The Multi-commodity Pickup-and-Delivery Traveling Salesman Problem

Hipólito Hernández-Pérez and Juan-José Salazar-González
DEIOC, Facultad de Matemáticas, Universidad de La Laguna, 38271 La Laguna, Tenerife, Spain

The "multi-commodity Pickup-and-Delivery Traveling Salesman Problem" (*m*-PDTSP) is a generalization of the well-known "Traveling Salesman Problem" in which cities correspond to customers providing or requiring known amounts of *m* different products, and the vehicle has a known capacity. Each customer must be visited exactly once by the vehicle serving the demands of the different products while minimizing the total travel distance. It is assumed that a unit of a product collected from a customer can be supplied to any other customer that requires this product. We introduce a mixed integer linear programming model for the *m*-PDTSP, discuss a classical decomposition technique, describe valid inequalities to strengthen the linear programming relaxation of the model, and detail separation procedures to develop a branch-and-cut procedure. Computational experiments on randomly generated instances with up to 30 customers, three products, and small vehicle capacities are analyzed. © 2013 Wiley Periodicals, Inc. NETWORKS, Vol. 63(1), 46–59 2014

Keywords: traveling salesman; pickup-and-delivery; branch-and-cut; multi-commodity flow

1. INTRODUCTION

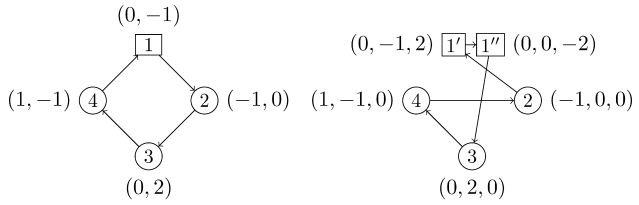
Many practical applications in transportation involve routing and delivery optimization problems. This article considers the following generalization of the traveling salesman problem (TSP). A finite set of cities is given and the travel cost from one city to another city is assumed to be known, and not necessarily symmetric. One specific city is considered to be a depot, whereas the other cities are identified as customers. Each customer requires some given quantities of different products and/or provides some given quantities of other different products. A unit of a product collected from a customer can be supplied to any customer that requires this product. It is assumed that the vehicle has a fixed capacity and must start and finish the route at the depot. The route must be a Hamiltonian tour through all the cities. Then, the multi-commodity pickup-and-delivery traveling salesman problem (m-PDTSP) is the problem of finding a route for the vehicle such that it picks up and delivers all the quantities of the different products satisfying the vehicle-capacity limitation and minimizing the total travel cost. Because each city is visited once, each unit of a product loaded on the vehicle stays on the vehicle until it is delivered to its destination. For that reason, we say that the m-PDTSP is a nonpreemptive problem.

Figure 1a shows an example of a feasible route for an instance involving four cities, two products and a vehicle with a capacity of two units. Customer 2 requires one unit of Product 1, Customer 3 delivers two units of Product 2, and Customer 4 delivers one unit of Product 1 and requires one unit of Product 2. Thus, the depot (represented by 1) can be seen as a customer (Customer 1) requiring one unit of Product 2.

The initial load of any product in the vehicle when leaving the depot is unknown, and must be determined within the optimization problem. In the route in Figure 1a, the vehicle leaves the depot with one unit of Product 1 to be delivered at Customer 2. This extra unit of Product 1 is returned by the vehicle to the depot at the end of the route.

Some variants of the m-PDTSP where the initial load of any product is fixed can also be solved through the approach described in this article with a slight modification of the instance. For example, if one wants to impose that the vehicle leaves the depot with zero load, then the modification consists of introducing a dummy product and replacing the depot by two customers: one customer delivers a demand of the dummy product equal to the vehicle capacity, and the other customer requires this demand of the dummy product. The travel cost between the dummy customers is defined by a low number so a min-cost route visits the pickup dummy customer immediately after the delivery one. Figure 1b illustrates the transformation of the 2-PDTSP instance in Figure 1a when the initial load of the vehicle leaving the depot is required to be zero. It has five cities, three products, and a vehicle capacity of two units. The depot in the restricted 2-PDTSP instance has been replaced by the dummy customers $1'$ and $1''$ in the 3-PDTSP. An optimal route for the (unrestricted) 3-PDTSP instance corresponds to an optimal route

Received November, 2011; accepted April, 2013
Correspondence to: J.-J. Salazar-González; e-mail: jjsalaza@ull.es
Contract grant sponsor: "Ministerio de Ciencia e Innovación," Spain (research project MTM2009-14039-C06-01).
DOI 10.1002/net.21521
Published online 1 October 2013 in Wiley Online Library (wileyonlinelibrary.com).



(a) unfixed initial load (b) zero initial load

FIG. 1. m-PDTSP routes.

for the restricted 2-PDTSP instance. The transformation is also possible without the dummy product, as proposed in [19] for $m = 1$.

An application of the m-PDTSP occurs in the context of inventory repositioning. Assume that a set of retailers are geographically dispersed in a region. Often, due to the random nature of the demands, some retailers have an excess of inventory of some products, whereas others have such strong sales that they need additional stock. In many cases, the firm may decide to transfer inventory from retailers that have experienced below average sales to those that have experienced above average sales. Determining the cheapest way to execute a given stock transfer (with the requirement that each localization has to be visited exactly once) is the m-PDTSP. Anily and Bramel [1] give applications for a problem related to the m-PDTSP, and these applications are also valid for the m-PDTSP.

Another application arises in the context of a self-service bike hiring system, where every night a capacitated vehicle must visit the bike stops in a city to collect or deliver bikes to restore the initial configuration of the system. Chemla et al. [11] and Raviv et al. [24], among others, approached the case where the bikes are all identical as a problem related to the 1-PDTSP. When there are different types of bikes (for example, with and without baby chairs), the problem is related to the m-PDTSP.

A particular case of the 2-PDTSP is the traveling salesman problem with pickups and deliveries (TSPPD). In the TSPPD, the customers are divided in two types, pickup customers and delivery customers, each one with a given demand of a product. There is a vehicle with a given capacity originally stationed in the depot. Travel costs are known. The total amount of product collected from pickup customers must be delivered only to the depot, and the product collected from the depot must be moved to the delivery customers. For example, this is the case when empty bottles must be collected from customers and taken to a warehouse, and full bottles must be delivered from the warehouse to the customers. Mosheiov [23] introduces the TSPPD and proposes applications and heuristic approaches. Anily and Mosheiov [2] present approximation algorithms for the TSPPD, here renamed TSP with pickups and backhauls, and Gendreau et al. [13] propose several heuristics tested on instances with up to 200 customers. Baldacci et al. [5] deal with the same problem, here named TSP with delivery and collection constraints, and present an exact algorithm based on a two-commodity network flow formulation which was able to prove the optimality of some TSPPD instances with 150 customers. Hernández-Pérez and Salazar-González [18] present a heuristic algorithm solving instances with up to 500 customers based on a transformation procedure of a TSPPD instance into a 1-PDTSP instance, and in [19] they present a branch-and-cut algorithm for the exact solution of TSPPD instances with up to 200 customers.

The literature have also addressed many other related problems. We now mention some articles dealing with one-commodity variants. Chalasani and Motwani [10] study the special case of the 1-PDTSP where the delivery and pickup quantities are all equal to one unit. This problem is called Q-delivery TSP where Q is the capacity of the vehicle. Anily and Bramel [1] consider the same problem with the name Capacitated TSP with pickups and deliveries. Chalasani and Motwani [10] propose a 9.5-approximation algorithm. Anily and Bramel [1] propose an algorithm with a better worst-case ratio. For the 1-PDTSP, Hernández-Pérez and Salazar-González [19] present an exact algorithm solving instances with up to 200 customers. Hernández-Pérez et al. [16] describe an hybrid algorithm that combines greedy randomized adaptive search procedure and variable neighborhood descent paradigms. Zhao et al. [27] propose a genetic algorithm that on average gives better results. Finally, Hanafi and coworkers [15] describe a general variable neighborhood search improving the best-known solution for all benchmark instances and solving instances with up to 1000 customers.

The one-to-one m-PDTSP is a particular case of the m-PDTSP where each commodity has one origin and one destination. It can be considered as a *DARP* without time windows requirements. In that sense, in the one-to-one m-PDTSP one assumes that the initial load of the vehicle when leaving the depot is zero unless the depot is the source of a commodity. Hernández-Pérez and Salazar-González [20] describe a branch-and-cut algorithm for this problem solving instances involving up to 24 customers and 15 commodities. Rodríguez-Martín and Salazar-González [25] propose and compare several metaheuristic approaches to solve instances with up to 300 customers and 600 commodities.

The TSP with Precedence Constraints (TSPPC), also known as the sequential ordering problem, is a particular case of the one-to-one m-PDTSP where there is no vehicle capacity. Bianco et al. [8] describe a dynamic programming algorithm for TSPPC. Balas et al. [3], Ascheuer et al. [4], and Gouveia and Pesneau [14] have proposed exact approaches based on integer programming formulations.

A problem closely related to the m-PDTSP which includes many sources and many destinations with various commodities is the nonpreemptive capacitated swapping problem (NCSP) proposed by Erdoğan et al. [12]. In the NCSP, there is one depot and a set of customers. Each customer may supply one item of a commodity and/or demand one item of another commodity. Every item of each commodity has a weight. When a customer supplies and demands one item of the same commodity, the customer is called a transshipment customer. The items cannot be split and they cannot be dropped off in an

intermediate customer. However, a transshipment customer can be used as an intermediate customer for the commodity supplied and demanded by this customer. The problem consists of finding a minimum cost route to satisfy all customers. Differences in this problem with respect to the m-PDTSP are: (1) a feasible solution may not be a Hamiltonian circuit; (2) each customer can only supply and/or demand one unit of one product; (3) there exist transshipment customers; (4) each commodity has a weight. Erdoğan et al. [12] describe a branch-cut-algorithm to solve the NCSP. Bordenave et al. [9] describe a branch-and-cut algorithm to solve the particular case of the NCSP where the vehicle capacity is equal to one unit and all the item weights are also one unit.

The literature addresses many other routing problems to pickup and deliver products. Savelsbergh and Sol [26] present a survey that summarizes the results up to 1995. More recently, Berbeglia et al. [6] and Berbeglia et al. [7] give a classification of the static and dynamic versions of pickup-and-delivery problems, respectively.

This article is structured as follows. Section 2 describes a mathematical formulation and a decomposition method for the m-PDTSP. Section 3 shows inequalities to strengthen the formulation. Section 4 proposes separation procedures for the new inequalities. Finally, section 5 discusses different strategies to solve the problem and analyzes computational results.

2. MATHEMATICAL MODEL

This section provides a mathematical model for the m-PDTSP on a complete directed graph $G = (V, A)$. The node set $V = \{1, \ldots, n\}$ represents the customers, included the depot which is denoted by 1. For each pair of customers i and j, we have the arc $a = (i, j) \in A$ and a travel cost c_{ij}. Let $K = \{1, \ldots, m\}$ be the set of products. For each customer, $i \in V$ and each product $k \in K$ let q_i^k be the demand of product k associated with customer i. When $q_i^k > 0$ customer i offers q_i^k units of product k and when $q_i^k < 0$ customer i requires $-q_i^k$ units of product k. We assume that $\sum_{i \in V} q_i^k = 0$ for all $k \in K$, that is, each product is conserved through the route. The capacity of the vehicle is denoted by Q.

To provide a mathematical model for the m-PDTSP, we introduce a 0-1 variable for each arc $a \in A$:

$$x_a := \begin{cases} 1 & \text{if and only if } a \text{ is routed by the vehicle,} \\ 0 & \text{otherwise,} \end{cases}$$

and a continuous variable for each arc $a \in A$ and for each product $k \in K$:

$f_a^k :=$ load of product k in the vehicle when going through arc a.

Given $S', S'' \subset V$ such that $S' \cap S'' = \emptyset$, we denote by $A(S' : S'')$ the set of arcs $\{(i, j) : i \in S' \text{ and } j \in S''\}$. For a subset $S \subset V$ we use $\delta^+(S)$ instead of $A(S : V \backslash S)$ and $\delta^-(S)$ instead of $A(V \backslash S : S)$. Finally, for a subset $A' \subseteq A$, we write $x(A')$ instead of $\sum_{a \in A'} x_a$.

Then, the m-PDTSP can be formulated as:

$$\min \sum_{a \in A} c_a x_a \tag{1}$$

subject to

$$x(\delta^-(\{i\})) = 1 \qquad \text{for all } i \in V \tag{2}$$

$$x(\delta^+(\{i\})) = 1 \qquad \text{for all } i \in V \tag{3}$$

$$x(\delta^+(S)) \geq 1 \qquad \text{for all } S \subset V \tag{4}$$

$$x_a \in \{0, 1\} \qquad \text{for all } a \in A \tag{5}$$

$$\sum_{a \in \delta^+(\{i\})} f_a^k - \sum_{a \in \delta^-(\{i\})} f_a^k = q_i^k \qquad \text{for all } i \in V \text{ and } k \in K \tag{6}$$

$$f_a^k \geq 0 \qquad \text{for all } a \in A \text{ and } k \in K \tag{7}$$

$$\sum_{k \in K} f_a^k \leq Q x_a \qquad \text{for all } a \in A. \tag{8}$$

Model (1)–(5) captures the TSP aspect of the m-PDTSP. Constraints (6) impose the load conservation for each commodity, and constraints (7) and (8) impose that the vehicle load must be nonnegative and less than or equal to Q. This model does not fix the initial load of the vehicle when leaving the depot. This load is determined a-posteriori by $\sum_{k \in K} \sum_{a \in \delta^+(\{1\})} f_a^k$.

The x_a variables of (1)–(8) represent a Hamiltonian circuit in G, but not all Hamiltonian circuits in G may define a feasible m-PDTSP solutions. The reason is the vehicle capacity. When Q is large enough (for example larger than the sum of all the pickup demands), the m-PDTSP is the TSP. However, this is not the case when the demand of a commodity by a customer (e.g., the depot) is Q. This situation forces the vehicle to enter or leave this customer (depot) with zero load, and therefore it imposes precedence constraints between the other customers. For example, a customer i that requires 5 units of a commodity k (i.e., $q_i^k = -5$) must be visited after a set of customers offering at least 5 units of the commodity k. Depending on these precedence constraints, model (1)–(8) could be infeasible in this situation. Indeed, as mentioned in the introduction, the m-PDTSP can also be used to solve a capacitated version of the TSPPC.

3. STRENGTHENING THE *M*-PDTSP MODEL

A better model than (1)–(8) can be given by tightening the bounds on the continuous variables, projecting out these variables, and introducing new valid inequalities exploiting the nature of the m-PDTSP solutions.

3.1. Strengthening the Bounds on the Continuous Variables

The amount of units of commodity k in the vehicle when going from customer i to customer j (i.e., f_{ij}^k) must satisfy $0 \leq f_{ij}^k \leq Q$ if the arc (i,j) is routed; but it must also satisfy $q_i^k \leq f_{ij}^k$ (the vehicle must transport the load picked up from customer i) and $-q_j^k \leq f_{ij}^k$ (the vehicle must transport the load delivered to customer j). Hence, $f_{ij}^k \geq \max\{q_i^k, -q_j^k, 0\}$ if $x_{ij} = 1$. Reciprocally, it must satisfy $f_{ij}^k \leq Q + \min\{q_i^k, -q_j^k, 0\}$ if $x_{ij} = 1$. On the other hand, the flow of all commodities through arc (i,j) must satisfy $\sum_{k\in K} f_{ij}^k \leq Q + \min\{0, \sum_{k\in K} q_i^k, -\sum_{k\in K} q_j^k\}$ (the vehicle must have enough free space to pick up the load from customer i and j). Therefore, inequalities (7) and (8) are strengthened by

$$\max\{q_i^k, -q_j^k, 0\}\, x_{ij} \leq f_{ij}^k \leq \left(Q + \min\{q_i^k, -q_j^k, 0\}\right) x_{ij} \quad \text{for all } (i,j) \in A \text{ and } k \in K \tag{9}$$

and

$$\sum_{k\in K} f_{ij}^k \leq \left(Q + \min\left\{\sum_{k\in K} q_i^k, -\sum_{k\in K} q_j^k, 0\right\}\right) x_{ij} \quad \text{for all } (i,j) \in A. \tag{10}$$

As a consequence, variable x_{ij} is fixed to zero when

$$\sum_{k\in K} \max\{q_i^k, -q_j^k, 0\} > Q + \min\left\{\sum_{k\in K} q_i^k, -\sum_{k\in K} q_j^k, 0\right\}. \tag{11}$$

For example, if $m = 2$, $Q = 1$, $q_i^1 = 1$, $q_i^2 = q_j^1 = 0$ and $q_j^2 = -1$ then the arc (i,j) cannot be routed. However, the reverse arc (j,i) can be routed.

3.2. Benders' Inequalities

The integer mixed linear programming model (1)–(6) and (9)–(10) motivates a method to solve the m-PDTSP. However, the main drawback of this model is the large number of continuous variables, which are only necessary to check the feasibility of a vector x.

Let us now give an alternative model based on projecting out the continuous variables by applying a Benders' decomposition technique. Observe that the equations in (6) can be relaxed to inequalities without adding new solutions. Then, according to Farkas' Lemma, the polytope described by (6)–(8) for a fixed vector x is feasible if and only if all extreme rays of the cone

$$\alpha_i^k - \alpha_j^k \leq \beta_{(i,j)} \quad \text{for all } (i,j) \in A \text{ and } k \in K \tag{12}$$

$$\alpha_i^k \geq 0 \quad \text{for all } i \in V \text{ and } k \in K \tag{13}$$

$$\beta_a \geq 0 \quad \text{for all } a \in A \tag{14}$$

satisfy

$$\sum_{a\in A} Q\beta_a x_a \geq \sum_{k\in K}\sum_{i\in V} \alpha_i q_i^k. \tag{15}$$

Inequalities (15) are called Benders' inequalities and they can be strengthened if the decomposition is applied to the linear system with (9)–(10) instead of (7)–(8). However, preliminary computational results show that the tighter system needs more time to be solved while the linear programming relaxations are no better with the replacement. Also with the replacement, we loose the possibility of rounding up the right-hand side coefficients in some cases and the notation becomes more complex. For that reason, we use (7)–(8) in this section.

When $m = 1$, Hoffman [21] gives a simple characterization of all the extreme rays of cone (12)–(14). Unfortunately, a similar result is unknown when $m > 1$. Motivated by Mirchandani [22], the following theorem describes some extreme rays with 0-1 coefficients of the cone (12)–(14).

Theorem 3.1.

i. For each $a' \in A$, let (α, β) be the vector defined by $\alpha_i^k = 0$ for all $i \in V$ and for all $k \in K$, $\beta_a = 0$ for all $a \in A\backslash\{a'\}$ and $\beta_{a'} = 1$. This vector is an extreme ray of cone (12)–(14).

ii. For each $k' \in K$, let (α, β) be the vector defined by $\alpha_i^{k'} = 1$ for all $i \in V$, $\alpha_i^k = 0$ for all $i \in V$ and $k \neq k'$ and $\beta_a = 0$ for all $a \in A$. This vector is an extreme ray of cone (12)–(14).

iii. For each sorted collection of subsets $\mathcal{S} = (S^1, \ldots, S^m)$, $S^k \subset V$ for all $k \in K$ (not all empty subsets), let (α, β) be the vector defined by $\alpha_i^k = 1$ for all $i \in S^k$ and $k \in K$, $\alpha_i^k = 0$ for all $i \in V\backslash S^k$ and $k \in K$, $\beta_a = 1$ for all $a \in \bigcup_{k\in K} \delta^+(S^k)$ and $\beta_a = 0$ for all $a \in A\backslash\left(\bigcup_{k\in K} \delta^+(S^k)\right)$. This vector is a ray of the cone (12)–(14).

iv. All the 0-1 extreme rays of the cone (12)–(14) are the ones described in (i), (ii) and (iii).

Proof. Obviously, (i) holds.

Proof of (ii). Let (α, β) be a vector defined as indicated in (ii). It satisfies the conditions of cone (12)–(14). Thus, it is a ray. Let us suppose that there are vectors $(\widehat{\alpha}, \widehat{\beta})$ and $(\bar{\alpha}, \bar{\beta})$ such that $(\alpha, \beta) = (\widehat{\alpha}, \widehat{\beta}) + (\bar{\alpha}, \bar{\beta})$. As all variables are greater or equal than 0 (Equations (13) and (14)), $\widehat{\alpha}_i^k = \bar{\alpha}_i^k = 0$ for all $k \neq k'$ and $i \in V$, and $\widehat{\beta}_a = \bar{\beta}_a = 0$ for all $a \in A$. If there is a $i' \in V$ with $\widehat{\alpha}_{i'}^{k'} = l > 0$, by Equation (12) we get that $\widehat{\alpha}_i^{k'} = l$ for all $i \in V$, and then $(\alpha, \beta) = l(\widehat{\alpha}, \widehat{\beta}) + (1-l)(\bar{\alpha}, \bar{\beta})$. Otherwise, if there is no a customer $i' \in V$ with $\widehat{\alpha}_{i'}^{k'} = l > 0$, then $(\widehat{\alpha}, \widehat{\beta})$ is vector 0.

Proof of (iii). Vector (α, β) defined in (iii) satisfies (12)–(14), because for each k, if $i \in S^k$ and $j \in V\backslash S^k$, then $\alpha_i^k - \alpha_j^k = 1$ and $\beta_{(i,j)} = 1$ (since $(i,j) \in \delta^+(S^k)$); otherwise $\alpha_i^k - \alpha_j^k \leq 0$.

Proof of (iv). Let vector (α, β) be such that $\alpha_i^k \in \{0,1\}$ for all $k \in K$ and for all $i \in V$ and $\beta_a \in \{0,1\}$ for all $a \in A$. If $\alpha_i^k = 0$ for all $k \in K$ and for all $i \in V$, then (α, β) is as (i) or can be decomposed by two or more vectors of (i). If $\beta_a = 0$

for all $a \in A$, there exists a $k' \in K$ and $i' \in V$ such that $\alpha_{i'}^{k'} = 1$. Then, $\alpha_i^{k'} = \alpha_{i'}^{k'} = 1$ for all $i \in V$ by equation (12) and (α, β) is as (ii) or can be decomposed by two or more vectors of (ii). Finally, let (α, β) such that there are $k' \in K$, $i' \in V$ and $a' \in A$ with $\alpha_{i'}^{k'} = 1$ and $\beta_{a'} = 1$. We define the sets $S^k = \{i \in V : \alpha_i^k = 1\}$ for $k \in K$. If there is a k' such that $S^{k'} = V$, then the vector $(\widehat{\alpha}, \widehat{\beta})$ with $\widehat{\alpha}_i^k = \alpha_i^k$ for all $k \neq k'$ and all $i \in V$, $\widehat{\alpha}_i^{k'} = 0$ for all $i \in V$ and $\widehat{\beta}_a = \beta_a$ is a ray and (α, β) can be decomposed by $(\widehat{\alpha}, \widehat{\beta})$ and a ray of form (ii). Thus, by Equation (12), $\beta_a = 1$ for all $a \in \bigcup_{k\in K} \delta^+(S^k)$. If there is $a' \in A \setminus \left(\bigcup_{k\in K} \delta^+(S^k)\right)$ with $\beta_{a'} = 1$, then the ray can be decomposed by a ray of form (iii) and one or more rays of form (i). Therefore, $\beta_a = 0$ for all $a \in A \setminus \bigcup_{k\in K} \delta^+(S^k)$ and the (α, β) is of form (iii). ∎

Extreme rays of cone (12)–(14) with no 0-1 coefficients exist. An example for $n = 4$ and $m = 2$ is the vector (α, β) defined by $\alpha_1^1 = \beta_{(1,2)} = 2$, $\alpha_3^1 = \alpha_4^1 = \alpha_1^2 = \alpha_3^2 = \beta_{(1,3)} = \beta_{(1,4)} = \beta_{(3,2)} = \beta_{(3,4)} = \beta_{(4,2)} = 1$, and $\alpha_2^1 = \alpha_2^2 = \alpha_4^2 = \beta_{(2,1)} = \beta_{(2,3)} = \beta_{(2,4)} = \beta_{(3,1)} = \beta_{(4,1)} = \beta_{(4,3)} = 0$. This example was found using the software PORTA.

Theorem 3.1 characterizes some inequalities in (15). In particular, each collection of customer subsets $\mathcal{S} = (S^1, \ldots, S^m)$ defines the following valid inequality for m-PDTSP:

$$x\left(\cup_{k\in K}\delta^+(S^k)\right) \geq \frac{1}{Q}\sum_{k\in K}\sum_{i\in S^k} q_i^k. \tag{16}$$

Observe that, by considering the collection $(V\setminus S^1, \ldots, V\setminus S^m)$ and because $x(\delta^+(S)) = x(\delta^-(S))$ for any customer set S, the family of inequalities (16) also contains:

$$x\left(\cup_{k\in K}\delta^+(S^k)\right) \geq -\frac{1}{Q}\sum_{k\in K}\sum_{i\in S^k} q_i^k.$$

Therefore, inequalities (16), with the right-hand side replaced by its absolute value and rounded up to the next integer value, give the following valid inequalities for m-PDTSP:

$$x\left(\cup_{k\in K}\delta^+(S^k)\right) \geq \left\lceil \frac{1}{Q}\left|\sum_{k\in K}\sum_{i\in S^k} q_i^k\right| \right\rceil. \tag{17}$$

Using the degree Equations (2) and (3), these inequalities can be rewritten as follows:

$$x\left(A(\cup_{k\in K}S^k)\setminus \cup_{k\in K}\delta^+(S^k)\right) \leq \left|\cup_{k\in K}S^k\right| - \left\lceil \frac{1}{Q}\left|\sum_{k\in K}\sum_{i\in S^k} q_i^k\right| \right\rceil. \tag{18}$$

3.3. Infeasible Paths

We describe a sufficient condition to detect whether a path cannot be part of a feasible solution. This issue is relevant because infeasible paths induce valid inequalities.

Extending the notation introduced in [18] for the 1-PDTSP, let us consider a path $\overrightarrow{P}$ defined by the customer sequence $v_1, \ldots, v_s$ for $s \leq n$. For simplicity of notation, we also use $\overrightarrow{P}$ to denote the set of arcs $\{(v_1, v_2), \ldots, (v_{s-1}, v_s)\}$. Let $l_i^k(\overrightarrow{P}) := l_{i-1}^k(\overrightarrow{P}) + q_{v_i}^k$ be the load of the vehicle when coming out from v_i if the vehicle follows this path and enters customer v_1 with load $l_0^k(\overrightarrow{P})$. Notice that $l_i^k(\overrightarrow{P})$ could be negative if $l_0^k(\overrightarrow{P}) = 0$ and, therefore, the minimum quantity of load of commodity k for a feasible solution through the path $\overrightarrow{P}$ is $-\min_{i=0}^s\{l_i^k(\overrightarrow{P})\}$. With this notation, a path $\overrightarrow{P}$ is infeasible if

$$\max_{i=0}^{s}\left\{\sum_{k\in K} l_i^k(\overrightarrow{P})\right\} - \sum_{k\in K}\min_{i=0}^{s}\left\{l_i^k(\overrightarrow{P})\right\} > Q. \tag{19}$$

Note that the left hand side of the inequality does not depend on the initial loads $l_0^k(\overrightarrow{P})$ for $k = 1, \ldots, m$. For example, if $l_0^1(\overrightarrow{P})$ increases in one, then $\max_{i=0}^s\left\{\sum_{k\in K} l_i^k(\overrightarrow{P})\right\}$ and $\sum_{k\in K}\min_{i=0}^s\left\{l_i^k(\overrightarrow{P})\right\}$ increase one also, but the difference remains the same.

A Hamiltonian tour x^* is infeasible for the m-PDTSP if it contains an infeasible path $\overrightarrow{P}$. The literature provides a simple valid inequality which is violated by x^* and which is the following:

$$\sum_{a\in\overrightarrow{P}} x_a \leq s - 2, \tag{20}$$

We next observe that for the m-PDTSP, there is a stronger inequality of type (17).

Let i_0 be the index where $\sum_{k\in K} l_i^k(\overrightarrow{P})$ is maximum over i, and let $i_1, \ldots, i_m$ be the indices where $l_i^k(\overrightarrow{P})$ are minimum for each $k \in K$, respectively. As x^* is a Hamiltonian tour, we may assume that $i_0 = s$, i.e., $\sum_{k\in K} l_i^k(\overrightarrow{P})$ is maximum in the last customer of the path. Let $S^k = \{v_{i_k+1}, \ldots, v_s\}$ for all $k \in K$. Observe that

$$\sum_{i\in S^k} q_i^k = l_s^k(\overrightarrow{P}) - \sum_{j=1}^{i_k} q_{v_j}^k = l_s^k(\overrightarrow{P}) - l_{i_k}^k(\overrightarrow{P}).$$

Therefore,

$$\left\lceil \frac{1}{Q}\sum_{k\in K}\sum_{i\in S^k} q_i^k \right\rceil = \left\lceil \frac{1}{Q}\left(\sum_{k\in K} l_s^k(\overrightarrow{P}) - \sum_{k\in K} l_{i_k}^k(\overrightarrow{P})\right) \right\rceil \geq 2$$

and

$$x^*\left(\cup_{k\in K}\delta^+(S^k)\right) = 1.$$

As a consequence, x^* violates, the inequality (17) defined by the collection $\mathcal{S} = (S^1, \ldots, S^m)$.

For example, consider a m-PDTSP instance with $m = Q = 3$ and Customers 2, 3, 4, and 5 with demand vectors $(-1, -1, +1)$, $(+1, +1, 0)$, $(-1, 0, 0)$, and $(+1, +1, 0)$,

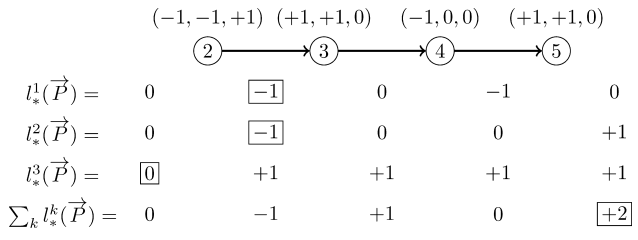


FIG. 2. Infeasible path when $m = 3$ and $Q = 3$.

respectively. The path defined by the customer sequence 2, 3, 4, and 5 is infeasible because

$$\max_{i=0}^{s}\left\{\sum_{k=1}^{m} l_i^k(\overrightarrow{P})\right\} - \sum_{k=1}^{m}\min_{i=0}^{s}\left\{l_i^k(\overrightarrow{P})\right\} = 4 > Q.$$

Moreover, $i_0 = 4, i_1 = i_2 = 1$, and $i_3 = 0$ are the indices where the maximum and the minima are achieved. Then, subsets $S^1 = S^2 = \{3, 4, 5\}$ and $S^3 = \{2, 3, 4, 5\}$ define a violated inequality (17): Figure 2 illustrates the calculations done in this example. This example shows a route violating the inequality (20) defined by the path 2, 3, 4, 5, that is,:

$$x_{23} + x_{34} + x_{45} \le 2,$$

This route also violates the stronger inequality (17) defined by subsets $S^1 = S^2 = \{3, 4, 5\}$ and $S^3 = \{2, 3, 4, 5\}$, which can be rewritten as in (18), that is:

$$x_{23} + x_{24} + x_{25} + x_{34} + x_{35} + x_{43} + x_{45} + x_{53} + x_{54} \le 2.$$

3.4. Capacity Constraints

An instance of the m-PDTSP can be relaxed to be an instance of the 1-PDTSP and this relaxation can be done in several ways. Given a nonempty subset of products $K' \subseteq K$, let $q_i = \sum_{k\in K'} q_i^k$. All valid inequalities for the 1-PDTSP instance where q_i is the demand of customer i are valid inequalities for the original m-PDTSP instance. The capacity inequalities for the 1-PDTSP are already in (17). For example, this occurs for a subset $S \subset V$ with $K' = \{k \in K : \sum_{i\in S} q_i^k > 0\}$, $S^k = S$ for $k \in K'$ and $S^k = \emptyset$ for $k \in K\backslash K'$, which is the following capacity constraint:

$$x(\delta^+(S)) \ge \left\lceil \frac{1}{Q}\sum_{k\in K'}\sum_{i\in S} q_i^k \right\rceil. \tag{21}$$

3.5. Clique Cluster Inequalities

The clique cluster inequalities of the 1-PDTSP (see [19]) can be adapted for the m-PDTSP. Let us consider a customer $v \in V$ and a collection of customer subsets $W_1, \ldots, W_s$ such that:

$$W_{j_1} \cap W_{j_2} = \{v\} \quad \text{for all } 1 \le j_1 < j_2 \le s,$$

$$\left|\sum_{k\in K', i\in W_j} q_i^k\right| \le Q \quad \text{for all } j \in \{1, \ldots, s\}$$

and for all $K' \subseteq K$,

$$\left|\sum_{k\in K', i\in W_{j_1}\cup W_{j_2}} q_i^k\right| > Q \quad \text{for all } 1 \le j_1 < j_2 \le s$$

and some $K' \subseteq K$.

Because at most one subset W_i can satisfy $x(\delta^+(W_i)) = 1$, then a valid inequality for m-PDTSP is:

$$\sum_{i=1}^{s} x(\delta^+(W_i)) \ge 2s - 1, \tag{22}$$

which is called clique cluster inequality.

3.6. Multistar Inequalities

Also the multistar inequalities of the 1-PDTSP (see [19]) are valid inequalities for the m-PDTSP by simply aggregating the commodities of a subset K' into one commodity. We now summarize two types of multistar inequalities.

Let $K' \subseteq K, N \subset V, \{S_1, \ldots, S_s\}$ be a collection of disjoint subsets in $V\backslash N$ such that $\sum_{i\in S_1, k\in K'} q_i^k, \ldots, \sum_{i\in S_s, k\in K'} q_i^k$ have all the same sign. Then, a generalized inhomogeneous multistar inequality is:

$$x(A(N)) \le |N| - \left| \frac{\sum_{i\in N, k\in K'} q_i^k}{Q} + \sum_{j=1}^{s} \frac{\sum_{i\in S_j, k\in K'} q_i^k}{Q} \times \left(x\left(A\left(N : S_j\right)\right) + x\left(A\left(S_j : N\right) - \left|S_j\right| + 1 + x\left(A\left(S_j\right)\right)\right)\right)\right|. \tag{23}$$

Let us consider the subsets $N \subset V, C \subseteq N$ and the collection of disjoint subsets $\{S_1, \ldots, S_s\}$ in $V\backslash N$. The subset N is called nucleus, C connector, and $\{S_1, \ldots, S_s\}$ set of satellites. Then, the inequality

$$\lambda x(A(N)) + \sum_{i=1}^{s}\left(x(A(C : S_i)) + x(A(S_i : C)) + x(A(S_i))\right) \le \mu + \sum_{i=1}^{s}(|S_i| - 1). \tag{24}$$

is valid for appropriate values of λ and μ. These constraints are called generalized homogeneous multistar inequality when $C = N$ and generalized homogeneous partial multistar inequality when $C \ne N$.

4. SEPARATION PROCEDURES

Given a solution x^* of a linear relaxation of a model and a family of constraints $\mathcal{F}$ valid for the integer solutions, a separation problem for $\mathcal{F}$ and x^* is the problem of proving that all constraints in $\mathcal{F}$ are satisfied by x^*, or finding a constraint in $\mathcal{F}$ violated by x^*. The scope of this section is to develop procedures to solve the separation problems associated with the classes of constraints introduced in the previous

section. The separation problem of constraints (4) is known, and described in section 4.1. The separation problems of the other constraints are more difficult, hence we describe mainly heuristic procedures. These procedures are applicable to the model (1)–(8), the stronger model (1)–(6) and (9)–(10), and the alternative model (1)–(5) and (15).

4.1. Separating Subtour Elimination Constraints (4)

Let $G^* := (V^*, A^*)$ be a network where $V^* := V$ is the vertex set, $A^* = \{a \in A : x_a^* > 0\}$ is the arc set, and x_a^* is the capacity of each $a \in A^*$. First, we check the inequalities $x_{ij} + x_{ji} \leq 1$ for violation. After, we check whether the graph G^* is connected or not. If it is not connected, then each connected component induces a violated subtour elimination constraint (4). Otherwise, we compute the cut with minimum capacity in G^* to possibly detect a violated subtour elimination constraint (4). Whenever a subset S defining a violated constraint (4) is found, we add the capacity constraint (21) associated with S. This approach is an exact separation procedure for (4) and a heuristic separation procedure for (21).

4.2. Separating Capacity Constraints (21)

A heuristic algorithm for capacity constraints (21) consists in building a list of candidates to be a set S defining a violated constraint. Given a (fractional) solution x^*, the list is initialized with n subsets of cardinality two, $S = \{i, j\}$, with the larger value $x_{ij}^* + x_{ji}^*$. Each set of cardinality $s \geq 3$ is generated from a set S of cardinality $s - 1$ in the list by inserting a new customer l verifying that $x^*(A(S : \{l\})) + x^*(A(\{l\} : S)) > 0$ and $S \cup \{l\}$ is not already in the list. If there are several customers j verifying these conditions, then the one which maximizes $x^*(A(S : \{l\})) + x^*(A(\{l\} : S))$ is chosen.

Another heuristic algorithm consists of selecting a subset K' of commodities and defining the 1-PDTSP instance with customer demands $\sum_{k \in K'} q_i^k$. Then, we use the algorithm in [17] to possibly find a set S defining a violated capacity constraint.

4.3. Separating Constraints (17)

We implement two heuristic procedures to separate the inequalities (17).

The first heuristic algorithm builds a Hamiltonian circuit $\overrightarrow{T}$ by inserting arcs with the largest value x_a^*. Ties are broken by considering the smallest value c_a. The feasibility of this circuit is checked in linear time by computing $\max_{i=0}^{n}\{\sum_{k \in K} l_i^k(\overrightarrow{T})\}$ and the $\min_{i=0}^{n}\{l_i^k(\overrightarrow{T})\}$ for each k. When an inequality of type (19) is satisfied, then we have found an infeasible path. This infeasible path induces a violated inequality (17). With the same approach, we also check the feasibility of the Hamiltonian circuit in the reverse direction.

This algorithm is a primal heuristic method because it provides an upper bound when the circuit is feasible. It is also interesting to observe that, by enlarging a branch-and-cut approach for the TSP with this separation algorithm, one has an exact approach for the m-PDTSP. Indeed, this algorithm finds a violated constraint when an integer solution x^* corresponds to a nonfeasible Hamiltonian circuit.

The second heuristic algorithm for separating inequalities (17) considers a different set of paths $\overrightarrow{P}$. The aim is to generate an infeasible path $\overrightarrow{P} = (v_1, \ldots, v_s)$ such that $x_{v_1v_2}^* + x_{v_2v_3}^* + \ldots + x_{v_{s-1}v_s}^* > s - 2$. To this end, we enumerate paths in the subgraph of G induced by the arcs with positive value in x^*. Starting from each vertex $v_1 \in V$, the path enumeration is done with a recursive procedure. Values $\max_{i=0}^{s}\{\sum_{k=1}^{m} l_i^k(\overrightarrow{P})\}$ and $\min_{i=0}^{s}\{l_i^k(\overrightarrow{P})\}$ for each k are updated at each iteration, together with the vertices v' and v_k'' where these maximum and minima are obtained. Note that any infeasible path starting from v_1 and containing v' and v_k'' also induces an infeasible path starting from v' or v_k'' for some k. Therefore, the procedure backtracks when $v_1 \notin \{v', v_k'' : k \in K\}$ or (20) holds. Each time a path $\overrightarrow{P}$ is enlarged through the procedure, Inequality (19) is checked.

4.4. Separating Clique Cluster (22) and Multistar Inequalities (23) and (24)

We use the separation algorithms described in [19], which can be summarized as follows.

The algorithm to separate clique cluster inequalities (22) considers each customer v as the intersection of the subsets W_j in a clique cluster. It searches for all paths starting from each customer in $\{i_1 \in V \backslash \{v\} : x_{vi_1}^* + x_{i_1v}^* > 0\}$ and the following customers $i_2, i_3, \ldots$ in the path are such that $x_{i_1i_2}^* + x_{i_2i_1}^* = 1$, $x_{i_2i_3}^* + x_{i_3i_2}^* = 1, \ldots$. For each $K' \subseteq K$, let us consider a 1-PDTSP where $q_i = \sum_{k \in K'} q_i^k$ (i.e., relax the problem to one commodity only). We now analyze the case where we search for a clique cluster with $\sum_{i \in W_{j_1} \cup W_{j_2}} q_i > Q$. Let i_l be the first customer of each path such that $q_{i_1} + \ldots + q_{i_l}$ is maximum. The set $W_j = \{v, i_1, \ldots, i_l\}$ is a candidate set for a clique cluster defining a violated inequality, and this is checked on x^*. In a similar way, we also search for a clique cluster with $\sum_{i \in W_{j_1} \cup W_{j_2}} q_i < -Q$.

The separation algorithm for the generalized homogeneous and generalized homogenous partial multistar inequalities (24) is based on greedy heuristics to find possible candidates for the nucleus N, connector C and satellites $\{S_1, \ldots, S_s\}$. For each N, C and $\{S_1, \ldots, S_s\}$, we compute bounds to the projection over $x(A(N))$ and $\sum \big(x(A(C : S_j)) + x(A(S_j : C))\big)$. The upper frontier of the convex hull of this projection induces valid inequalities of (24), which are checked for the violation of x^*. Hence, given a fractional solution x^*, the candidates for nucleus sets are saved in a list and are obtained using the first heuristic algorithm for capacity constraints (21). This list contains sets N with $2 \leq |N| \leq n/2$. When the nucleus N has been selected from the nucleus list, to detect the satellite subsets S_i for the generalized homogeneous multistar inequalities (24) with $C = N$, we use the same greedy heuristic used to detect the sets W_i in the clique cluster inequalities (22).

Again, for each $K' \subset K$ and searching for satellites with positive demands and another searching for satellites with negative demands. All the valid inequalities (24) with $C = N$ obtained by considering the projection from $\{N, S_1, \ldots, S_s\}$ are checked for the violation of x^*. We also check the inequalities (24) with $C = N$ obtained by considering the projection from $\{N, S_1, \ldots, S_s\}\backslash\{S_j\}$ where S_j is the satellite set with the smallest absolute value demand. This procedure is repeated removing a satellite set at a time until there are two satellite sets (i.e., $s = 2$).

The separation procedure for generalized homogeneous partial multistar inequalities (24) with $C \neq N$ takes the same list of nucleus sets. For each nucleus N and each cardinality from 1 to the maximum of 4 and $|N| - 1$, a connector C is built such that $x^*(C : V\backslash N)$ is as large as possible and the process continues as when $C = N$.

Finally, the separation algorithm for the generalized inhomogeneous multistar inequalities (23) uses the same list of nucleus sets described above and the same satellite constructor. As can be observed from (23), the inequality is stronger if we consider all candidate satellites.

4.5. Separating Benders' Inequalities (15)

Benders' inequalities can be separated by solving a linear program. A solution x^* of the relaxation of model (1)–(8) defines a feasible m-PDTSP solution if, and only if, the linear system (6), (9), and (10) is feasible, or equivalently the dual problem is bounded (with optimal objective value equals to zero). If the linear system (called subproblem) is infeasible, an unbounded dual ray defines a constraint to avoid the solution x^* through a cutting-plane approach for solving the m-PDTSP. More precisely, let α_i^k be the dual variable associated with the Equalitie (6) and $\beta_{(i,j)}$ be the dual variables associated with inequalities (10). Let $\eta_{(i,j)}^k$ and $\theta_{(i,j)}^k$ be the dual variables associated with the lower and upper bound on $f_{(i,j)}^k$ in (9). Then, the valid inequality defined by a ray of the dual subproblem is like (15) but with the coefficient $Q\beta_{(i,j)}$ replaced by

$$\left(Q + \min\left\{\sum_{k\in K} q_i^k, -\sum_{k\in K} q_j^k, 0\right\}\right)\beta_{(i,j)}$$
$$+\sum_{k\in K}\left(\left(Q + \min\{q_i^k, -q_j^k, 0\}\right)\theta_{(i,j)}^k - \max\{q_i^k, -q_j^k, 0\}\eta_{(i,j)}^k\right).$$

Based on computational experiments, checking the feasibility of (6), (9), and (10) consumes more computational time than checking the feasibility of (6)–(8), while the bound from the linear programming relaxation did not change in our experiments. For that reason, we separate (15) with the coefficient $Q\beta_{(i,j)}$ on the $x_{(i,j)}$ variables. An advantage is that in many cases all $\beta_{(i,j)}$ values are integer numbers, thus we add the rounded Benders' inequality:

$$\sum_{a\in A}\beta_a x_a \geq \left\lceil \frac{1}{Q}\sum_{k\in K}\sum_{i\in V}\alpha_i q_i^k \right\rceil.$$

5. COMPUTATIONAL RESULTS

To evaluate the performances of the proposed inequalities in practice, we have implemented different approaches to solve the m-PDTSP to optimality. We used the branch-and-cut framework of CPLEX 12.1, and the implementations were executed on a personal computer with an Intel Core 2 Duo processor at 3.33 Ghz with 3.25 Gb of RAM.

We have considered two classes of m-PDTSP instances, both based on the random Euclidean generator proposed in Mosheiov [23] for the TSPPD. Briefly, in the first class of instances, each customer offers a demand of only one commodity or requires a demand of only one commodity. The second class of instances allows a customer to offer and/or require a demand for several commodities. For both classes, the random generator produces $n-1$ random pairs of coordinates in the square $[-500, 500] \times [-500, 500]$, and the depot is located in $(0, 0)$. The travel cost c_{ij} is computed as the Euclidean distance between the points i and j. Also for both classes, the depot does not demand any commodity (i.e. $q_1^k = 0$ for all $k \in K$). In the first class, the demand q_i^k is randomly generated in $[-10, 10]$ if $i > m + 1$ and $k - 1 = i \pmod m$, and $q_i^k = 0$ otherwise. For each $k \in K$, we set $q_i^k = 0$ for $1 < i \leq m+1$ and $i \neq k+1$, while q_{k+1}^k is defined so $\sum_{i\in V} q_i^k = 0$. If $q_{k+1}^k \notin [-10, 10]$ for some k, then the random values q_i^k are generated again. In the second class, the demands q_i^k are randomly generated in $[-10/m, 10/m]$ for all $1 < i < n$ and all $k \in K$. For each $k \in K$, the value q_n^k is defined to ensure $\sum_{i\in V} q_i^k = 0$. If $q_n^k \notin [-10/m, 10/m]$, then the random generator is applied to produce new q_i^k values. In both classes, $-10 \leq \sum_{k\in K} q_i^k \leq 10$ for all i. For that reason, we have considered instances with $Q \geq 10$. We generated a group of 10 random instances for each value $m \in \{2, 3\}$, each $n \in \{20, 25, 30\}$, and each $Q \in \{25, 20, 17, 15, 12, 10\}$. Therefore, our benchmark m-PDTSP collection contains 720 instances.

Our first set of experiments aims to compare different implementations based on the results in this article. For example, it is interesting to conclude whether in practice it is convenient or not to work with the flow variables, or whether it is efficient or not to include a specific separation procedure. Because there are many combinations of the material given in sections 2–4, we have selected four of them because they are the most representative. These configurations are the following:

- M1: A branch-and-cut code to solve Model (1)–(6) and (9)–(10). It applies all separation procedures described in section 4 except the separation of Benders' inequalities (15).
- M2: A branch-and-cut code solving the TSP model (1)–(5) with all the separation procedures described in section 4.
- M3: Same as M2, but the separation procedures for clique cluster inequalities (22), generalized homogeneous (partial) multistar inequalities (24), generalized inhomogeneous multistar inequality (23), and Benders' inequalities (15) are applied only on branch-and-cut nodes with a depth level smaller or equal to 10.

TABLE 1. Comparing different branch-and-cut implementations.

Instance	M1	M2	M3	M4
m2n20Q10s444c2	487.3	68.3	34.8	63.0
m2n20Q10s888c2	410.6	49.2	52.9	45.2
m2n20Q15s555c2	21.8	5.2	5.5	2.1
m2n25Q10s444c2	3600.0	730.4	214.5	281.4
m2n25Q10s888c1	3600.0	973.4	327.1	321.3
m2n25Q15s666c2	315.4	44.6	22.4	18.1
m2n25Q20s666c2	121.7	121.1	19.7	29.6
m3n20Q10s111c1	994.6	75.6	15.9	61.7
m3n20Q10s555c1	1944.0	405.5	110.0	110.0
m3n20Q10s888c2	3600.0	499.3	453.3	384.1
m3n20Q15s888c2	300.5	51.4	25.3	16.5
m3n25Q10s000c1	3600.0	756.7	246.8	281.3
m3n25Q10s111c1	3600.0	817.5	198.6	193.2
m3n25Q10s666c1	3600.0	1881.2	649.3	1960.9
m3n25Q10s999c1	3600.0	835.3	273.5	389.8
m3n25Q15s222c2	3600.0	660.7	201.8	243.2
m3n25Q15s333c2	1927.6	259.7	114.3	107.4
m3n25Q15s555c1	3600.0	481.0	435.3	107.2
m3n25Q15s555c2	1090.2	167.0	39.5	48.9
m3n25Q20s555c2	870.5	648.7	150.3	119.5
Average	2044.2	476.6	179.5	239.2

M4: Same as M2, but the separation procedures for clique cluster inequalities (22), generalized homogeneous (partial) multistar inequalities (24), generalized inhomogeneous multistar inequality (23), and Benders' inequalities (15) are not applied.

In all the combinations, we have activated the separation procedure for inequalities (17), which also contains a primal heuristic that may help the branch-and-cut algorithm.

Table 1 shows the computational time (in seconds) of the four approaches on a sample of 20 instances randomly selected among the 720 instances of our benchmark collection. The instance name shows the number of commodities, the number of customers (including the depot), the vehicle capacity, the seed used in our random generator, and the class number of the instance. We have used a time limit of 3600 s for each execution. The main observation from this table is that it is better to project out the continuous variables. In addition, it is also clear that the separation procedures help but are also time consuming. For that reason, the best setting was obtained when the separation procedures were activated only in the top-level nodes of the branch-and-cut search, that is, implementation M3. For the rest of the section, M3 is the branch-and-cut approach that we used to solve the m-PDTSP. In this implementation, we sequentially applied the separation procedures in a particular order, and only applied one procedure when the previous procedure did not find any violated inequality. Clearly different orders lead to different results, and it was not possible to find an order that outperformed the others on all instances. The only clear result was to move the separation of the Benders' inequalities to the end of the order because solving a linear program was the most expensive separation procedure of our collection.

Our second set of experiments aims to show the implicit complexity of solving a m-PDTSP instance. We have considered the example 1 given in Mosheiov [23], where the locations of the depot and the customers are given. To produce instances of the m-PDTSP, we have considered $m = 3$ and have generated m random numbers in $[-4, 4]$ for each customer. The obtained data are illustrated in Figure 3, so it can be used by other researchers to have our m-PDTSP instances. We have set two families of instances. In a first family, the initial load of the vehicle is unfixed, that is, they are direct m-PDTSP instances. In a second family, the initial load of the vehicle is forced to be zero, that is, we solve the restricted version of the m-PDTSP where the vehicle leaves from (and returns to) the depot with zero load. To this end, we have used the transformation described in section 1.

Each family of instances differs only in the vehicle capacity Q. In the first family, for a large Q, we obtain the TSP solution, and this occurs when $Q \geq 19$. In the second family, for a large Q, we obtain the TSPPC solution, which occurs when $Q \geq 21$. Because customer 18 requires a total demand of 7 units, there is no solution when $Q < 7$. For that reason, we have enumerated and solved all the possible values of Q for each family of instances, leading to 28 instances.

Table 2 shows the performance of our approach (M3) on these 28 instances. The heading of each column of Table 2 means:

Q: the capacity of the vehicle;
Cuts: the numbers of inequalities generated by the separation procedures; Sec refers to subtour elimination constraints (4), Cap. refers to capacity constraints (21), paths refers to inequalities (17), cliq. refers to clique cluster inequalities (22), hm refers to generalized homogeneous multistar inequalities (24) with $C = N$, hpm refers to generalized homogeneous partial multistar inequalities (24) with $C \neq N$, and im refers to generalized inhomogeneous multistar inequalities (23), sublp refers to Benders' inequalities (15);
r-lb/ub: the percentage of the lower bound over the best upper bound at the end of the root node;
ub: the best upper bound found during the branch-and-cut algorithm;
B&C: the number of explored branch-and-cut nodes;

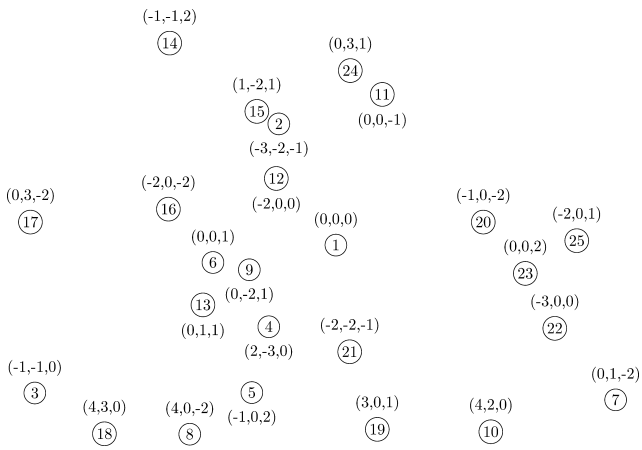


FIG. 3. Data of a TSPPD instance in Mosheiov [23].

TABLE 2. Computational results for example 1 of Mosheiov [23].

	Cuts												
Q	*Sec*	*Cap.*	Paths	Cliq.	Hm	Hpm	Im	Sublp	r-lb/ub	ub	B&C	*Time*	*Gap*
19	14	0	2	0	0	0	0	0	100.0%	4431	1	0.3	0.0%
18	28	2	38	0	0	0	0	0	98.6%	4493	19	0.9	0.0%
17	48	2	73	0	0	0	0	0	97.0%	4572	59	1.1	0.0%
16	39	25	62	0	24	0	0	0	96.7%	4614	54	1.2	0.0%
15	78	70	305	0	24	0	0	0	95.8%	4744	874	7.4	0.0%
14	63	100	158	0	16	0	0	0	96.1%	4744	447	3.4	0.0%
13	114	183	471	0	9	0	0	0	92.5%	4926	1076	10.6	0.0%
12	173	467	1411	4	25	4	0	3	93.9%	5047	3635	36.8	0.0%
11	395	1542	5559	19	41	42	0	0	89.1%	5458	31,258	936.3	0.0%
10	245	1177	3612	11	96	32	2	0	89.1%	5692	35,672	598.8	0.0%
9	412	3431	5903	35	115	104	1	0	88.1%	6028	190,479	10,000.0	2.0%
8	268	2576	5641	29	242	180	4	4	87.1%	6350	105,181	4811.3	0.0%
7	240	4098	4508	49	374	175	9	8	86.2%	6843	57,949	2906.8	0.0%
21	36	336	73	0	7	35	0	0	96.0%	4915	170	5.8	0.0%
20	40	436	118	1	5	43	0	0	92.4%	5119	498	7.7	0.0%
19	49	398	120	0	5	22	0	0	92.8%	5119	726	12.5	0.0%
18	36	424	114	1	14	32	0	0	92.9%	5119	535	9.6	0.0%
17	42	603	196	3	11	26	0	0	93.4%	5119	607	12.0	0.0%
16	57	695	200	1	57	22	0	1	93.0%	5119	686	11.6	0.0%
15	79	752	230	2	86	28	0	0	92.4%	5173	929	13.9	0.0%
14	75	1240	477	2	128	43	0	0	90.3%	5295	1989	31.3	0.0%
13	64	658	470	8	116	20	0	0	89.6%	5376	2586	37.7	0.0%
12	149	1976	1793	12	235	44	0	5	87.2%	5639	13,172	242.7	0.0%
11	225	3387	3974	28	222	85	0	1	86.9%	5852	37,159	1533.7	0.0%
10	142	3037	4797	27	291	107	3	2	86.5%	6113	126,689	4914.8	0.0%
9	205	6065	6505	42	392	198	5	5	84.8%	6431	130,679	10,000.0	2.3%
8	139	3795	3958	66	695	260	22	9	85.2%	6600	131,355	6449.7	0.0%
7	245	5859	3775	87	664	167	10	7	87.6%	6911	30,038	1590.9	0.0%

time: the total computational time in seconds;
gap: the gap between the best lower bound and the best upper bound when the algorithm stops.

This table shows that the instances are slightly more difficult to solve when the initial load of the vehicle is fixed a-priory. Observe that we solve this restricted variant of the m-PDTSP by adding a dummy commodity and by replacing the depot with two dummy customers: a pickup customer giving Q units of the dummy commodity and a delivery customer requiring Q units of the dummy commodity. This is the reason why this variant becomes more difficult than the m-PDTSP with no requirement on the initial load. Indeed, after the modification, we obtain a nonrestricted m-PDTSP where the vehicle capacity is equal to the demand of a customer, which is the smallest value of a vehicle capacity to allow a feasible route. In addition, if the number of commodities in the original problem was $m = 3$ then, after the modification, we have a problem with $m+1 = 4$ commodities, which also contributes to the complexity of the problem.

Another feature with a direct impact on the complexity of the problem is the vehicle capacity. When Q is smaller, then the problem is more difficult, except when Q is close to being the smallest value that allows a feasible m-PDTSP solution. In this situation, the inequality (11) holds for many pairs (i, j) of customers, and therefore many variables x_{ij} can be removed. For this set of experiments, the time limit was set to 10000 s, and the two instances with $Q = 9$ remain unsolved. Analyzing the columns with the number of cuts, the most successful separation procedure was the separation of inequalities (17), based on finding infeasible paths. On the other hand, the separation of the generalized inhomogeneous multistar inequalities (23) generated a tiny number of inequalities, and by deactivating this separation the total time for solving these instances can be slightly reduced. An

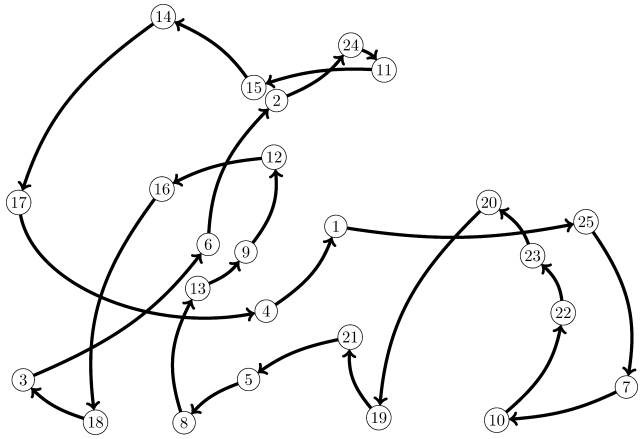


FIG. 4. Optimal route of a 3-PDTSP on Mosheiov's data with $Q = 7$ and unfixed initial load.

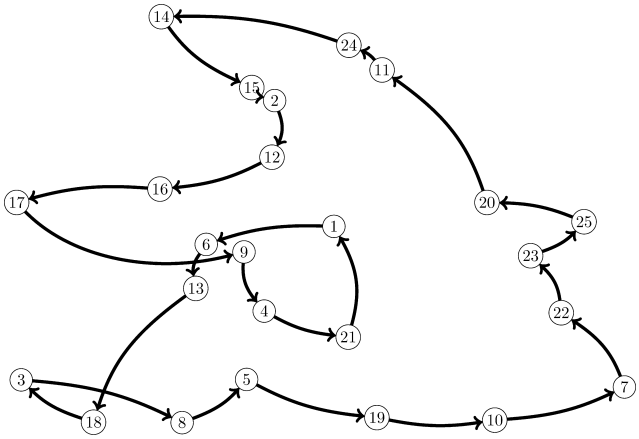


FIG. 5. Optimal route of a 3-PDTSP on Mosheiov's data with $Q = 21$ and initial load fixed to zero.

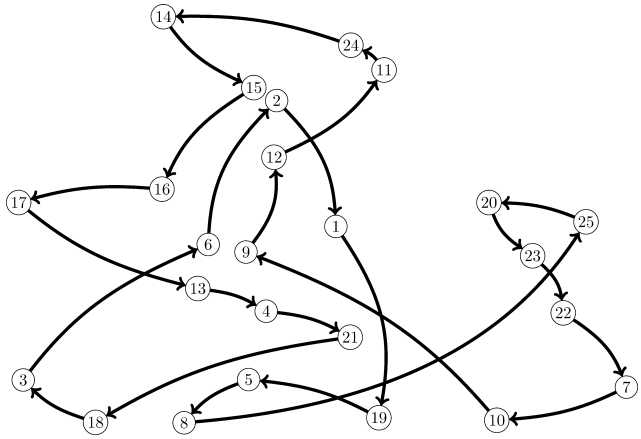


FIG. 6. Optimal route of a 3-PDTSP on Mosheiov's data with $Q = 7$ and initial load fixed to zero.

interesting observation is that the separation of Benders' inequalities is an exact procedure to find violated inequalities (15). Because the numbers in column *sublp* are small, we deduce that the other separation procedures have found the necessary inequalities to ensure the *m*-PDTSP feasibility of TSP solutions.

TABLE 3. Computational results on the first class of *m*-PDTSP instances.

			Cuts												
n	*m*	Q	Sec	Cap.	Paths	Cliq.	Hm	Hpm	Im	Sublp	r-lb/ub	ub	Time	*t.l.*	Gap
20	2	10	42.2	129.7	194.7	4	22.8	10.8	0.2	0.2	96.3%	125.1%	3.6	0	0.00%
		12	42.1	89.3	153.5	3.7	18.8	7.2	0.8	0.1	95.8%	118.3%	2.3	0	0.00%
		15	26.4	17.7	30.7	0.1	3.4	0.7	0	0	98.2%	108.6%	0.3	0	0.00%
		17	22.9	9.1	19.5	0	0	0	0.1	0	98.9%	105.4%	0.2	0	0.00%
		20	17.9	2.9	3.2	0	0	0	0	0	89.5%	102.7%	0.0	0	0.00%
		25	16.8	1.1	1.2	0	0	0	0	0	99.7%	100.9%	0.0	0	0.00%
20	3	10	98.4	230.5	917	5.5	21.2	6.2	0.1	0.4	91.5%	124.5%	20.6	0	0.00%
		12	72.2	75	373	0.8	10.8	2.1	0	0.1	94.2%	112.8%	4.3	0	0.00%
		15	29.5	9.2	42.7	0	0.3	0.1	0	0	87.2%	105.8%	0.6	0	0.00%
		17	22.6	1.8	17	0	0	0	0	0	98.5%	102.8%	0.2	0	0.00%
		20	16.2	0.1	1.5	0	0	0	0	0	89.8%	100.6%	0.0	0	0.00%
		25	15.6	0	0	0	0	0	0	0	99.9%	100.0%	0.0	0	0.00%
25	2	10	87.5	424.8	743.8	15.6	106.8	37.5	1.1	1.4	93.3%	128.1%	70.4	0	0.00%
		12	70.1	140.8	304.7	6.6	24.8	11	0.2	0.1	95.5%	116.4%	6.8	0	0.00%
		15	34.3	27.5	46.9	0.8	3	0.3	0	0	98.9%	106.3%	0.5	0	0.00%
		17	27.2	15	17.6	0.1	0.5	0	0	0	99.0%	103.5%	0.4	0	0.00%
		20	23.3	4.8	5.6	0	0.1	0	0	0	99.2%	101.8%	0.1	0	0.00%
		25	20.7	0.2	0.1	0	0	0	0	0	99.6%	100.0%	0.1	0	0.00%
25	3	10	274.5	1263.8	3725.7	23.6	128.1	60.2	2.9	2.5	89.8%	133.5%	890.6	2	0.68%
		12	151.8	443.7	1720.3	7.7	66	32.1	0.9	1.3	92.1%	121.1%	231.4	0	0.00%
		15	76	98.2	414	0.3	13.6	2.1	0	0.1	95.4%	111.3%	25.1	0	0.00%
		17	45.5	22	130.3	0	4.1	0.1	0	0.1	87.2%	106.2%	3.4	0	0.00%
		20	29	4.4	27.1	0	0.7	0	0	0	98.7%	102.5%	0.5	0	0.00%
		25	20.8	0	4.1	0	0	0	0	0	99.5%	100.2%	0.1	0	0.00%
30	2	10	180.6	1357.7	2413.5	30.9	290.6	83.8	5.7	2.7	91.8%	136.9%	852.3	1	0.53%
		12	153.5	661.1	1331.3	11.4	122.8	46	1.1	1.2	93.4%	125.3%	244.3	0	0.00%
		15	108.9	236.3	423.4	2.5	38.4	10.9	0.6	0	95.2%	115.1%	18.4	0	0.00%
		17	80.7	100.5	236.1	0.4	20.7	2.1	0	0.1	96.0%	110.8%	11.0	0	0.00%
		20	46.4	32.6	54.6	0.1	2.4	0.3	0	0	98.0%	105.2%	2.4	0	0.00%
		25	30.9	9.3	9.9	0	1.8	0	0	0	98.8%	101.8%	0.6	0	0.00%
30	3	10	611.8	4283.5	8775.9	14.8	130.9	45.2	2.1	6.4	82.8%	148.6%	2262.3	5	8.95%
		12	273.7	973	3477.2	2.6	58.1	30.8	0.6	0.7	91.4%	123.1%	1013.9	1	1.20%
		15	140.5	265.4	1061.1	1	31.3	11.7	0.3	0.1	94.8%	111.9%	272.8	0	0.00%
		17	85.8	63.7	308	0.1	7.7	3.3	0	0.1	96.1%	108.1%	18.8	0	0.00%
		20	53.4	20.1	107	0.1	4.1	0.5	0	0	87.8%	103.7%	3.6	0	0.00%
		25	29.2	2.4	12.7	0	2.9	0	0	0	89.0%	101.1%	0.7	0	0.00%

TABLE 4. Computational results on the second class of m-PDTSP instances.

			Cuts												
n	m	Q	Sec	Cap.	Paths	Cliq.	Hm	Hpm	Im	Sublp	r-lb/ub	ub	Time	*t.l.*	Gap
20	2	10	50.5	573.2	384.2	42	281.1	78.9	4.2	4.4	91.9%	151.7%	18.0	0	0.00%
		12	44.2	215	195.3	15.2	66.5	24.2	0.9	0.8	95.2%	132.8%	3.4	0	0.00%
		15	37.2	82.2	103	2.7	9.8	4	0	0.1	96.4%	120.5%	1.1	0	0.00%
		17	32.9	40.2	76.3	0.8	1.5	1.6	0	0.1	96.8%	115.0%	0.7	0	0.00%
		20	26.1	16.2	34.5	0	3.7	0	0	0	97.7%	110.0%	0.3	0	0.00%
		25	19.2	1.7	8.2	0	0	0	0	0	99.3%	103.0%	0.1	0	0.00%
20	3	10	45.9	1076.9	493.9	41.9	251.5	79.6	1.8	5.5	93.1%	153.7%	85.4	0	0.00%
		12	54	449.2	413.7	24.6	117.1	45.2	1.3	4.1	93.1%	140.7%	25.1	0	0.00%
		15	54.6	243.7	278.9	9.5	30.5	19.5	0	0.8	95.1%	126.2%	5.3	0	0.00%
		17	44.8	104.1	169.3	2.4	11.8	5.3	0	0.5	96.5%	118.3%	1.8	0	0.00%
		20	33.7	47.6	77	1.2	3.8	0.6	0	0	96.9%	111.1%	1.0	0	0.00%
		25	31.9	11.8	40.5	0	0.8	0.2	0	0	98.0%	105.2%	0.5	0	0.00%
25	2	10	99.1	2003.8	973.3	51.2	462.3	70.1	15	8.2	93.4%	158.9%	759.2	2	0.77%
		12	109.2	1007.4	681.3	29.6	196.9	48.9	1.7	3.2	95.4%	144.1%	382.1	1	0.04%
		15	52.9	250.5	194.8	6.9	62.4	11	0.6	0.6	96.2%	127.9%	6.3	0	0.00%
		17	62.7	192.6	232.2	6.7	30.4	12.5	0.2	0.4	96.3%	121.7%	8.3	0	0.00%
		20	53.8	134.8	239.3	1.3	14.4	2.8	0.2	0.1	96.7%	114.1%	9.0	0	0.00%
		25	35.1	34.2	64.8	0	2.7	3	0.1	0	97.8%	106.7%	1.5	0	0.00%
25	3	10	147.3	4968.0	2475.0	81.4	676.1	148.5	11.5	20.6	89.0%	162.3%	1819.3	4	1.42%
		12	146.9	2990.2	2339	64.4	358.8	141.9	1.4	10.7	89.8%	146.6%	947.3	2	0.95%
		15	138.2	1166.3	1390.4	25.7	137.8	59.5	0.5	4.3	93.0%	131.0%	222.8	0	0.00%
		17	107.6	555.6	771.4	8.8	52.7	17	0.1	1.9	93.6%	123.9%	47.5	0	0.00%
		20	107.8	329.8	646.5	2.9	34.1	15.8	0.1	1.3	94.0%	114.3%	39.2	0	0.00%
		25	39.2	22.1	64.8	0	4.6	0	0	0.2	98.0%	105.3%	1.2	0	0.00%
30	2	10	150.1	6004.1	1752.3	94.3	1055.7	170.8	45.4	13.9	89.5%	167.6%	1436.3	3	3.92%
		12	166	2785	1563.3	73.2	577.8	144.9	7.9	9.3	92.0%	146.2%	1297.8	3	0.62%
		15	124.7	1468	1333.2	42.4	266.5	87.7	4.8	4.9	93.5%	130.4%	794.8	1	0.38%
		17	131.5	1082.7	1041.7	26	230	67.6	1.3	1.8	94.2%	123.5%	409.1	0	0.00%
		20	100.8	602.4	715.2	7	104.3	32.4	0.5	0.3	95.7%	114.8%	314.6	0	0.00%
		25	54.7	135.8	176.3	1.3	21.3	13.4	0	0	97.2%	107.2%	23.8	0	0.00%
30	3	10	213.8	11,659.9	3510.5	142.5	1329.1	282.8	11.8	48	81.9%	189.4%	3578.2	8	9.83%
		12	183.5	6024.8	3408.3	91.1	634.9	183.6	2.9	22.4	88.9%	155.6%	2739.8	6	2.71%
		15	243	3827.4	3430.7	42.7	289	108	1.2	12.4	91.6%	136.1%	1625.5	2	0.92%
		17	210.3	2702.5	2935.3	21.7	164.3	58.7	1.1	6.3	91.4%	128.8%	1222.5	3	1.70%
		20	203.3	1145.4	1945.5	6.2	74.4	24	0.2	3.6	92.3%	119.6%	560.8	1	0.11%
		25	114.2	188.9	578.9	0.4	28.3	2.7	0	1.8	94.6%	110.9%	63.7	0	0.00%

Figure 4 shows the optimal solution found by our M3 implementation on the m-PDTSP instance with $Q = 7$ and unfixed initial load. The nonplanar drawing of the optimal solution, although the travel distances are Euclidean, makes the complexity of the problem clear. Figure 5 shows the optimal solution of the TSPPC, which is the m-PDTSP with $Q \geq 21$ and the initial load of the vehicle fixed to zero. The optimal solution of this restricted problem when $Q = 7$ is in Figure 6, which is also a route far from being a planar drawing.

The third set of experiments aims to show details and limitations of our M3 implementation. To this end, we have solved the 720 m-PDTSP instances in the benchmark collection described at the beginning of this section. Table 3 summarizes the result on the first class of instances (i.e., instances where each customer picks up one commodity and/or delivers one commodity) and Table 4 on the second class (i.e., instances where each customer may provide or require demands of several commodities). The results are grouped according to the values (n, m, Q), and the average results are given in each cell of each table. The meanings of the column headings are the ones described for Table 2, except that now each value is the average over the number of feasible m-PDTSP instances in each group. This number is 10 for all the groups, except for two groups in Table 4: the group with $(n, m, Q) = (20, 2, 10)$ which was 9, and the group with $(n, m, Q) = (25, 3, 10)$ which was 8. Tables 3 and 4 do not show the column *ub* because we think that average objective values of optimal solutions are of no use. Instead, they show a new column *ub/tsp* that gives the average percentage of the best m-PDTSP upper bound with respect to the TSP optimal objective value. There is also a new column *t.l.* which shows the number of m-PDTSP instances that were not solved within our time limit of 3600 s. We do not show the column B&C in these tables for lack of space.

From Tables 3 and 4, again, the difficulty of solving an instance is highly dependent on Q. The problem is more complex when Q is close to the largest total demand of a customer, which is value 10 in our instances. Comparing 3

and 4, it is also obvious that the problem is harder to solve when the pickup (or delivery) demand of a customer concerns more than one commodity. In this case, the best m-PDTSP objective value was 189.4% over the optimal TSP objective value, which means that the length of the best m-PDTSP route is almost double the length of an optimal TSP route. When analyzing the number of inequalities, for both families and all groups, the separation procedure of inequalities (17) was the one generating more violated cuts. Our experiments confirm that, even if this separation is heuristically solved, it is responsible for the small number of generated Benders' inequalities. In addition, it is worth noting that the separation procedure of inequalities (17) also provides the branch-and-cut code with a primal heuristic that helps to find good upper bounds. Activating the separation procedures of the clique cluster and multistar inequalities become useful on difficult m-PDTSP instances. For example, without these separation procedures, our code does not solve within the time limit any instances with $(n, m, Q) = (30, 3, 10)$ and a fixed initial load. This is also coherent with our conclusion from comparing M3 and M4 in Table 1.

6. CONCLUSIONS

This article proposes an exact approach to solve the m-PDTSP. This problem is a very interesting and complex routing problem as it generalizes or is related to many other variants that have been addressed in the vehicle routing literature during recent years. It concerns the problem of finding a minimum-cost Hamiltonian route for a capacitated vehicle that must collect and deliver several products, each one with possibly several origins and several destinations. Although by default the initial load of the vehicle is unfixed, the restricted variant where the vehicle is required to leave the depot with zero load can be addressed.

An integer linear programming model based on two-index variables has been described, together with a set of valid inequalities to strengthen the linear programming relaxations. New and useful inequalities have been proposed to remove infeasible paths in a solution. Separation procedures have been described and implemented in a branch-and-cut approach. The performances of the implemented code have been analyzed on a benchmark collection of 720 randomly generated instances with different features. The implementation was able to solve instances with up to 30 customers, three commodities, and very small vehicle capacities.

REFERENCES

[1] S. Anily and J. Bramel, Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries, Nav Res Log 46 (1999), 654–670.

[2] S. Anily and G. Mosheiov, The traveling salesman problem with delivery and backhauls, Oper Res Lett 16 (1994), 11–18.

[3] E. Balas, M. Fischetti, and W.R. Pulleyblank, The precedence-constrained asymmetric traveling salesman polytope, Math Program 68 (1995), 241–265.

[4] N. Ascheuer, M. Jünger, and G. Reinelt, A branch-and-cut algorithm for the asymmetric traveling salesman problem with precedence constraints, Comput Optim Appl 17 (2000), 61–84.

[5] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, An exact algorithm for the traveling salesman problem with deliveries and collections, Networks 42 (2003), 26–41.

[6] G. Berbeglia, J. Cordeau, I. Gribkovskaia, and G. Laporte, Static pickup and delivery problems: A classification scheme and survey, Top 15 (2007), 1–31.

[7] G. Berbeglia, J. Cordeau, and G. Laporte, Dynamic pickup and delivery problems, Eur J Oper Res 202 (2010), 8–15.

[8] L. Bianco, A. Mingozzi, S. Riccardelli, and M. Spadoni, Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming, INFOR 32 (1994), 19–32.

[9] C. Bordenave, M. Gendreau, and G. Laporte, A branch-and-cut algorithm for the nonpreemptive swapping problem, Nav Res Log 56 (2009), 478–486.

[10] P. Chalasani and R. Motwani, Approximating capacitated routing and delivery problems, SIAM J Comput 28 (1999), 2133–2149.

[11] D. Chemla, F. Meunier, and R. Wolfler Calvo, Bike hiring system: solving the rebalancing problem in the static case (2011). Working Paper.

[12] G. Erdoğan, J.F. Cordeau, and G. Laporte, A branch-and-cut algorithm for solving the non-preemptive capacitated swapping problem, Discrete Appl Math 158 (2010), 1599–1614.

[13] M. Gendreau, G. Laporte, and D. Vigo, Heuristics for the traveling salesman problem with pickup and delivery, Comput Oper Res 26 (1999), 699–714.

[14] L. Gouveia and P. Pesneau, On extended formulations for the precedence constrained asymmetric traveling salesman problem, Networks 48 (2006), 77–89.

[15] N. Mladenović, D. Urošević, S. Hanafi, and A. Ilić, A general variable neighborhood search for the one-commodity pickup-and-delivery travelling saleman problem, Eur J Oper Res 220 (2012), 270–285.

[16] H. Hernández-Pérez, I. Rodríguez-Martín, and J.J. Salazar-González, A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem, Comput Oper Res 36 (2009), 1639–1645.

[17] H. Hernández-Pérez and J.J. Salazar-González, A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, Discrete Appl Math 145 (2004), 126–139.

[18] H. Hernández-Pérez and J.J. Salazar-González, Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, Transp Sci 38 (2004), 245–255.

[19] H. Hernández-Pérez and J.J. Salazar-González, The one-commodity pickup-and-delivery traveling salesman problem: inequalities and algorithms, Networks 50 (2007), 258–272.

[20] H. Hernández-Pérez and J.J. Salazar-González, The multi-commodity one-to-one pickup-and-delivery traveling salesman problem, Eur J Oper Res 196 (2009), 987–995.

[21] A.J. Hoffman, "Some recent applications of the theory of linear inequalities to extremal combinatorial analysis," Proc.

Symp. in Applied Mathematics, R. Bellman and Jr. Hall M. (Editors), 1960, pp. 113–127.

[22] P. Mirchandani, Projections of the capacitated network loading problem, Eur J Oper Res 122 (2000), 534–560.

[23] G. Mosheiov, The traveling salesman problem with pickup and delivery, Eur J Oper Res 79 (1994), 299–310.

[24] T. Raviv, M. Tzur, and I.A. Forma, Static repositioning in a bike-sharing system: Models and solution approaches, Working Paper, 2011.

[25] I. Rodríguez-Martín and J.J. Salazar-González, Hybrid heuristic approaches for the multi-commodity one-to-one pickup-and-delivery traveling salesman problem, J Heuristics 18/6 (2012), 849–867.

[26] M.W.P. Savelsbergh and M. Sol, The general pickup and delivery problem, Transp Sci 29 (1995), 17–29.

[27] F. Zhao, S. Li, J. Sun, and D. Mei, Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem, Comput Ind Eng 56 (2009), 1642–1648.