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 One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals

Güneş Erdoğan
Gilbert Laporte
Roberto Wolfler Calvo

August 2013

CIRRELT-2013-46

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
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals

Güneş Erdoğan[1], Gilbert Laporte[2], Roberto Wolfler Calvo[3]

[1] Southampton Management School, University of Southampton, Highfield, Southampton, SO17 1BJ, United Kingdom

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Management Sciences, HEC Montréal, 3000, Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[3] Laboratoire d'Informatique de Paris Nord (IPN), Institut Galilée, Université Paris-Nord, 99, Avenue Jean-Baptiste Clément, 93430 Villetaneuse, Paris, France

Abstract. This study introduces the One-Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals (1-PDTSP-DI), a generalization of the One Commodity Pickup and Delivery Traveling Salesman Problem (1-PDTSP). In the 1-PDTSP-DI, the vertices are required to have an inventory lying between given lower and upper bounds and initially have an inventory which does not necessarily lie between these bounds. The problem consists of redistributing the inventory among the vertices, using a single capacitated vehicle, so that the bounding constraints are satisfied and there positioning cost is minimized. An application of this problem arises in rebalancing operations for shared bicycle systems. The repositioning subproblem associated with a fixed route is shown to be a minimum cost network problem. Two integer programming formulations for the 1-PDTSP-DI are presented, together with valid inequalities adapted from constraints derived in the context of other routing problems. One of these formulations is appropriate for a Benders Decomposition scheme. Computational results for instances adapted from the 1-PDTSP are provided for both approaches.

Keywords. Traveling salesman, pickup and delivery, branch-and-cut.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 39682-10. This support is gratefully acknowledged.

---

* Corresponding author: Gilbert.Laporte@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1 Introduction

The *One-Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals* (1-PDTSP-DI) is defined on a complete directed graph $G = (V, A)$, where $V = \{0, \ldots, n\}$ is the set of vertices and A is the set of arcs. Vertex 0 is called the *depot* while the remaining vertices are called *customers*. Associated with each vertex $i \in V$, are three parameters (l_i, b_i, u_i) corresponding respectively to the lower bound, the current supply, and the upper bound of the feasible amount of a commodity at the vertex. With each arc $(i, j) \in A$ is associated a travel cost t_{ij}, and the cost associated with the handling of one commodity unit is denoted as h. A vehicle of capacity Q leaves the depot, performs a tour visiting each vertex at most once to perform a pickup or a delivery, and returns to the depot. At the end of the tour, the resulting inventory at every vertex i must lie within the interval $[l_i, u_i]$. The objective is to minimize the total travel and handling cost.

The 1-PDTSP-DI is a generalization of the *One-Commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP) studied by Hernández-Pérez and Salazar-González (2004, 2007), in which a single capacitated vehicle visits customers to pick up and deliver the same commodity. Any instance of the 1-PDTSP can be converted to an instance of the 1-PDTSP-DI, in which a pickup vertex i with supply $b_i^{'} > 0$ is assigned the parameters $(0, b_i^{'}, 0)$, a delivery vertex j with demand $b_j^{'} < 0$ is assigned the parameters $(-b_j^{'}, 0, -b_j^{'})$, and a vertex k with $b_k^{'} = 0$ is represented by two copies of the vertex with parameters $(0, 1, 0)$ and $(1, 0, 1)$ connected by two arcs with a cost of $-M$. This relationship proves that the 1-PDTSP-DI is NP-hard. The demand intervals introduce a degree of flexibility associated with *transshipment vertices*, i.e. vertices $i \in V$ for which $l_i \leq b_i \leq u_i$. A transshipment vertex may or may not be visited, which may help decrease the cost of routing by supplying or demanding commodities as required. Notably, the *Swapping Problem* (SP) introduced by Anily and Hassin (1992) also involves pickups and deliveries with transshipment vertices and multiple commodities. Branch-and-cut algorithms for the SP were developed by Bordenave et al. (2009, 2012) and by Erdoğan et al. (2010). Bordenave et al. (2010) have also developed construction and improvement heuristics for the SP.

Applications of the 1-PDTSP-DI are encountered in rebalancing operations arising in shared bicycle systems, which have attracted the attention of several groups of researchers in recent years. These have been studied from several perspectives: evaluating the mobility patterns of users, determining the number and location of stations, maximizing user satisfaction, and minimizing the cost of relocating the bicycles. Raviv et al. (2013) categorize the bicycle relocation problems as *static* and *dynamic*, which occur when the system activity level is low and high, respectively. Benchimol et al. (2012) focus on the *Static Stations Balancing Problem*, and provide a 9.5-approximation algorithm for this problem. Raviv et al. (2013) study the *Static Repositioning Problem* (SRP), in which the objective is to minimize a convex nonlinear function representing user dissatisfaction. The authors present four integer programming formulations for the SRP, together with computational results. Another closely related problem is that of Chemla et al. (2013), where a bicycle station can be visited more than once for a pickup or a delivery. The authors provide a branch-and-cut algorithm as well as a tabu search algorithm for this problem.

Most of the problems cited above are based on a single demand or supply value for every customer, which restricts the vehicle to picking or delivering a preset number of commodities. The 1-PDTSP-DI is therefore more general, and empirically more difficult, because these values must lie within an interval. It is a special case of the models presented in Raviv et al. (2013) since these authors use a convex user dissatisfaction function which can be set to zero inside the interval and to infinity outside it. Let us define the *deficit* of vertex i as $d_i = \max\{l_i - b_i, 0\}$ and its *excess* as $e_i = \max\{b_i - u_i, 0\}$. Transforming an instance of the 1-PDTSP-DI into one of the 1-PDTSP by setting the demand (supply) of a vertex to be equal to its deficit (excess) may yield an infeasible instance since the sum of the demands may not be equal to the sum of supplies.

In this study, we provide two exact algorithms for the 1-PDTSP-DI. To gain a better insight into the 1-PDTSP-DI, we first study the subproblem of computing pickup and delivery quantities when the vehicle route is fixed. We show that this subproblem is a minimum cost network flow problem (MCNFP), whether $h = 0$ or not. We also present a model for the general problem consisting of simultaneously determining the vehicle route as well as the pickup and delivery quantities. We develop a standard branch-and-cut algorithm as well as a Benders decomposition algorithm, and we present computational results for both cases.

The remainder of this paper is organized as follows. In Section 2, we study the subproblem corresponding to a fixed route, without and with handling cost. In Section 3 we present an integer linear programming formulation for the 1-PDTSP-DI based on our findings in Section 2, as well as valid inequalities we have adapted from the routing literature. A Benders decomposition scheme for the integer linear programming formulation is provided in Section 4. A unified branch-and-cut algorithm capable of handling both algorithms is described in Section 5. This is followed by computational results in Section 6, and by conclusions in Section 7.

2 The Fixed Route Subproblem

We first focus on the subproblem of determining the pickup and delivery amounts when the vehicle route is fixed. For the sake of simplicity, we assume that the vertices are numbered in the order they are visited. The two cases for which $h = 0$ and $h \geq 0$ will be treated separately.

2.1 The fixed route subproblem with no commodity handling cost

The fixed route subproblem without handling cost is called the 1-PDTSP-DIF and is defined on an auxiliary graph $\hat{G} = (\hat{V}, \hat{A})$. Denote the set of vertices by $\hat{V} = \hat{V}_1 \cup \hat{V}_2$, with $\hat{V}_1 = V$ and $V_2 = \{n+1\}$. The supply of vertex $i \in \hat{V}_1$ is $\hat{b}_i = b_i$, and $\hat{b}_{n+1} = -\sum_{i \in V} b_i$. Denote the set of arcs by $\hat{A} = \hat{A}_1 \cup \hat{A}_2$, where $\hat{A}_1$ and $\hat{A}_2$ are constructed as follows. For every vertex $i \in \hat{V}_1 \setminus \{0, n\}$, insert an arc $(i, i+1)$ into $\hat{A}_1$, with cost 0, lower bound 0 and upper bound Q. These arcs represent the number of units transported to the next vertex. We also insert two arcs $(0, 1)$ and $(n, 0)$ into $\hat{A}_1$, with cost 0, lower bound 0 and upper bounds $\min\{b_0, Q\}$ and Q, respectively. These arcs represent number of units leaving and entering the depot. The flow on the arc $(0, 1)$ is also bounded above by the supply at the depot, in order to avoid commodities from flowing through the depot. For every vertex $i \in \hat{V}_1$, insert an arc $(i, n+1)$ into $\hat{A}_2$, with cost 0, lower bound l_i and upper bound u_i. This arc represents the final amount of the commodity left at vertex i. Define the set of arcs leaving vertex i as $\delta^+(i)$, and the set of arcs entering vertex i as $\delta^-(i)$. Let z_{ij} denote the commodity flow on arc (i, j). We write $z(S)$ to denote the sum of the z variables in arc set S, i.e. $z(S) = \sum_{(i,j) \in S} z_{ij}$. We then have to solve

$$\text{(1-PDTSP-DIF)} \quad z(\delta^+(i)) - z(\delta^-(i)) = \hat{b}_i \quad (i \in \hat{V}) \tag{1}$$

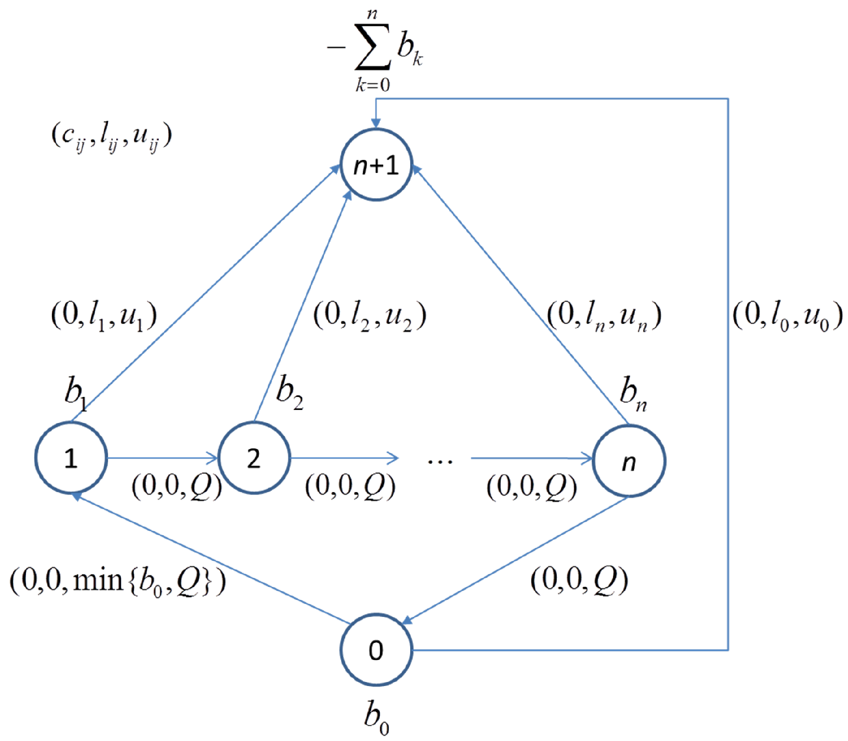

Figure 1: Instance of the 1-PDTSP-DIF

$$l_i \le z_{ij} \le u_i \quad ((i,j) \in \hat{A}_2) \tag{2}$$
$$0 \le z_{01} \le \min\{b_0, Q\} \tag{3}$$
$$0 \le z_{ij} \le Q \quad ((i,j) \in \hat{A}_1 \setminus \{(0,1)\}). \tag{4}$$

Figure 1 depicts an instance of the 1-PDTSP-DIF. Using an enhanced capacity scaling algorithm, together with the fact that the number of arcs is $O(n)$, the problem stated above can be solved in $O(n^2 \log n^2)$ time (Ahuja et al. 1993).

2.2 The fixed route subproblem with commodity handling cost

The fixed route subproblem with handling cost is called the 1-PDTSP-DIHF and is defined on an auxiliary graph $\bar{G} = (\bar{V}, \bar{A})$. Denote the set of vertices by $\bar{V} = \bar{V}_1 \cup \bar{V}_2 \cup \bar{V}_3$, with $\bar{V}_1 = V$ and $\bar{V}_3 = \{2n+2\}$. We construct $\bar{V}_2$ by including a vertex for every vertex $i \in V$, where the copy of vertex i in $\bar{V}_2$ is $n+1+i$. Set the supply of vertex $i \in \bar{V}_1$ as $\bar{b}_i = b_i$ and the demand of vertex $n+1+i \in \bar{V}_2$ as $\bar{b}_{n+1+i} = -b_i$. Let $\bar{b}_{2n+2} = 0$. Denote the set of arcs $\bar{A} = \bar{A}_1 \cup \bar{A}_2 \cup \bar{A}_3$, where $\bar{A}_1$, $\bar{A}_2$, and $\bar{A}_3$ are constructed as follows. For vertices $i \in \bar{V}_1 \setminus \{0, n\}$, insert an arc $(i, i+1)$ into $\bar{A}_1$, with cost 0, lower bound 0 and upper bound Q. These arcs represent the number of units transported to the next vertex. We also insert two arcs $(0, 1)$ and $(n, 0)$ into $\bar{A}_1$, with cost h, lower bound 0 and upper bound $\min\{b_0, Q\}$ and Q, respectively. These arcs represent the commodity amounts carried out of and into the depot, respectively, and also account for the handling cost at the depot. Note that the flow on the arc $(0, 1)$ is also bounded above by the supply at the depot in order to avoid commodities from flowing through the depot. For every vertex in $\bar{V}_1$, insert an arc $(i, n+1+i)$ into $\bar{A}_2$, with cost 0, lower bound l_i and upper bound u_i. These arcs represent the amount of commodity units left at vertex i after the visit of the vehicle. For every vertex in $V_2 \setminus \{n+1\}$, insert two arcs $(n+1+i, 2n+2)$ and $(2n+2, n+1+i)$ into $\bar{A}_3$, with cost h, lower bound 0 and upper bound Q. For vertex $n+1$, also insert two arcs $(n+1, 2n+2)$ and $(2n+2, n+1)$ into $\bar{A}_3$, with cost 0, lower bound 0 and upper bound Q. These arcs represent the number of units unloaded and loaded at vertex i, respectively. Note that the handling cost at the depot has been modeled in a different way from that of the other vertices, since handling occurs *twice* at the depot, once in the beginning and once in the end of the tour. Let us define $\bar{A}_3' = \bar{A}_3 \cup \{(0, 1), (n, 0)\}$ as the set of arcs with a positive cost, for the sake of simplicity. The resulting formulation for this subproblem is then

$$
\begin{aligned}
\text{(1-PDTSP-DIFH)} \quad & \text{minimize} \sum_{(i,j)\in \bar{A}_3'} h\, z_{ij} && (5)\\
& \text{subject to } z(\delta^+(i)) - z(\delta^-(i)) = \bar{b}_i \quad (i \in \bar{V}) && (6)\\
& l_i \le z_{ij} \le u_i \quad ((i,j) \in \bar{A}_2) && (7)\\
& 0 \le z_{01} \le \min\{b_0, Q\} && (8)
\end{aligned}
$$

$$0 \leq z_{ij} \leq Q \quad ((i,j) \in \bar{A} \setminus \bar{A}_2 \cup \{(0,1)\}). \tag{9}$$

Figure 2 depicts an instance of the 1-PDTSP-DIHF. As in the case of 1-PDTSP-DIHF, this problem can be solved in $O(n^2 \text{log} n^2)$ time (Ahuja et al. 1993).

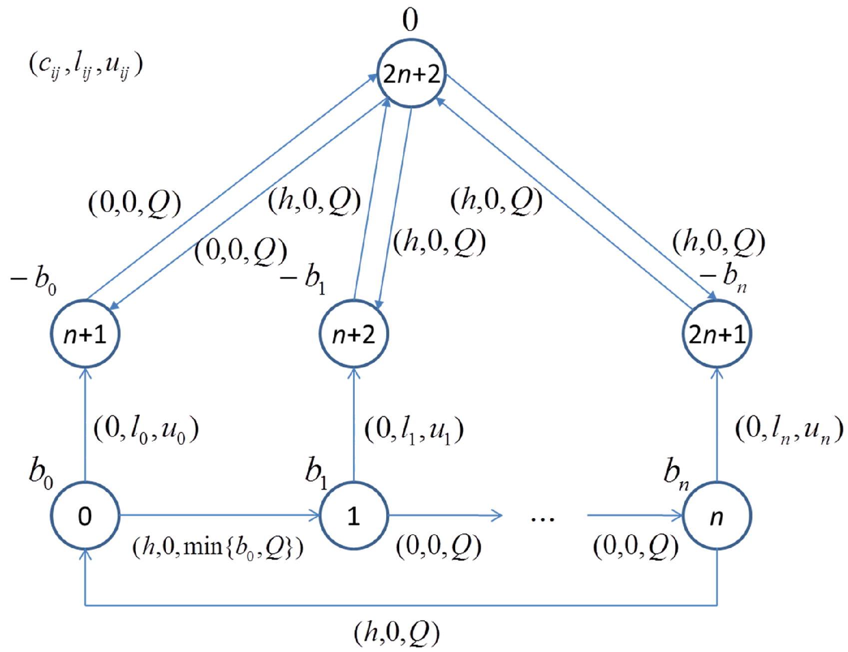

Figure 2: Instance of the 1-PDTSP-DIHF.

3 Integer Programming Formulation and Valid Inequalities for the General Problem

We now construct an integer programming formulation for 1-PDTSP-DI based on the minimum cost network flow formulation for the 1-PDTSP-DIHF presented in Section 2. Although it is possible to construct two different models, based on both network flow formulations, we opt to use the 1-PDTSP-DIHF model of Section 2.2 since it can handle both cases. To improve the lower bounds yielded by the relaxation of the formulation, we first strengthen the capacity constraints (17) using the implications of integrality. We next adapt the clique cluster inequalities for the 1-PDTSP (Hernández-Pérez and Salazar-González 2007) to the 1-PDTSP-DIH. Finally, we adapt the arc-vertex inequalities, the strong connectivity constraints, and the strong 2-matching inequalities from the Covering Tour Problem (CTP) (Gendreau et al. 1997), the similarity of which enables us to adapt its valid inequalities. The proofs of validity of these constraints are identical for both problems.

3.1 Formulation

Before presenting the formulation, we redefine $\bar{A}_1$ as $\bar{A}_1 = \{(i,j) : i,j \in V, i \neq j\}$. With this new definition, the circuits on which the pickup and delivery decisions were considered in Section 2 become cliques, and we can incorporate routing decisions to the model. Also define the vehicle requirement of a subset $S \subset V$ as $r(S) = \max\{\lceil \sum_{i \in S}(b_i - u_i)/Q \rceil, \lceil \sum_{i \in S}(l_i - b_i)/Q \rceil\}$. Denote the mandatory vertices as $\bar{T} = \{i \in \bar{V}_1 : i = 0 \text{ or } d_i > 0 \text{ or } e_i > 0\}$. Let x_{ij} be equal to 1 is the vehicle travels from vertex i to j, and 0 otherwise. We write $x(S)$ to denote the sum of the x variables in arc set S, i.e. $x(S) = \sum_{(i,j) \in S} x_{ij}$. Finally, let y_i be equal to 1 if vertex i is visited, and 0 otherwise. The formulation is

$$\text{(1-PDTSP-DIH)} \quad \text{minimize} \sum_{(i,j) \in \bar{A}_1} t_{ij} x_{ij} + \sum_{(i,j) \in \bar{A}'_3} h\, z_{ij} \tag{10}$$

$$\text{subject to } x(\delta^+(i)) = y_i \quad (i \in \bar{V}_1) \tag{11}$$

$$x(\delta^-(i)) = y_i \quad (i \in \bar{V}_1) \tag{12}$$

$$x(\delta^+(S)) \geq r(S) \quad (S \subset \bar{V}_1) \tag{13}$$

$$x(\delta^-(S)) \geq r(S) \quad (S \subset \bar{V}_1) \tag{14}$$

$$z(\delta^+(i)) - z(\delta^-(i)) = \bar{b}_i \quad (i \in \bar{V}) \tag{15}$$

$$z_{0j} \leq \min\{b_0, Q\} x_{ij} \quad ((0,j) \in \bar{A}_1) \tag{16}$$

$$z_{ij} \leq Q x_{ij} \quad ((i,j) \in \bar{A}_1 : i \neq 0) \tag{17}$$

$$l_i \leq z_{ij} \leq u_i \quad ((i,j) \in \bar{A}_2) \tag{18}$$

$$z_{ij} \leq Q y_i \quad ((i,j) \in \bar{A}_3) \tag{19}$$

$$x_{ij} = 0 \text{ or } 1 \quad ((i,j) \in \bar{A}) \tag{20}$$

$$y_i = 0 \text{ or } 1 \quad (i \in \bar{V}_1 \setminus \bar{T}) \tag{21}$$

$$y_i = 1 \quad (i \in \bar{T}) \tag{22}$$

$$z_{ij} \geq 0 \quad ((i,j) \in \bar{A}). \tag{23}$$

Equalities (11) set the outflow of a vertex to 1 if it is visited, and to 0 otherwise. Similarly, Equalities (12) set the inflow of a vertex to 1 if it is visited, and to 0 otherwise. Inequalities (13) and (14) are the connectivity constraints which define the minimum outdegree and indegree of a subset of vertices as a function of the capacity requirements. Equalities (15) enforce the flow conservation conditions. Constraints (16) and (17) are capacity constraints, which restrict the flow of commodities on an arc to the capacity of the vehicle. Constraints (19) forbid loading and unloading operations at a vertex if it is not visited. Constraints (18) ensure that the demand interval requirements are met. Constraints (20) and (21) are integrality constraints. Constraints (22) force the depot, the vertices with positive deficits, and the vertices with positive excesses to be visited. Finally, constraints (23) are non-negativity constraints.

3.2 Implications of integrality

We now strengthen the capacity constraints (16) and (17) by using coefficients which imply that a vehicle leaving a vertex should have sufficient capacity to pick up the excess of the vertex if it has an excess, or to deliver the deficit of the vertex if it has a deficit. Let α_{ij} denote the upper bound on z_{ij}:

$$\alpha_{ij} = \begin{cases} Q & (i,j \neq 0 \text{ and } b_i \geq l_i \text{ and } b_j \leq u_j) \\ Q + b_i - l_i & (i,j \neq 0 \text{ and } b_i < l_i \text{ and } b_j \leq u_j) \\ Q + u_j - b_j & (i,j \neq 0 \text{ and } b_i \geq l_i \text{ and } b_j > u_j) \\ \min\{Q + b_i - l_i, Q + u_j - b_j\} & (i,j \neq 0 \text{ and } b_i < l_i \text{ and } b_j > u_j) \\ \min\{b_0, Q\} & (i = 0 \text{ and } b_j \leq u_j) \\ \min\{b_0, Q + u_j - b_j\} & (i = 0 \text{ and } b_j > u_j) \\ \min\{u_0, Q\} & (j = 0). \end{cases}$$

With this definition of α_{ij}, we can unify and strengthen (16) and (17) as

$$z_{ij} \leq \alpha_{ij} x_{ij} \quad ((i,j) \in \bar{A}_1). \tag{24}$$

3.3 Clique cluster inequalities

Clique cluster inequalities are an adaptation of the *rank inequalities* for the *Stable Set Problem.* Hernández-Pérez and Salazar-González (2007) have adapted the clique inequalities to the 1-PDTSP by considering subsets of vertices $W_1, ..., W_m$ satisfying the following conditions: 1) all subsets intersect at a single vertex v: $(W_k \cap W_l = \{v\}, 1 \leq k < l \leq m)$; 2) the capacity requirement of any subset does not exceed the vehicle capacity $(r(W_k) \leq 1, 1 \leq k \leq m)$; 3) the capacity requirement of the union of any two subsets exceeds the vehicle capacity $(r(W_k \cup W_l) \geq 2, 1 \leq k < l \leq m)$. By construction of the customer subsets, only customers in one subset W_k can be visited consecutively, implying $x(\delta^+(W_k)) = 1$. The remaining subsets W_l must satisfy $x(\delta^+(W_l)) \geq 2$. Summing up the inequality and the following inequalities we obtain

$$\sum_{k=1}^{m} x(\delta^+(W_k)) \geq 2m - 1. \tag{25}$$

We emphasize that the transshipment vertices are an exception, since they do not exist in 1-PDTSP, and their existence may result in $r(W_k) = 0$ or $(r(W_k \backslash \{v\}) = 0$. To handle this exception, we modify the second condition as follows: 2) the capacity requirement of any customer subset is exactly one, even when the intersection vertex is deleted $(r(W_k) = 1$ and $r(W_k \setminus \{v\}) = 1$, $1 \leq k \leq m)$.

3.4 Arc-vertex constraints

Proposition 1. The inequalities

$$x_{ij} \leq y_i \quad ((i, j) \in \bar{A}_1) \tag{26}$$

and

$$x_{ij} \leq y_j \quad ((i, j) \in \bar{A}_1) \tag{27}$$

are valid for 1-PDTSP-DIH.

3.5 Strong connectivity constraints

Proposition 2. The following inequalities are valid for 1-PDTSP-DIH:

$$\sum_{i \in S, j \in \bar{V}_1 \setminus S} x_{ij} \geq 1 \quad (S \subset \bar{V}_1 : 2 \leq |S| \leq |\bar{V}_1| - 2, \bar{T} \setminus S \neq \emptyset, S \cap \bar{T} \neq \emptyset). \quad (28)$$

We refer the reader to the study by Gendreau et al. (1997) for a proof of Proposition 2.

3.6 Strong 2-matching constraints

Proposition 3. The following inequalities are valid for 1-PDTSP-DIH:

$$\sum_{i,j \in H : i \neq j} x_{ij} + \sum_{(i,j) \in A'} (x_{ij} + x_{ji}) \leq \sum_{i \in H} y_i + \frac{1}{2}(|A'| - 1), \qquad (29)$$

for all $H \subset \bar{V}_1$ and $A' \subset \bar{A}_1$ satisfying
(i) $|\{i,j\} \cap H| = 1 \quad ((i,j) \in A')$,
(ii) $\{i,j\} \cap \{k,l\} = \emptyset \quad ((i,j) \neq (k,l) \in A')$,
(iii) $|A'| \geq 3$ and odd.

We refer the reader to the study by Gendreau et al. (1997) for a proof of Proposition 3. To identify the violated strong 2-matching constraints, we use the heuristic of Padberg and Rinaldi (1990).

4 Benders Decomposition Scheme

The formulation 1-PDTSP-DIH includes a MCNFP as a subproblem, which allows us to apply Benders decomposition. To this end, we remove the flow variables z_{ij} along with the associated constraints, and we add an auxiliary variable to construct the master problem.

$$\text{(MP)} \quad \text{minimize} \sum_{(i,j) \in \bar{A}_1} t_{ij} x_{ij} + w \qquad (30)$$

$$\text{subject to } x(\delta^+(i)) = y_i \quad (i \in \bar{V}_1) \qquad (31)$$

$$x(\delta^-(i)) = y_i \quad (i \in \bar{V}_1) \qquad (32)$$

$$x(\delta^+(S)) \geq r(S) \quad (S \subset \bar{V}_1) \tag{33}$$
$$x(\delta^-(S)) \geq r(S) \quad (S \subset \bar{V}_1) \tag{34}$$
$$x_{ij} = 0 \text{ or } 1 \quad ((i,j) \in \bar{A}) \tag{35}$$
$$y_i = 0 \text{ or } 1 \quad (i \in \bar{V}_1 \setminus \bar{T}) \tag{36}$$
$$y_i = 1 \quad (i \in \bar{T}) \tag{37}$$
$$w \geq 0. \tag{38}$$

The slave problem is constructed by fixing the x_{ij} and y_i variables to an optimal solution of the MP, and is given below:

$$\text{(SP)} \quad \text{minimize} \sum_{(i,j)\in\bar{A}} \bar{c}_{ij} z_{ij} \tag{39}$$
$$\text{subject to } z(\delta^+(i)) - z(\delta^-(i)) = \bar{b}_i \quad (i \in \bar{V}), \tag{40}$$
$$z_{ij} \leq \alpha_{ij} x^*_{ij} \quad ((i,j) \in \bar{A}_1), \tag{41}$$
$$z_{ij} \geq l_i \quad ((i,j) \in \bar{A}_2) \tag{42}$$
$$z_{ij} \leq u_i \quad ((i,j) \in \bar{A}_2) \tag{43}$$
$$z_{ij} \leq Q y^*_i \quad ((i,j) \in \bar{A}_3) \tag{44}$$
$$z_{ij} \geq 0 \quad ((i,j) \in \bar{A}). \tag{45}$$

The next step is to construct the dual of the slave problem whose solution will yield valid inequalities. Let us associate the dual variable sets $v^1, ..., v^5$ with constraints sets (40) – (44) of the SP. The dual slave problem is then

$$\text{(DSP)} \quad \text{maximize} \sum_{i\in\bar{V}} \bar{b}_i v^1_i + \sum_{(i,j)\in\bar{A}_1} \alpha_{ij} x^*_{ij} v^2_{ij} + \sum_{(i,j)\in\bar{A}_2} (l_i v^3_{ij} + u_i v^4_{ij}) + \sum_{(i,j)\in\bar{A}_3} Q y^*_i v^5_{ij} \tag{46}$$
$$\text{subject to } v^1_i - v^1_j + v^2_{ij} \leq \bar{c}_{ij} \quad ((i,j) \in \bar{A}_1) \tag{47}$$
$$v^1_i - v^1_j + v^3_{ij} + v^4_{ij} \leq \bar{c}_{ij} \quad ((i,j) \in \bar{A}_2) \tag{48}$$
$$v^1_i - v^1_j + v^5_{ij} \leq \bar{c}_{ij} \quad ((i,j) \in \bar{A}_3) \tag{49}$$
$$v^1 \text{ unrestricted}, v^2, v^4, v^5 \leq 0, v^3 \geq 0. \tag{50}$$

If the optimal solution value of the DSP is strictly greater than that of the auxiliary variable w^*, we add an *optimality cut* of the following form to the MP, using the optimal solution v^* of the DSP:

$$w \geq \sum_{i\in\bar{V}} \bar{b}_i v_i^{1*} + \sum_{(i,j)\in\bar{A}_1} \alpha_{ij} x_{ij} {v_{ij}^2}^* + \sum_{(i,j)\in\bar{A}_2} (l_i {v_{ij}^3}^* + u_i {v_{ij}^4}^*) + \sum_{(i,j)\in\bar{A}_3} Q y_i {v_{ij}^5}^*. \quad (51)$$

Note that these inequalities are required only when $h > 0$. If the DSP is unbounded, we add a *feasibility cut* of the following form to the MP, using the unbounded ray v^*:

$$\sum_{i\in\bar{V}} \bar{b}_i v_i^{1*} + \sum_{(i,j)\in\bar{A}_1} \alpha_{ij} {v_{ij}^2}^* x_{ij} + \sum_{(i,j)\in\bar{A}_2} (l_i {v_{ij}^3}^* + u_i (v_{ij}^4)^*) + \sum_{(i,j)\in\bar{A}_3} Q {v_{ij}^5}^* y_i \leq 0. \quad (52)$$

The feasibility cuts can also be upgraded to yield the *rounded Benders inequalities*:

$$\sum_{(i,j)\in\bar{A}_1} \lfloor \alpha_{ij} {v_{ij}^2}^* \rfloor x_{ij} + \sum_{(i,j)\in\bar{A}_3} \lfloor Q {v_{ij}^5}^* \rfloor y_i \leq \lfloor -\sum_{i\in\bar{V}} \bar{b}_i v_i^{1*} - \sum_{(i,j)\in\bar{A}_2} (l_i {v_{ij}^3}^* - u_i {v_{ij}^4}^*) \rfloor. \quad (53)$$

5 A Unified Branch-and-Cut Algorithm

Our unified branch-and-cut algorithm, capable of handling both 1-PDTSP-DIH and MP, can be summarized as follows.

Step 1 (Root node). Construct a subproblem consisting of the initial formulation and insert this subproblem in a list.
Step 2 (Node selection). If the list is empty, stop. Else select and remove a subproblem from the list.
Step 3 (Subproblem solution). Solve the subproblem. If the objective function value is less than the best lower bound, go to Step 2.
Step 4 (Constraint generation). Identify violated members of the arc-vertex constraints, strong connectivity constraints, strong 2-matching constraints, clique cluster inequalities, and add them to the subproblem. For

MP, also solve DSP and identify violated feasibility cuts and optimality cuts. If at least one constraint is generated, go to Step 3.
Step 5 (Integrality check). If the solution is integer, update the best known solution, and go to Step 2.
Step 6 (Branching). Construct two subproblems by branching on a binary fractional variable. Add the subproblems to the list and go to Step 2.

To separate the connectivity constraints (13), (14), and the clique cluster inequalities (25), we have used the separation algorithms described in Hernández-Pérez and Salazar-González (2007). To identify the violated strong 2-matching inequalities, we have used the heuristic of Padberg and Rinaldi (1990).

6 Computational Results

We have implemented the branch-and-cut algorithm using C++ and the Callable Library of CPLEX 12.1 on the IRIDIS 3 computing cluster having Intel Nehalem nodes with two 4-core processors and 22 GB RAM. We have adapted a subset of the 1-PDTSP instances provided by Hernández-Pérez and Salazar-González (2007), which are available at `http://hhperez.webs.ull.es/PDsite/`. Note that these instances associate a negative quantity with a vertex to denote a demand, and a positive quantity to denote a supply. We have assumed all our vertices to have an identical lower bound p, and an identical upper bound q. Given the set of supply and demand values $b_i^{'}$, we have set $p = -\min_{i \in V}\{b_i^{'}\}$, and used the following rule to determine the current supply of vertex i:

$$b_i = \begin{cases} b_i^{'} + q & (b_i^{'} > 0) \\ p + b_i^{'} & (b_i^{'} < 0) \\ \lfloor (p+q)/2 \rfloor & (b_i^{'} = 0). \end{cases}$$

This choice of p ensures that $b_i \geq 0$, $d_i = -b_i^{'}$ if $b_i^{'} < 0$, and $e_i = b_i^{'}$ if $b_i^{'} > 0$, for all $i \in V$. Consequently, our parameter for experimentation becomes q, which is constrained to be at least equal to p. We have experimented with $q - p \in \{1, 2, 3, 4, 5\}$, $h = 0$ and $h \geq 0$. As h grows, the 1-PDTSP-DIH becomes increasingly similar to 1-PDTSP as extra handling is discouraged. We have observed that the CPU time requirement is almost identical for relatively small values of h and $h = 0$, and we only report results for the first case. We have adapted 240 instances, which yielded 1200 instances when combined with the range of $q - p$. Table 1 provides results for $|V| = 30$. Tables 2 and 3 present the results for $|V| = 40$ and $|V| = 50$, respectively. The column "Name" refers to ten instances that share it, e.g. row n30q10 corresponds to average results on 10 instances. The detailed results are available upon request from the first author.

We first present computational results corresponding to the branch-and-cut algorithm used in the solution of 1-PDTSP-DIH. When $|V| = 30$, all instances are solved optimally within six minutes. Table 1 shows that the problem becomes easier as $q-p$ increases. This is due to the fact that vertices with a deficit can absorb more units, vertices with an excess can supply more, and transshipment vertices have a larger degree of freedom in both directions. Table 1 also shows that solution quality also improves when $q - p$ becomes larger. This is reflected by the column "gain" which reports the decrease in the objective function with respect to the base case "$q - p = 0$". When $|V| = 40$, all instances are also solved optimally within 12 minutes. Table 2 reports the average CPU times in seconds. When $|V| = 50$, it is not always possible to solve the problem optimally within two hours of computing time, and our algorithm successfully solves 367 out of 400 instances. The hardest instances are those with low vehicle capacities, and the maximum observed optimality gap is 6.39%. In Table 3, we report the average gaps with respect to the best known lower bound generated by the branch-and-cut algorithm. It can be observed that the average gaps are less than 1% over all instances, and the average computing times are below 40 minutes.

Table 1: Computational results for 1-PDTSP-DIH, $|V| = 30$

			$\mathbf{q-p=1}$		$\mathbf{q-p=2}$		$\mathbf{q-p=3}$		$\mathbf{q-p=4}$		$\mathbf{q-p=5}$	
Name	n	Q	**CPU**	**Gain**	**CPU**	**Gain**	**CPU**	**Gain**	**CPU**	**Gain**	**CPU**	**Gain**
n30q10	30	10	8.49	6.22%	3.55	8.60%	16.24	10.18%	8.41	11.08%	35.86	11.93%
n30q15	30	15	8.45	4.76%	7.94	6.93%	4.85	7.91%	5.59	8.47%	6.21	9.22%
n30q20	30	20	4.31	2.22%	9.90	2.93%	4.19	3.62%	4.30	4.07%	2.42	4.54%
n30q25	30	25	2.51	0.99%	1.75	5.18%	2.18	6.10%	1.13	6.74%	0.89	7.38%
n30q30	30	30	3.65	0.93%	2.00	3.11%	2.67	3.65%	1.71	4.17%	1.12	4.56%
n30q35	30	35	1.00	1.09%	1.62	1.79%	1.66	2.05%	0.85	2.68%	1.73	3.00%
n30q40	30	40	1.45	0.64%	2.01	1.33%	1.73	1.58%	0.87	2.25%	1.60	2.39%
n30q45	30	45	1.45	0.37%	1.87	1.04%	1.19	1.21%	0.64	1.79%	0.91	1.89%
Average			3.91	2.15%	3.83	3.86%	4.34	4.54%	2.94	5.16%	6.34	5.61%

The results obtained by solving MP improve upon those of 1-PDTSP-DIH in general. CPU times for $|V| = 30$ presented in Table 4 show the superiority of MP over 1-PDTSP. The only exception is a single instance for $q - p = 5$,

Table 2: CPU times for 1-PDTSP-DIH, $|V| = 40$

Name	n	Q	$\mathbf{q-p=1}$	$\mathbf{q-p=2}$	$\mathbf{q-p=3}$	$\mathbf{q-p=4}$	$\mathbf{q-p=5}$
n40q10	40	10	120.07	207.16	21.67	35.29	2.73
n40q15	40	15	31.55	15.18	12.75	16.51	5.30
n40q20	40	20	17.65	5.95	4.25	5.21	3.10
n40q25	40	25	3.49	2.01	2.89	3.09	2.72
n40q30	40	30	7.10	1.40	2.57	2.68	2.11
n40q35	40	35	8.04	1.65	2.09	1.74	2.56
n40q40	40	40	3.15	1.46	1.59	1.78	1.58
n40q45	40	45	5.48	1.37	1.54	1.40	1.58
Average			24.57	29.52	6.17	8.46	2.71

Table 3: Computational results for 1-PDTSP-DIH, $|V| = 50$

			$\mathbf{q-p=1}$		$\mathbf{q-p=2}$		$\mathbf{q-p=3}$		$\mathbf{q-p=4}$		$\mathbf{q-p=5}$	
Name	**n**	**Q**	**Gap**	**CPU**	**Gap**	**CPU**	**Gap**	**CPU**	**Gap**	**CPU**	**Gap**	**CPU**
n50q10	50	10	1.4%	4726.9	0.4%	2805.4	0.5%	2399.7	0.5%	2332.7	0.0%	336.4
n50q15	50	15	0.9%	2674.1	0.2%	3399.2	0.2%	2907.1	0.1%	2446.7	0.3%	3120.9
n50q20	50	20	0.5%	1354.7	0.1%	1925.7	0.0%	1669.1	0.7%	2403.7	0.2%	1686.8
n50q25	50	25	0.0%	355.4	0.0%	364.6	0.0%	104.9	0.0%	186.8	0.0%	111.1
n50q30	50	30	0.0%	113.4	0.0%	98.6	0.0%	68.7	0.0%	80.8	0.0%	74.1
n50q35	50	35	0.0%	161.3	0.0%	37.0	0.0%	44.5	0.0%	34.9	0.0%	27.7
n50q40	50	40	0.0%	153.5	0.0%	41.4	0.0%	32.5	0.0%	15.4	0.0%	20.5
n50q45	50	45	0.0%	28.8	0.0%	25.6	0.0%	14.3	0.0%	13.6	0.0%	21.7
Average			0.35%	1196.03	0.10%	1087.19	0.09%	905.10	0.17%	939.34	0.06%	674.90

for which CPLEX struggled to find the integer optimal solution for about 40 minutes. The same situation is reflected in Table 5 for $|V| = 40$, where MP is solved 4.5 times faster than 1-PDTSP-DIH on average. The computational reach of both formulations seems to be $|V| = 50$, with MP being solved 1.5 times faster than 1-PDTSP on average, and 14 more instances being successfully solved to optimality within two hours of CPU time. Table 6 provides computational results for $|V| = 50$.

Table 4: CPU times for MP, $|V| = 30$

Name	n	Q	$\mathbf{q-p=1}$	$\mathbf{q-p=2}$	$\mathbf{q-p=3}$	$\mathbf{q-p=4}$	$\mathbf{q-p=5}$
n30q10	30	10	2.28	1.11	3.74	6.79	233.14
n30q15	30	15	4.13	2.77	1.39	3.99	5.85
n30q20	30	20	3.34	2.89	3.61	1.64	1.71
n30q25	30	25	0.79	1.01	0.78	0.58	0.35
n30q30	30	30	1.07	0.51	0.56	0.37	0.31
n30q35	30	35	0.62	0.76	0.44	0.57	0.39
n30q40	30	40	0.96	0.77	0.74	0.88	0.64
n30q45	30	45	0.88	0.83	0.71	0.44	0.45
Average			1.76	1.33	1.50	1.91	30.35

Table 5: CPU times for MP, $|V| = 40$

Name	n	Q	$q - p = 1$	$q - p = 2$	$q - p = 3$	$q - p = 4$	$q - p = 5$
n40q10	40	10	18.65	31.78	4.55	21.80	1.76
n40q15	40	15	3.55	6.07	3.50	4.10	1.47
n40q20	40	20	3.67	1.24	1.53	1.13	0.85
n40q25	40	25	1.74	0.61	0.83	1.10	0.72
n40q30	40	30	0.93	0.73	0.60	0.64	0.56
n40q35	40	35	0.72	0.55	0.58	0.66	0.66
n40q40	40	40	1.12	0.68	0.56	0.66	0.60
n40q45	40	45	0.38	0.40	0.40	0.38	0.43
Average			3.84	5.26	1.57	3.81	0.88

Table 6: Computational results for MP, $|V| = 50$

			$q - p = 1$		$q - p = 2$		$q - p = 3$		$q - p = 4$		$q - p = 5$	
Name	**n**	**Q**	**Gap**	**CPU**	**Gap**	**CPU**	**Gap**	**CPU**	**Gap**	**CPU**	**Gap**	**CPU**
n50q10	50	10	0.8%	3674.11	1.4%	3274.31	0.9%	2498.76	0.3%	2704.61	0.0%	139.50
n50q15	50	15	1.0%	1876.86	0.1%	1950.06	0.2%	1316.63	0.1%	1291.37	0.1%	1339.25
n50q20	50	20	0.0%	288.05	0.0%	245.04	0.0%	332.51	0.0%	826.50	0.0%	702.80
n50q25	50	25	0.0%	42.59	0.0%	38.94	0.0%	30.67	0.0%	27.01	0.0%	21.15
n50q30	50	30	0.0%	13.16	0.0%	19.84	0.0%	22.06	0.0%	13.07	0.0%	10.89
n50q35	50	35	0.0%	13.78	0.0%	9.18	0.0%	8.25	0.0%	5.79	0.0%	4.25
n50q40	50	40	0.0%	24.49	0.0%	12.32	0.0%	5.75	0.0%	4.89	0.0%	3.42
n50q45	50	45	0.0%	12.82	0.0%	4.99	0.0%	3.22	0.0%	3.60	0.0%	3.12
Average			0.22%	743.23	0.19%	694.34	0.14%	527.23	0.06%	609.60	0.01%	278.05

7 Conclusions

We have introduced, modeled, and solved the One-Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals, a generalization of the One-Commodity Pickup and Delivery Traveling Salesman Problem encountered namely in the repositioning of bicycles in public sharing systems. We have first studied the subproblem arising when the vehicle route is fixed, and we have showed it to be a minimum cost network flow problem, whether handling cost is zero or not. Based on the network flow formulation, we have developed two integer programming formulations. We have imported valid inequalities from the routing literature, and we have developed a branch-and-cut algorithm as well as a Benders decomposition scheme. We have described a unified branch-and-cut algorithm capable of solving both models. We have also presented computational results for instances adapted from 1-PDTSP. The branch-and-cut algorithm is capable of solving instances with up to 50 vertices by either formulation. However, the Benders based formulation and algorithm yield much shorter computation times and can solve more instances.

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New York.

Anily, S., R. Hassin. 1992. The swapping problem. *Networks* **22** 419–433.

Benchimol, M., P. Benchimol, B. Chappert, A.D.L. Taille, F. Laroche, F. Meunier, L. Robinet. 2012. Balancing the stations of a self service "bike hire" system. *RAIRO Operations Research.* Forthcoming.

Bordenave, C., M. Gendreau, G. Laporte. 2009. A branch-and-cut algorithm for the non-preemptive swapping problem. *Naval Research Logistics* **56** 478–486.

Bordenave, C., M. Gendreau, G. Laporte. 2010. Heuristics for the mixed swapping problem. *Computers & Operations Research* **37** 108–114.

Bordenave, C., M. Gendreau, G. Laporte. 2012. A branch-and-cut algorithm for the preemptive swapping problem. *Networks* **59** 387–399.

Chemla, D., F. Meunier, R. Wolfler Calvo. 2013. Bike sharing system: solving the static rebalancing problem. *Discrete Optimization* Forthcoming.

Erdoğan, G., J.-F. Cordeau, G. Laporte. 2010. A branch-and-cut algorithm for the non-preemptive capacitated swapping problem. *Discrete Applied Mathematics* **158** 1599–1614.

Gendreau, M., G. Laporte, F. Semet. 1997. The covering tour problem. *Operations Research* **45** 568–576.

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

Hernández-Pérez, H., J.-J. Salazar-González. 2007. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks* **50** 258–272.

Padberg, M.W., G. Rinaldi. 1990. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming* **47** 219–257.

Raviv, T., M. Tzur, I. A. Forma. 2013. Static repositioning in a bike-sharing system: Models and solution approaches. *EURO Journal on Transportation and Logistics* Forthcoming, published online 8 January 2013.