', S^- \cap S' = \emptyset\}$ using a maximum flow computation. After S has been computed the heuristic checks whether the four sets defines a violate inequality and whether (S,S_1,S_2,E_S) is SE-infeasible. If (S,S_1,S_2,E_S) cannot be proved to be SE-infeasible we apply a heuristic that greedily removes vertices from S and updates E_S accordingly. After each vertex removal (S,S_1,S_2,E_S) is checked for SE-infeasibility.

Priority	Name	Separation procedure
1	Subtour elimination constraints (SEC)	Exact.
2	Precedence constraints (PC)	Exact.
3	π-/σ-constraints (PSC)	Heuristic.
3	Generalized order constraints (GOC)	Exact for $m = 2$, heuristic for $m > 2$.
3	Generalized lifted subtour elimination constraints (GLSEC)	Heuristic.
3	Lifted subtour elimination constraints (LSEC)	Exact.
3	Depot constraints (DC)	Heuristic.
3	Doubly generalized order matching constraints (DGOMC)	Heuristic.
3	Start-end constraints (StEnC)	Heuristic.

Table 1: Overview of the implemented separation procedures

The second heuristic for the first subproblem works on a graph $G' = (V', E')$ where $V' = P \cup D$ and $E' = \{e \in E(V') : x^*_e > 0\}$. Each edge $e \in E'$ is assigned a weight $w_e = 1 - x^*_e$. For each set $B_1 \subseteq A_1$ and $B_2 \subseteq A_2$ the second heuristic builds a set $S = B_1 \cup B_2$ and considers each pair i, j such that $i \in B_1$ and $j \in B_2$. The heuristic computes a shortest path between i and j in the graph G' and adds all vertices on the path to S. One checks whether the sets $S, S_1 = B_1, S_2 = B_2$ and $E_S = e \in E(S) : x^*_e > 0$ violate inequality (16). If they do, a check is made to determine whether (S, S_1, S_2, E_S) is SE-infeasible.

Every time a violated start-end inequality is detected (by either heuristic) one checks whether the inequality can be strengthened by greedily adding more edges to E_S.

6 Implementation details

The branch-and-cut algorithm was implemented using CPLEX 10.1 and the Concert library. The search tree is explored using a best bound strategy. The algorithm branches on the x_{ij} variables. The variable to branch on is chosen using CPLEX's implementation of strong branching.

Table 1 provides an overview of the implemented separation routines. The first column shows the priority of the separation algorithm. Separation procedures with lower priority are always called before procedures with higher priority. Procedures with the same priority are called in the order listed in the table. If a separation procedure finds one or more valid inequalities, then the succeeding separation procedures are not called. If a separation procedure A fails to find a violated inequality and another procedure B with the same priority finds violated inequalities, then we start from procedure B the next time we invoke separation procedures of the same priority.

Adding violated valid inequalities to the LP relaxation normally improves the lower bound, but separating inequalities and solving linear programs with many constraints can be time consuming. It is therefore important to achieve a good balance between cutting and branching. In our algorithm we chose to only add a cut if it is violated by 0.2 or more. Also, for the first 100 branch-and-bound nodes we separate inequalities at every node, but thereafter we only separate inequalities at every 20th node.

CPLEX allows the user to activate a number of generic cuts such as disjunctive cuts and Gomory fractional cuts. Apparently it only uses the inequalities from the initial relaxation in order to generate new cuts. User cuts added on the fly are not taken into account. The cuts generated by CPLEX would not be very useful as our initial relaxation only contains equality (2) and inequalities (3). To make the generic cuts generated by CPLEX more useful our algorithm solves the root node twice. In the first phase the algorithm generates all violated inequalities that can be detected using the separation procedures described in Section 5. After this pass the algorithm extracts the generated inequalities that are binding when all generated cuts are applied. These inequalities along with equality (2) and inequalities (3) form the initial relaxation for the second pass. In the second pass CPLEX is allowed to generate generic cuts and our algorithm generates TSPPD specific inequalities. As a result CPLEX is able to generate its generic cuts using a richer initial relaxation.

Upper bounds are calculated using a simple large neighborhood search (LNS) heuristic. The LNS heuristic was proposed by Shaw [1998]; here we use a variant similar to what was proposed by Ropke and Pisinger [2006]. The heuristic improves an initial solution by repeatedly removing a set of requests (destroying the solution) and reinserting them in the solution (repairing the solution). The requests to remove are randomly selected and up to 50% of the requests in the solution can be removed. The requests are reinserted by ordering them in a random fashion and inserting them one at a time. When a request is inserted its pickup and delivery vertices are placed in the positions that least increase the overall cost. After performing a destroy and repair step the algorithm accepts the new solution if it is better than the current best and discards it otherwise. The descent stops after a predetermined number of iterations or after a given number of iterations without improvements. The full process is repeated n times using random starting solutions.

A simple local search procedure has been implemented to help intensify the search. It considers each request in turn. If relocating the pickup and delivery pair to another position in the tour decreases the overall cost, then that move is performed. This is continued as long as improving moves can be found. The relocate local search procedure is applied every time a new solution has been constructed by the insertion procedure and it is also applied to the partial tour obtained after removing requests with a 50% probability.

7 Computational results

All algorithms were implemented in C++ and were run on an AMD Opteron 250 computer (2.4 GHz) running Linux. Section 7.1 describes the data sets used for the computational tests and Section 7.2 investigates the impact of the valid inequalities proposed in the paper. Section 7.3 analyses the results of the branch-and-cut algorithm.

7.1 Data sets

We have performed tests on three data sets. The first set of instances was created by generating $2n+1$ points randomly in the square $[0,1000] \times [0,1000]$. The first point is used as the depot while the remaining points are paired to form requests (point i is paired with point $n+i$). The travel costs (c_{ij}) were set equal to the Euclidean distances. Instances with $n = 5, 10, 15, 20, 25, 30, 35$ were created, five for each size. The instances are named probnX, where X is one of the letters a, b, c, d and e used to distinguish between instances with the same size.

Two other data sets were proposed by Renaud et al. [2002]. One set of instances (the second data set in this paper) was generated from TSP instances from the TSPLIB (Reinelt [1991]). This data set contains instances ranging from 25 to 220 requests. We only used the instances with less than 51 requests in our tests. A TSP instance with an odd number of vertices is transformed into a TSPPD instance by designating the first vertex as the depot. Pickup-delivery pairs are formed by choosing a random vertex as pickup. Its corresponding delivery is selected using one of the following rules: *Rule A*: Select the delivery vertex from among the five closest neighbors (not yet selected) of the pickup vertex. *Rule B*: Select the delivery vertex from among the ten closest neighbors (not yet selected) of the pickup vertex. *Rule C*: Select the delivery vertex randomly from the remaining vertices. For each TSP instance three TSPPD instances were created, one for each of the three rules. The last letter in the instance name indicates the rule used to create the instance.

The third set of instances was also proposed by Renaud et al. [2002]. These instances were constructed as follows. A random TSP instance was constructed and solved to optimality. The first constructed vertex was chosen as depot and the tour was followed from this vertex to create pickup-delivery pairs. To create a pickup-delivery pair, the next unselected vertex on the tour was chosen as pickup and the delivery was chosen randomly among the remaining unselected vertices. This creates a TSPPD instance for which the optimal TSP solution also is optimal for the TSPPD. Ten instances with 50 requests and ten instances with 100 requests were created.

For all test sets travel costs c_{ij} are rounded to the nearest integer. All data sets can be downloaded from `www.diku.dk/~sropke`.

7.2 Impact of the valid inequalities

This section reports on the impact of the valid inequalities on the lower bound. Tables 2 and 3 show the impact of adding violated valid inequalities. Table 2 contains results for the first data set (randomly generated instances), while Table 3 contains results for the TSPLIB instance proposed by Renaud et al. [2002] (second data set). The basis of the comparison is the lower bound obtained by solving the linear relaxation of (1)–(6). The subtour elimination (4) and precedence constraints (5) are separated exactly when solving this model.

In this section we add an inequality if it is violated by at least 0.01. This is different from the branch-and-cut algorithm where we are more conservative and only add an inequality if it is violated by at least 0.2. We choose to add more inequalities in this section to better show the potential of each family of inequalities.

The columns in the tables should be interpreted as follows: n is the number of requests, *SEC+PC* reports the integrality gap of the linear relaxation of (1)–(6). The integrality gap is calculated as $100(\bar{z}-\underline{z}')/\bar{z}$, where $\underline{z}'$ is the value of the LP relaxation of (1)–(6) and $\bar{z}$ is the upper bound (either best known heuristic solution or optimal solution if it is known). The columns *GOC*, *DGOMC*, *LSEC*, *GLSEC*, *DC*, *PSC* and *StEnC* (see Table 1 for definitions of the abbreviations) show by how much the integrality gap is closed when the particular family of valid inequalities is added to the LP relaxation of (1)–(6). The entries are calculated as $100(\underline{z}-\underline{z}')/(\bar{z}-\underline{z}')$, where $\underline{z}$ is the lower bound after adding the particular family, and $\underline{z}'$ and $\bar{z}$ are defined as before. Entries are left blank if the initial LP relaxation solved the IP to optimality. The column *All* shows the gap closed when using all separation procedures described in Section 5 and the column *All+CPLEX* shows the gap closed when using the same inequalities as in the *All* column, plus the general purpose cuts provided by CPLEX.

The results show that the GOC and GLSEC are the inequalities that work best toward closing the integrality gap. For the first data set (Table 2) the LSEC, PSC and StEnC also turn out to be worthwhile. For the second data set (Table 3) the StEnC are less powerful while DGOMC, LSEC, GLSEC, PSC appear to be roughly equally important. It is interesting to note the behavior of the DGOMC on the second data set. The inequalities work best on the A instances while they are less effective on the C instances. We believe this is because the pickup and delivery vertices associated with the same request are close to each other in the A instances, and therefore it is easier to construct the T sets in the inequality that has to contain both the pickup and delivery of a selected request.

Tables 4 and 5 show the time it takes to solve the LP relaxations considered in Tables 2 and 3. The table groups results according to the instance size and reports the average computation time over all instances with the same size. The first column in each table indicates the instance size. Note that the last row provides an average over all instances in the data set.

We see that the StEnC are the most time consuming to separate for data set 1. It may seem counter-intuitive that it sometimes takes less time to separate all inequalities than it does just to separate StEnC inequalities. The reason for this behaviour is that the time consuming StEnC separation procedure is called more often when only this type of inequality along with SEC and PC are separated. When all the other inequalities are enabled the StEnC separation procedure is called less often as the other inequalities are able to close most of the gap. For data set 2 the most time consuming inequalities are GOC, StEnC and GOMC. The instances in the second data set (Table 5) are larger than in the first and we therefore observe larger running times for this data set.

7.3 Performance of the branch-and-cut algorithm

We now evaluate the performance of the branch-and-cut algorithm. We have tested the algorithm on the three data sets introduced in Section 7.1. The branch-and-cut algorithm uses all developed separation routines and the generic CPLEX cuts were also enabled. All experiments were performed within a time limit of four hours.

Results for the first, second and third data sets are presented in Tables 6, 7 and 8. The GOC and DGOMC separation procedures were disabled for the experiments on the third data set as the implemented separation procedures for these inequalities do not scale well with instance size.

The columns in the tables should be interpreted as follows: *Best IP* is the best integer solution, i.e. either the optimal solution (if known) or the best solution found by the heuristic, *seconds* is the computation time in seconds, a check mark in the *Opt* column indicates that the instance was solved to optimality, *Root LB* is lower bound at the root node, *Best LB* is the best lower bound proved after branching, *RLB/UB* reports the value $100\underline{z}/\bar{z}$, where $\underline{z}$ is the lower bound in the root node and $\bar{z}$ is the best integer solution, *BLB/UB* reports the value $100\underline{z}'/\bar{z}$, where $\underline{z}'$ is the best lower bound after branching, *BC nodes* is the number of nodes in the branch-and-cut tree, *#cuts* is the number of cuts applied (excluding CPLEX cuts).

The results reported in Tables 6 and 7 show that the algorithm is able to solve all instances with up 20 requests (42 vertices) within one minute, and most of the instances with 25 requests could be solved within the time limit. Some instances with 30 and 35 requests could be solved as well. Most of the larger instances in Table 7 seem to be out of reach of the current algorithm, although a few instances with 49 requests might be solved if the time limit was increased to one or two days (i.e. KROB99B and RAT99C). For the larger instances the algorithm manages to prove reasonable lower bounds after branching. These lower bounds are often within 5% of the upper bound.

Table 8 tells a different story. Here all instances are solved within a relatively short time even though the data set contains instances with up to 100 requests. The reason is that the optimal solutions for these instances are identical to the optimal TSP solutions as explained earlier. This implies that we are able to obtain very good lower bounds at the root node as it is well know that the linear relaxation of (3), (4) and (6) provides a tight lower bound on the optimal TSP solution. Adding TSPPD inequalities improves the lower bound even further.

We are aware of only two papers reporting computational testing of exact algorithms for the TSPPD. In Kalantari et al. [1985] instances with up to 15 requests are solved to optimality. The authors also solve an instance with 18 requests, but this instance has the same structure as the instances in our data set 3: the optimal solution to the TSPPD coincides with the optimal TSP solution. In Ruland and Rodin [1997] instances with up to 15 requests are also solved to optimality. Unfortunately the data sets used in the two aforementioned papers are not available, and therefore a direct comparison is not possible. We can conclude that our approach more than doubles the number of requests in the largest instance solved to optimality (35 requests); we do not consider the instances of the third data set here because of their particular structure.

8 Conclusions

We have presented new polyhedral results, valid inequalities and separation procedures for the *Traveling Salesman Problem with Pickup and Delivery*, a very difficult combinatorial optimization problem. Using these results we have devised an exact branch-and-cut algorithm capable of solving instances involving up to 35 pickups and delivery requests, thus more than doubling the previous record of 15 requests.

		Gap	Gap closed							Gap closed	
Name	n	SEC+PC	GOC	DGOMC	LSEC	GLSEC	DC	PSC	StEnC	All	All+CPLEX
prob5a	5	2.5	65.6	92.6	78.1	78.1	18.1	81.1	27.8	100.0	100.0
prob5b	5	0.0									
prob5c	5	0.0									
prob5d	5	0.0									
prob5e	5	5.0	76.0	14.4	100.0	100.0	0.0	50.2	100.0	100.0	100.0
prob10a	10	12.4	64.4	24.8	34.6	64.8	6.0	29.7	36.6	85.2	93.7
prob10b	10	16.1	88.5	35.1	20.6	50.9	6.7	37.2	56.8	95.6	100.0
prob10c	10	4.1	100.0	44.1	66.1	77.4	48.5	73.0	96.0	100.0	100.0
prob10d	10	5.0	41.3	23.3	10.4	45.3	5.8	50.8	44.2	94.3	100.0
prob10e	10	3.8	49.8	9.9	17.7	17.7	3.3	19.5	53.3	82.0	100.0
prob15a	15	11.0	54.4	22.8	27.4	30.0	11.2	28.8	30.5	73.4	84.7
prob15b	15	19.8	58.2	27.4	40.9	51.3	0.5	31.9	26.4	68.6	76.6
prob15c	15	6.2	97.0	34.2	62.0	69.7	16.0	68.3	58.1	100.0	100.0
prob15d	15	12.3	67.3	18.7	23.6	32.9	4.5	22.2	27.6	75.3	80.9
prob15e	15	5.2	100.0	44.0	38.7	59.1	11.4	41.4	21.9	100.0	100.0
prob20a	20	11.3	87.0	26.1	39.6	46.3	9.2	29.3	24.8	94.0	99.6
prob20b	20	13.2	77.2	12.7	15.6	16.3	11.8	23.5	18.3	88.2	91.7
prob20c	20	12.1	58.0	19.8	32.7	38.8	3.6	29.0	19.4	74.5	80.1
prob20d	20	8.7	69.4	12.9	16.0	25.3	4.7	19.8	18.0	76.7	80.4
prob20e	20	12.2	51.3	11.9	26.2	32.1	1.4	24.1	29.5	62.2	66.3
prob25a	25	17.0	38.1	9.2	8.2	19.2	2.7	6.2	4.9	42.0	46.9
prob25b	25	14.7	51.3	16.9	25.3	29.3	4.6	18.8	17.0	56.5	59.4
prob25c	25	14.2	52.2	14.5	27.0	30.2	2.9	21.5	22.5	63.4	67.8
prob25d	25	14.9	39.4	9.4	12.8	22.4	6.3	15.6	11.5	44.9	48.5
prob25e	25	12.2	58.9	18.5	15.8	26.3	2.0	20.7	7.3	70.1	74.0
prob30a	30	18.4	51.5	12.8	23.6	27.4	4.2	19.2	17.6	58.6	60.4
prob30b	30	16.4	55.1	25.5	26.8	35.5	9.9	27.9	22.5	63.8	66.1
prob30c	30	14.7	55.4	11.7	13.1	18.5	4.1	11.1	3.3	56.7	61.1
prob30d	30	12.5	61.0	7.1	21.1	22.5	1.7	18.4	7.8	68.5	70.8
prob30e	30	16.5	49.3	13.2	18.0	20.4	6.2	17.3	8.3	54.4	58.3
prob35a	35	14.9	55.2	14.1	13.8	23.4	6.2	17.1	15.9	63.5	66.1
prob35b	35	20.3	43.7	17.0	15.0	22.0	0.9	17.4	2.6	47.9	50.2
prob35c	35	16.6	56.3	15.6	17.4	28.4	8.0	16.8	13.4	64.2	67.1
prob35d	35	16.5	47.2	7.9	14.5	20.0	5.6	19.0	11.7	53.9	55.5
prob35e	35	17.5	38.0	6.9	8.8	10.8	1.6	9.1	2.3	42.5	44.3
Avg.		10.4	61.2	21.1	28.5	37.3	7.2	28.6	26.8	72.5	76.6

Table 2: Impact of the valid inequalities on data set 1.

		Gap	Gap closed							Gap closed	
Name	n	SEC+PC	GOC	DGOMC	LSEC	GLSEC	DC	PSC	StEnC	All	All+CPLEX
EIL51A	25	8.2	15.4	31.4	8.1	8.1	0.0	35.4	0.6	56.5	68.1
EIL51B	25	8.6	31.8	17.5	15.2	15.2	5.8	13.3	0.4	45.6	53.2
EIL51C	25	11.9	35.9	0.1	3.5	4.5	5.3	14.6	0.0	43.0	46.9
ST69A	34	12.1	37.1	28.9	17.1	18.6	0.0	19.1	1.0	59.1	62.1
ST69B	34	11.9	38.3	19.6	19.4	20.3	0.0	18.2	5.3	51.9	54.5
ST69C	34	15.6	46.3	11.9	18.3	22.4	5.7	19.7	1.2	54.5	57.0
EIL75A	37	8.4	18.1	18.6	11.9	12.0	0.0	14.4	0.0	33.2	38.1
EIL75B	37	10.6	24.5	11.8	17.9	21.9	0.8	14.8	2.5	31.9	34.5
EIL75C	37	9.0	28.6	3.9	5.5	12.3	2.1	12.5	1.3	34.6	40.1
PR75A	37	17.7	37.0	30.5	22.1	26.0	0.0	24.4	13.3	54.3	55.7
PR75B	37	15.1	39.2	18.4	15.0	24.6	0.9	21.3	13.9	47.9	51.6
PR75C	37	14.4	55.8	8.2	28.3	31.1	0.3	29.7	20.8	62.3	65.0
KROA99A	49	15.3	41.8	40.5	31.8	32.4	0.0	32.5	1.3	62.3	65.7
KROA99B	49	19.7	43.0	24.1	18.5	26.5	0.4	19.3	4.7	50.1	51.8
KROA99C	49	17.0	48.4	13.8	17.2	21.1	4.0	12.8	2.3	52.0	53.9
KROB99A	49	14.3	41.3	30.6	31.5	33.5	0.0	21.7	4.4	54.3	56.6
KROB99B	49	12.7	49.5	30.1	27.6	37.2	0.1	27.4	4.2	71.0	73.8
KROB99C	49	14.8	62.4	6.1	20.4	26.2	3.7	9.9	1.3	65.8	67.3
KROC99A	49	20.7	41.2	30.4	25.6	31.6	0.1	27.7	3.3	55.4	57.7
KROC99B	49	18.8	36.7	32.0	18.4	22.4	4.7	26.9	5.9	51.0	52.5
KROC99C	49	20.8	46.3	7.7	16.0	19.9	0.4	14.3	7.0	50.6	52.4
KROD99A	49	15.5	31.7	33.1	25.7	27.9	1.4	29.5	1.6	56.8	59.0
KROD99B	49	17.3	42.8	29.9	22.0	30.9	0.4	20.6	2.0	55.3	57.8
KROD99C	49	17.1	44.3	8.6	12.2	17.7	0.8	8.6	9.8	46.8	48.9
KROE99A	49	15.3	25.4	45.5	17.8	22.8	0.0	33.3	5.2	61.7	63.7
KROE99B	49	17.1	33.5	24.7	12.9	22.3	0.6	17.6	2.7	44.6	46.6
KROE99C	49	14.8	43.1	15.9	13.1	20.4	5.2	20.0	3.6	52.7	54.8
RAT99A	49	12.5	33.9	31.8	34.5	34.6	0.2	33.7	11.3	56.4	60.8
RAT99B	49	16.9	35.7	15.2	22.7	26.1	0.0	19.5	9.9	42.0	44.2
RAT99C	49	11.0	43.4	13.8	27.4	30.1	12.0	39.0	19.6	62.2	64.1
EIL101A	50	8.6	20.5	21.6	9.9	14.6	1.3	15.9	0.0	33.3	36.4
EIL101B	50	9.4	27.7	10.6	10.0	15.9	0.8	15.5	0.0	34.0	38.4
EIL101C	50	7.7	18.8	5.5	3.3	7.0	2.2	12.1	0.5	29.1	33.0
Avg.		14.0	37.0	20.4	18.2	22.4	1.8	21.1	4.9	50.4	53.5

Table 3: Impact of the valid inequalities on data set 2.

n	SEC+PC	GOC	DGOMC	LSEC	GLSEC	DC	PSC	StEnC	All	All+CPLEX
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
10	0.0	0.1	0.1	0.0	0.1	0.0	0.1	8.4	2.7	3.1
15	0.0	0.2	0.7	0.1	0.2	0.1	0.2	8.9	6.9	12.9
20	0.1	1.1	1.3	0.3	0.5	0.1	0.6	8.6	4.9	16.8
25	0.2	2.5	4.0	0.6	1.3	0.3	1.1	6.0	10.5	44.8
30	0.3	7.1	6.0	1.2	2.1	0.4	2.7	11.1	24.1	73.2
35	0.3	10.3	12.0	1.2	3.1	0.5	4.3	11.0	32.3	100.3
Avg.	0.2	5.3	5.6	0.7	1.4	0.3	2.2	8.7	18.5	56.8

Table 4: Computation time (in seconds) for the root node on data set 1.

n	SEC+PC	GOC	DGOMC	LSEC	GLSEC	DC	PSC	StEnC	All	All+CPLEX
25	0.1	2.3	4.3	0.3	0.5	0.2	0.9	0.3	10.1	29.7
34	0.4	5.7	22.1	1.3	2.0	0.5	3.4	1.0	54.4	129.0
37	0.4	6.2	21.6	1.4	2.8	0.4	4.1	9.5	43.0	120.3
49	1.3	25.6	80.5	4.3	7.3	1.6	14.0	58.1	130.6	429.8
50	1.3	18.6	60.2	2.7	4.5	1.6	10.0	1.4	77.5	192.2
Avg.	1.0	17.5	55.7	3.0	5.1	1.1	9.7	33.6	92.0	288.2

Table 5: Computation time (in seconds) for the root node on data set 2.

Name	n	Best IP	Seconds	Opt	Root LB	Best LB	RLB/UB	BLB/UB	BC Nodes	#cuts
prob5a	5	3585	0	✓	3585.00	3585.00	100.0	100.0	1	19
prob5b	5	2565	0	✓	2565.00	2565.00	100.0	100.0	1	6
prob5c	5	3787	0	✓	3787.00	3787.00	100.0	100.0	1	6
prob5d	5	3128	0	✓	3128.00	3128.00	100.0	100.0	1	6
prob5e	5	3123	0	✓	3123.00	3123.00	100.0	100.0	1	45
prob10a	10	4896	3	✓	4851.08	4896.00	99.1	100.0	4	134
prob10b	10	4490	2	✓	4490.00	4490.00	100.0	100.0	1	135
prob10c	10	4070	0	✓	4070.00	4070.00	100.0	100.0	1	93
prob10d	10	4551	1	✓	4551.00	4551.00	100.0	100.0	1	93
prob10e	10	4874	4	✓	4874.00	4874.00	100.0	100.0	1	97
prob15a	15	5150	8	✓	5030.48	5150.00	97.7	100.0	6	268
prob15b	15	5391	21	✓	5085.48	5391.00	94.3	100.0	45	580
prob15c	15	5008	0	✓	5008.00	5008.00	100.0	100.0	1	165
prob15d	15	5566	14	✓	5417.93	5566.00	97.3	100.0	15	390
prob15e	15	5229	0	✓	5229.00	5229.00	100.0	100.0	1	140
prob20a	20	5698	12	✓	5647.92	5698.00	99.1	100.0	9	438
prob20b	20	6213	20	✓	6125.61	6213.00	98.6	100.0	20	473
prob20c	20	6200	19	✓	6042.74	6200.00	97.5	100.0	28	444
prob20d	20	6106	17	✓	5985.96	6106.00	98.0	100.0	28	369
prob20e	20	6465	58	✓	6181.51	6465.00	95.6	100.0	115	962
prob25a	25	7332	14400		6632.90	7168.14	90.5	97.8	14377	3244
prob25b	25	6665	3138	✓	6259.86	6665.00	93.9	100.0	7952	2266
prob25c	25	7095	291	✓	6767.27	7095.00	95.4	100.0	878	1255
prob25d	25	7069	14323	✓	6515.85	7069.00	92.2	100.0	21627	3873
prob25e	25	6754	72	✓	6529.99	6754.00	96.7	100.0	125	704
prob30a	30	7309	14400		6745.14	7196.27	92.3	98.5	8296	3238
prob30b	30	6857	2843	✓	6461.94	6857.00	94.2	100.0	4528	2262
prob30c	30	7723	1891	✓	7258.73	7723.00	94.0	100.0	4269	2019
prob30d	30	7310	573	✓	7028.66	7310.00	96.2	100.0	1783	1372
prob30e	30	7213	14400		6683.29	7166.34	92.7	99.4	10523	3412
prob35a	35	7746	2104	✓	7338.12	7746.00	94.7	100.0	3541	1649
prob35b	35	7904	14400		7084.73	7496.03	89.6	94.8	6167	2764
prob35c	35	7949	14400		7509.94	7858.39	94.5	98.9	8427	3194
prob35d	35	7905	14400		7306.69	7686.77	92.4	97.2	9025	2944
prob35e	35	8530	14400		7690.94	8069.74	90.2	94.6	9791	3562
				28			96.5	99.5		

Table 6: Branch-and-cut results for data set 1 - Randomly generated instances

Name	n	Best IP	Seconds	Opt	Root LB	Best LB	RLB/UB	BLB/UB	BC Nodes	#cuts
EIL51A	25	464	94	✓	449.3	464.0	96.8	100.0	302	748
EIL51B	25	469	267	✓	449.0	469.0	95.7	100.0	1272	1167
EIL51C	25	488	14400		455.6	486.7	93.4	99.7	15230	4325
ST69A	34	764	7142	✓	726.7	764.0	95.1	100.0	7647	2400
ST69B	34	771	14400		726.1	763.0	94.2	99.0	9084	2651
ST69C	34	793	14400		737.0	773.4	92.9	97.5	9407	3243
EIL75A	37	583	14400		552.0	574.0	94.7	98.5	9097	3914
EIL75B	37	601	14400		559.6	581.9	93.1	96.8	9484	3696
EIL75C	37	590	14400		556.3	580.5	94.3	98.4	7921	3106
PR75A	37	130531	14400		119920.0	125386.0	91.9	96.1	6770	2325
PR75B	37	128397	14400		118855.0	123689.0	92.6	96.3	9328	2298
PR75C	37	124509	14400		118081.0	123313.0	94.8	99.0	8556	2331
KROA99A	49	24980	14400		23565.8	24485.0	94.3	98.0	7657	2590
KROA99B	49	26552	14400		23981.0	24957.5	90.3	94.0	5946	2416
KROA99C	49	25769	14400		23664.0	24715.8	91.8	95.9	3740	3050
KROB99A	49	25631	14400		23984.9	25046.9	93.6	97.7	5197	2752
KROB99B	49	25384	14400		24492.9	25166.3	96.5	99.1	7095	2315
KROB99C	49	25795	14400		24499.6	25449.7	95.0	98.7	4231	2826
KROC99A	49	26146	14400		23817.6	24787.7	91.1	94.8	5994	2435
KROC99B	49	25602	14400		23241.4	24194.3	90.8	94.5	5239	2563
KROC99C	49	26065	14400		23375.5	24293.7	89.7	93.2	3622	2597
KROD99A	49	25392	14400		23728.9	24638.3	93.5	97.0	7570	2550
KROD99B	49	26179	14400		24191.8	25020.5	92.4	95.6	5574	2624
KROD99C	49	26041	14400		23696.4	24759.5	91.0	95.1	4262	2757
KROE99A	49	25879	14400		24402.0	25174.8	94.3	97.3	4124	2925
KROE99B	49	26591	14400		24107.2	25123.1	90.7	94.5	5835	2936
KROE99C	49	26021	14400		24200.7	25169.3	93.0	96.7	3746	2821
RAT99A	49	1401	14400		1331.2	1368.4	95.0	97.7	7262	2671
RAT99B	49	1464	14400		1321.4	1371.2	90.3	93.7	4814	2488
RAT99C	49	1370	14400		1314.9	1360.2	96.0	99.3	3880	2899
EIL101A	50	695	14400		656.6	674.6	94.5	97.1	7999	3088
EIL101B	50	705	14400		662.7	680.2	94.0	96.5	7781	3189
EIL101C	50	690	14400		654.5	668.5	94.9	96.9	4430	3065
				3			93.4	97.1		

Table 7: Branch-and-cut results for data set 2 - Renaud et al. TSPLIB instances

Name	n	Best IP	Seconds	Opt	Root LB	Best LB	RLB/UB	BLB/UB	BC Nodes	#cuts
N101P1	50	799	8	✓	799.00	799.00	100.0	100.0	1	312
N101P2	50	729	11	✓	728.33	729.00	99.9	100.0	2	159
N101P3	50	748	4	✓	748.00	748.00	100.0	100.0	1	254
N101P4	50	807	4	✓	807.00	807.00	100.0	100.0	1	137
N101P5	50	783	4	✓	783.00	783.00	100.0	100.0	1	54
N101P6	50	755	4	✓	755.00	755.00	100.0	100.0	1	31
N101P7	50	767	5	✓	767.00	767.00	100.0	100.0	1	182
N101P8	50	762	6	✓	762.00	762.00	100.0	100.0	1	65
N101P9	50	766	9	✓	764.40	766.00	99.8	100.0	3	175
N101P10	50	754	8	✓	754.00	754.00	100.0	100.0	1	309
N201P1	100	1039	135	✓	1039.00	1039.00	100.0	100.0	1	547
N201P2	100	1086	90	✓	1086.00	1086.00	100.0	100.0	1	394
N201P3	100	1070	75	✓	1070.00	1070.00	100.0	100.0	1	425
N201P4	100	1050	142	✓	1050.00	1050.00	100.0	100.0	1	546
N201P5	100	1052	121	✓	1052.00	1052.00	100.0	100.0	1	510
N201P6	100	1059	151	✓	1058.93	1059.00	100.0	100.0	2	521
N201P7	100	1036	128	✓	1036.00	1036.00	100.0	100.0	1	713
N201P8	100	1079	192	✓	1079.00	1079.00	100.0	100.0	1	1302
N201P9	100	1050	88	✓	1050.00	1050.00	100.0	100.0	1	789
N201P10	100	1085	198	✓	1085.00	1085.00	100.0	100.0	1	954
				20			100.0	100.0		

Table 8: Branch-and-cut results for data set 3 - Renaud et al. instances generated from an optimal TSP tour.

Acknowledgements

This work was partially supported by the Canadian National Sciences and Engineering Research Council under grants 227837-04 and 39682-05. This support is gratefully acknowledged.

References

E. Balas, M. Fischetti, and W.R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241-265, 1995.

T. Christof and A. Löbel. Porta – a polyhedron representation and transformation algorithm. http://www.iwr.uni-heidelberg.de/groups/compot/software/PORTA/index.html. ZIB, Konrad-Zuse-Zentrum für Informationstechnik Berlin.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573-586, 2006.

J.-F. Cordeau, G. Laporte, and S. Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. In B.L. Golden, S. Raghavan, and E.A. Wasil, editors, the Vehicle Routing Problem, Latest Advances and Challenges. Springer, Boston, 2007.

I. Dumitrescu. Polyhedral results for the pickup and delivery travelling salesman problem. Technical Report CRT-2005-27, Centre for Research on Transportation, Montreal, 2005. URL http://www.crt.umontreal.ca/~irina/CRTdumitrescu.pdf.

M.T. Fiala Timlin and W.R. Pulleyblank. Precedence constrained routing and helicopter scheduling: heuristic design. *Interfaces*, 22(3):100-111, 1992.

M. Gendreau, A. Hertz, and G. Laporte. The traveling salesman problem with backhauls. *Computers & Operations Research*, 23:501-508, 1996.

P. Healy and R. Moll. A new extension of local search applied to the dial-a-ride problem. *European Journal of Operational Research*, 83:83-104, 1995.

H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145:126-139, 2004.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, pages 33-65. Springer, Boston, 2005.

B. Kalantari, A.V. Hill, and S.R. Arora. An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research*, 22:377-386, 1985.

G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, Chichester, 1988.

G. Reinelt. TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, 3:376-384, 1991.

J. Renaud, F.F. Boctor, and J. Ouenniche. A heuristic for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, 29:1129-1141, 2002.

J. Renaud, F.F. Boctor, and G. Laporte. Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, 29:1129-1141, 2002.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455-472, 2006.

K.S. Ruland. *Polyhedral Solution to the Pickup and Delivery Problem*. PhD thesis, Sever Institute of Washington University, 1994.

K.S. Ruland and E.Y. Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & Mathematcis with Application*s, 33:1-13, 1997.

M.W.P. Savelsbergh. An efficient implementation of local search algorithms for constrained routing problems. European Journal of Operational Research, 47:75-85, 1990.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming),* volume 1520 of *Lecture Notes in Computer Science*, pages 417-431, Springer, Berlin, 1998.