Computers and Operations Research 113 (2020) 104805

Contents lists available at ScienceDirect

Computers and Operations Research

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

Pickup and delivery problem with incompatibility constraints

Pablo Factorovich[a,b], Isabel Méndez-Díaz[a,c], Paula Zabala[a,c,*]

[a] Departamento de Computación, FCEN, Universidad de Buenos Aires, Argentina
[b] Departamento de Ciencia y Tecnología, Universidad Nacional de Quilmes, Argentina
[c] Instituto de Investigación en Ciencias de la Computación (ICC), CONICET-UBA, Argentina

ARTICLE INFO

Article history:
Received 20 February 2019
Revised 31 July 2019
Accepted 15 September 2019
Available online 16 September 2019

Keywords:
Pickup and delivery
Incompatibility
Branch and cut

ABSTRACT

The purpose of this paper is to present a new version of the One-to-One Pickup and Delivery Problem in which a single vehicle must comply with requests for transportation from specific collect points to specific delivery points. The problem we consider, besides looking for a minimum cost route, adds extra constraints that forbid some requests to be in the vehicle at the same time.

We first begin to formally define the problem and show how it is related to the classic graph coloring problem. Then, we introduce a comparative analysis of the computational performance of three integer programming formulations. Some polyhedral results of the most promising formulation are presented in order to strengthen the LP relaxation for increasing the computational efficacy of the model. We implement separation algorithms, a primal heuristic for finding feasible solutions and a branching strategy. All these elements were considered to the development of a Branch and Cut algorithm which is tested on a comprehensive test bed of instances. The algorithm proves to be capable of overcoming state-of-the-art mixed-integer solvers, both in number of solved instances and computational time.

1. Introduction

Pickup and Delivery Problems (PDPs) represent an important family of optimization routing problems whose aim is to design optimal routes to comply with a set of requests. Each request consists of satisfying a pick demand on every origin point (pickup point) and delivering the demand on every destination point (delivery point).

Pickup and Delivery problems, which have been intensively studied in the literature, have a large set of applications such as logistics, courier services, bus routing, laundry service of hotels, transportation for handicapped people, etc. Many variants of the problem arise from side constraints like time windows, multiple or mono vehicle, order loading and unloading sequences, vehicle capacity, splitting loads, etc. We refer the reader to Berbeglia et al. (2007), Parragh et al. (2008) and Iori and Martello (2010) for comprehensive surveys of the PDP and its variations which describe classification schemes as well as solution methods.

In this paper, we will focus on the Single Vehicle One-to-One Pickup and Delivery Problem which is a generalization of the well-known Traveling Salesman Problem (TSP) (Applegate et al., 2006). The problem is related to the Sequential Ordering Problem (Letchford and Salazar-González, 2016), also known as the Asymmetric Traveling Salesman Problem with precedence constraints (Sarin et al., 2005). There is a set of pickup points which must be visited exactly once by a single vehicle attending their demands while minimizing the total travel distance. The vehicle starts the route in an origin point (s) and finishes in a destination depot (t). Moreover, any unit of the product collected from a pickup customer must be delivered to a specific delivery point along the route, therefore establishing precedence constraints in the order the vehicle should travel from one point to the other.

If the request refers to people who have to be transported from one point to another, the problem is often called the dial-a-ride problem. As in the general case of Pick and Delivery problems, typical features of routing problems can also be considered: route duration, ride time, multiple objectives, time windows, etc. In Battarra et al. (2014), Cordeau et al. (2008), Cordeau and G.Laporte (2007), Ho et al. (2018) and Molenbruch et al. (2017), a comprehensive literature review and state of the art of the problem are provided.

A different scenario arises for the transportation of certain incompatible goods like food and detergents or hazardous materials. This legal matter is addressed by every country's legislation. The UN recommends, through its Economic and Social Council's

* Corresponding author.
E-mail addresses: factorovich@gmail.com (P. Factorovich), imendez@dc.uba.ar (I. Méndez-Díaz), pzabala@dc.uba.ar (P. Zabala).

https://doi.org/10.1016/j.cor.2019.104805
0305-0548/

Committee (ONU, 2007), a serie of regulations for the transport of hazardous materials which are embodied in the legislation of many countries. These regulations prohibit the concurrent transportation of certain incompatible goods. This is usually accomplished with the use of different vehicles or specialized vehicles with compartments to serve requests in conflict.

In this work, we propose an alternative approach where the same vehicle is used to accomplish the tasks and simply forbids those routes that include incompatible goods in the vehicle at the same time. We will name this problem *Pickup and Delivery Problem with Incompatibilities* (from now on PDPI). If we consider instances where every pair of goods is incompatible, we have a TSP instance since every pickup point has to be followed by its delivery point in a feasible solution. So, we can conclude that PDPI is NP-hard. As far as we know, there are no articles on the particular version of the problem we propose.

In this paper, based on an integer programming formulation, we develop a Branch and Cut algorithm which proves to be an efficient method for solving PDPI.

The remainder of this paper is organized as follows: in Section 2 we describe some relevant works related to incompatibility and their applications. In Section 3, we formally definite PDPI and establish a relationship with the vertex coloring problem. Three integer programming models are presented in Section 4, which are computationally evaluated in Section 5. Section 6 presents some polyhedral results and a collection of valid inequalities for the resulting most promising model. Our approach for solving PDPI using this model is described in Section 7, followed by our numerical results in Section 8. Finally, we are able to draw some conclusions in Section 9.

We close this section by introducing all the notation and definitions used throughout the paper. Let $G = (V, E)$ be a graph and $S \subset V$. $G[S] = (S, E_S)$ is the induced subgraph of G by S with $E_S = \{(u, v) : u, v \in S \text{ and } (u, v) \in E\}$. S is a clique if $(u, v) \in E$ for all $u, v \in S$. S is a stable set or independent set if $(u, v) \notin E$ for all $u, v \in S$. The stability number of G, $\alpha(G)$, is the maximum size of an independent set in G. A sequence $v_1, \ldots, v_k$ of pairwise distinct vertices is a path in G if $(v_1, v_2), \ldots, (v_{k-1}, v_k) \in E$. A path is Hamiltonian if it contains every vertex of the graph. A path is a cycle if in addition $(v_1, v_k) \in E$. A hole is an induced graph which is a cycle.

The neighborhood of v is $N(v) = \{u : u \in V \text{ and } (u, v) \in E\}$ and $d(v) = |N(v)|$ is the degree of v. Two vertices u and v are adjacent if $u \in N(v)$. A coloring of G is an assignment of colors to the vertices in V where no two adjacent vertices share the same color. The minimum number of colors to define a coloring of G is called its chromatic number and it is denoted by $\chi(G)$.

2. Literature review

Incompatibility is a very general term that usually implies conflicts among things with mutually exclusive needs or risky interaction.

In routing problems, it could arise because a vehicle type can be incompatible with the transportation of some items and/or with serving some locations. There could be incompatibilities among items to be present in a vehicle at the same time or among items and locations due to storage or reception requirements.

In the literature, several publications have addressed incompatibility in distribution of goods in many routing problems. As far as we know, the incompatibility issue is frequently tackled through the use of different vehicles/routes or specialized vehicles with compartments to serve requests in conflict. The former leads to solutions which require more number of vehicles/routes, and the latter requires more expensive vehicles (and it could still be dangerous in case of an accident).

In Sun (2002) the author presents a classical transportation problem where goods have to be shipped from source locations to warehouses with the goal of minimizing cost transportation. Extra constraints forbid a warehouse to store goods which could cause damage or deterioration to each other or to the facility. Two B&B algorithms for solving the problem are developed and computationally tested. A similar situation is considered in Goossens and Spieksma (2009) where there is a set of supply nodes with a given availability of a good and a set of demand nodes each with a given demand. The authors consider the variant where, for each demand node, a set of pairs of supply nodes is given such that at most one supply node of each given pair is allowed to send items to that demand node. In both works, the authors refer to the problem as *The Transportation Problem with Exclusionary Side Constraints*.

A more generalized situation for a transportation problem is presented in Ficker et al. (2018) where the problem is named *The transportation problem with conflicts*. In this case, each source location could have pairs of demand points which could be attended at most one in each pair. Moreover, each demand point may have pairs of source locations such that it can accept goods at most from one source of each conflicting pair. The paper presents complexity results based on particular structures of the conflict graph which emerges from the conflicting pairs. Applications of this problem arise in the assignment of patients to hospital rooms (Vancroonenburg et al., 2014) where incompatibility could appear by gender and/or kind of illness, and storage management (Cao, 1992), Cao and Uebe (1995) where containers have to be set in rows of a storage yard but some containers must not be in the same row, due to their content or size.

In the context of *The Multi-vehicle Traveling Purchaser Problem*, incompatibility has also been considered. Given a fleet of homogeneous vehicles with a given capacity and a set of different suppliers, the objective of this problem is to satisfy product demands at the minimum traveling and purchasing cost. In addition, pairwise incompatibility constraints among products must be fulfilled; these constraints establish that two incompatible products can not be transported on the same vehicle. In Gendreau et al. (2016), the paper focuses on the particular version of unitary demands applied, for example, in an extension of the school-bus location and routing problem (Riera-Ledesma and Salazar-González, 2012). The authors propose a column generation approach where the columns represent a tour visiting a subset of suppliers where incompatible products cannot be part of the same tour. In Manerba and Mansini (2015), a branch-and-cut method is proposed to the general case as well as a heuristic that provides good initial solution for the exact approach.

Regarding health care applications, Manerba and Mansini (2016) consider a Nurse Routing Problem. The problem aims to scheduling the visits of a set of nurses to a set of patients by considering workload constraints. Potential incompatibility restrictions among pairs of services that can not be performed at the same patient during the same day are also considered. The authors model the problem as a variant of the Multi-vehicle Traveling Purchaser Problem and develop a branch-and-price approach.

In Battarra et al. (2009) a variant of *The Multiple Trip Vehicle Routing Problem* is addressed. The problem looks for determining optimal routes to be followed by a fleet of vehicles with a given capacity in order to satisfy customer demands. Each vehicle could perform more than one route and the objective is to minimize the size of the required fleet. Many additional constraints are considered, like time windows, processing time of loading/unloading operations and maximum length route. In that work, the authors also add that goods belonging to multiple commodities cannot

be transported together on the same vehicle. The problem originates from the distribution of goods to supermarkets and hypermarkets from a centralized depot where the commodities are vegetables, milk, meat and non-perishable items. An iterative solution approach based on the decomposition of the problem into simpler ones is proposed.

The Directed Profitable Rural Postman Problem is an interesting arc routing problem that looks for a tour that maximizes the sum of the differences between profit and cost associated for going through an arc. The variant presented in Colombi et al. (2017) adds incompatibility constraints which may exist between nodes and profitable arcs leaving them. The problem comes from a real case application in the domain of transportation services. Two mathematical formulations are presented and a matheuristic procedure is proposed.

In *The Petrol Station Replenishment Problem* it is necessary to optimize the delivery of petroleum products by using a fleet of heterogeneous and, in general, compartmented tank trucks. Usually, there are some constraints that impose time windows, minimum and maximum quantities to be delivered and incompatibility between stations if there is no truck that can deliver all their orders together. In Cornillier et al. (2012), a matheuristic procedure is proposed based on an integer programming formulation where columns represent a trip that verifies that incompatible stations cannot be visited during the same trip. *Multi-Compartment Vehicle Routing Problem* is also considered, among others, in the context of glass waste collection (Henkea et al., 2019), milk collection (Caramia and Guerriero, 2010) and oil collection (Lahyani et al., 2015).

Ceselli et al. (2009) address the problem to compute a daily plan for a real complex vehicle routing problem which considers several operational difficulties (time windows, driver's work time, splitting up orders, etc), and incompatibility constraints between goods. The authors propose a column generation algorithm where the columns represent a feasible vehicle duty (a sequence of routes) where incompatibilities among goods imply that they are not transported in the same route.

A different way to deal with incompatible goods is presented in Hu et al. (2015) where incompatibility is represented by a contamination factor and the goal is to find a tradeoff among transportation cost, incompatibility and service quality. A similar problem is considered in Paredes-Belmar et al. (2017) where industrial HAZMAT (dangerous goods, hazardous materials and items) with different risk levels are transported with a truck fleet. A truck could carry different items; however, the consequences of an accident involving a combination of items could cause a lot of damage. Then, the objective is to find routes for transporting the items which minimize the exposure of the population to hazard and the total transportation cost.

As well as routing problems, incompatibility has also been considered in other problems. An extension of classic *Parallel Machine Scheduling Problem*, where a set of jobs is scheduled on identical parallel machines, results when it is necessary to deal with conflictivity between jobs impossible to be scheduled on the same machine (Kowalczyk and Leus, 2017). The goal is to minimize the maximum completion time of processing all jobs. Practical applications come from TV advertisement scheduling and resource assignment in workforce planning.

In *The Bin Packing Problem*, items of different weights must be packed into a minimum number of identical bins of a given capacity. In the variant with conflicts (Gendreau et al., 2004; Sadykov and Vanderbeck, 2012), pairs of items may not be assigned to the same bin. The problem arises in real-world applications like product delivery, workforce planning and examination scheduling.

3. Formal definition and connection with graph coloring problem

We start giving a formal definition of PDP. Let n be the number of requests and $G = (V_G, A_G)$ a directed graph where $V_G = \{1, \ldots, 2n, s, t\}$ is the vertex set and A_G the arc set, where each arc (i, j) has a positive cost $c_{i,j}$. Vertices s and t represent origin and destination depots. The sets $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ denote the pickup and delivery sets, respectively. Requests are pairs $(i, i+n)$, where vertex $i \in P$ and $i+n \in D$. Note that $(i, j) \in A_G$ for all $i, j \in V_G$, except that, for all $i \in P$, $(i+n, i) \notin A_G$ and (s, i) and $(i+n, t)$ are the only incident arcs to s and t, respectively.

The goal of the problem is to find the minimum cost Hamiltonian path (path that visits each vertex of the graph exactly once) among those starting from s, finishing at t and satisfying all pickup and delivery requests.

In PDPI we add an incompatibility undirected graph $I = (V_I, E_I)$ associated to the instance, where $V_I = P$ and $(i, j) \in E_I$ iff goods picked at i and j must not be in the vehicle at the same time, i.e., i and j are incompatible. Then, the goal of PDPI is to find the minimum cost Hamiltonian path among those starting from s and finishing at t, which satisfy the pickup and delivery requests and forbid incompatible goods from being at the same time in the vehicle.

Notice that vertices of graph G refer to origin or destination of requests, not customer locations. Therefore, a location could be origin or destination of several requests and we do not impose that all of them must be satisfied during a single visit. They are considered individual tasks that are performed together or separately, depending on conflicts and costs.

Let us consider an example of 4 requests. In Fig. 1, the graph $G = (V_G, A_G)$ shows all arcs between nodes. If no incompatibility has to be satisfied, then Fig. 2 shows a feasible path P_1. This is a feasible path since all pickup points are visited before their corresponding delivery ones. Fig. 3 shows a graph I which represents pairs of incompatibilities. It is easy to see that P_1 results not feasible since, for example, goods picked at 3 and 4 are present in the vehicle at the same time. In Fig. 4 we show P_2, a feasible path that satisfies precedence between pairs of pickup and delivery points and all incompatibility constraints.

In order to provide a better understanding of the problem we are introducing, we show the relationship between this problem and the classic vertex coloring problem. This connection allows us to establish that the chromatic number of I is a lower bound for

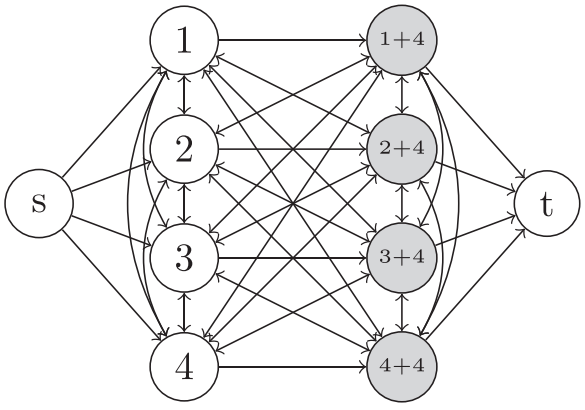

Fig. 1. Graph $G = (V_G, A_G)$.

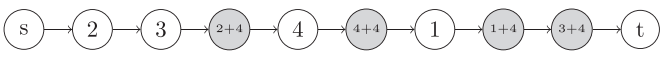

Fig. 2. P_1 a feasible path for PDP.

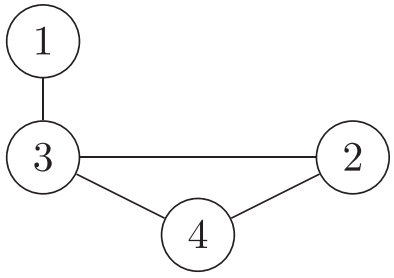

Fig. 3. Incompatibility Graph *I*.

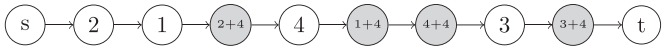

Fig. 4. P_2 a feasible path for PDPI.

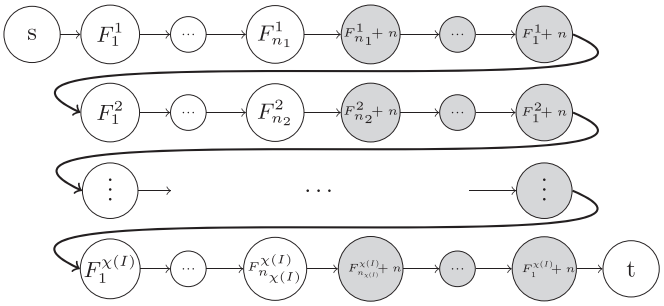

Fig. 5. A feasible Hamiltonian path based on coloring classes .

the number of delivery-pickup arcs used in a valid solution. Moreover, it helps to fully comprehend the valid inequalities that will be presented later.

Proposition 1. *Let $G = (V_G, A_G)$ be a directed graph for a PDPI instance. Consider arc cost $c_{i,j} = 1$ if i and j are delivery and pickup nodes, respectively, and $c_{i,j} = 0$ for every other arc in A_G. Moreover, let $I = (V_I, E_I)$ be the associated incompatibility graph, $\chi(I)$ its chromatic number and $\Pi(G, I)$ the optimum value for PDPI. Then, $\Pi(G, I) = \chi(I) - 1$*

Proof. We will show $\Pi(G_I, I) \leq \chi(I) - 1$ and $\Pi(G_I, I) \geq \chi(I) - 1$.

$\Pi(G_I, I) \leq \chi(I) - 1$

Let us consider a valid $\chi(I)$-coloring of I and $C_i \subset V_I$ the independent set of vertices that are assigned with color i, for all $i \in 1 \dots \chi(I)$. We enumerate vertices in C_i by $F_1^i, \dots, F_{n_i}^i$, $n_i = |C_i|$, and we denote $F_j^i + n$ the corresponding delivery point of F_j^i. Let us consider the Hamiltonian path of Fig. 5, built on sequences related to each color class. This Hamiltonian path trivially satisfies every incompatibility in I and pickup-delivery precedences. It has exactly $\chi(I) - 1$ delivery-pickup arcs: $(F_1^i + n, F_1^{i+1})$. Therefore, it is a valid PDPI solution with cost equal to $\chi(I) - 1$.

$\Pi(G_I, I) \geq \chi(I) - 1$

Let us suppose $\Pi(G_I, I) < \chi(I) - 1$. Then, there exists a valid Hamiltonian path H which includes less than $\chi(I) - 1$ delivery-pickup arcs. We split H into $H_1 \dots H_{\Pi(G_I,I)+1}$ subpaths which are determined by the $\Pi(G_I, I)$ delivery-pickup arcs in H. Let us define a coloring of I by assigning color i to the pickup vertices in H_i. It is a valid $\Pi(G_I, I) + 1$-coloring of I since there are not two incompatible goods in each subpath. But, we suppose that $\Pi(G_I, I) < \chi(I) - 1$, which lead us to a contradiction. □

4. Mathematical formulations

We consider three integer linear formulations for this problem. The first one (**PDPInc1**) is based on the model proposed in Ruland and Rodin (1997) for the Pick and Delivery problem. Let us consider binary variable $x_{i,j}$ that values 1 if arc (i, j) is used along the route, 0 otherwise. Moreover, for any vertex set S, $X(S)$ is defined as $\sum_{i,j \in S} x_{i,j}$. With this notation and using these variables, PDPI can be formulated as the following integer program:

$$\text{minimize} \quad \sum_{(i,j)\in A_G} c_{i,j} x_{i,j}$$

$$\text{subject to} \quad \sum_{i|(i,j)\in A_G} x_{i,j} = 1 \quad \forall j \in V_G \setminus \{s\} \tag{1}$$

$$\sum_{j|(i,j)\in A_G} x_{i,j} = 1 \quad \forall i \in V_G \setminus \{t\} \tag{2}$$

$$X(S) \leq |S| - 1 \quad \forall S \subset V_G,\ 2 \leq |S| \tag{3}$$

$$X(S) \leq |S| - 2 \quad \forall i \in P, \forall S \subset (V_G \setminus \{s, i+n\}) \text{ such that } \{i, t\} \subseteq S \tag{4}$$

$$X(S) \leq |S| - 3 \quad \forall (i,j) \in E_I, \forall S \subset (V_G \setminus \{s, i+n, j+n\}) \text{ such that } \{i, j, t\} \subseteq S$$

$$x_{i,j} \in \{0, 1\} \quad (i, j) \in A_G \tag{5}$$

The objective function minimizes the total routing cost. Constraints (1) and (2) guarantee that each vertex is visited exactly once. Constraints (3) are the well known subtour elimination constraints (SEC), which exclude solutions with disconnected tours. Constraints (4) establish precedence between a pickup vertex and its delivery vertex (see Ruland and Rodin, 1997). Finally, constraints (5) are added to forbid paths with incompatibilities.

To show that the last constraints remove incompatibilities, suppose that there exists a path from s to t accomplishing constraints (1) to (5), where the incompatibility between i and j is violated. W.l.o.g, assume that i is visited before j. Consider S the set of vertices belonging to the subpath that starts in i and ends in j, plus t. Since the precedence between a pickup vertex and its delivery is ensured by constraints (4) and the incompatibility is violated, we conclude that $i + n, j + n \notin S$. Therefore, by (5), $X(S) \leq |S| - 3$. Being that j must not be the last vertex before t, the constraint forces the subpath to break in 3 and t is isolated. Then, the subpath that starts in i and ends in j is not a connected subgraph, therefore representing a contradiction.

We should see that these constraints do not remove feasible solutions. Let us consider a feasible solution to PDPI, where there exists S that does not satisfy (5). Given that (4) must be satisfied, we conclude that $X(S) = |S| - 2$. Therefore, there are two disconnected subpaths. Vertices i, j on one side and t on the other, must belong to different subpaths because $i + n, j + n \notin S$ and the precedence between a pick and its delivery must be satisfied. Then, i and j belongs to the same subpath where $i + n$ is not included and hence the incompatibility constraint is violated. A contradiction.

The second model (**PDPInc2**) also considers $x_{i,j}$ variables as well as $y_{i,j}$ binary variables which take value 1 iff vertex i is somewhere before vertex j in the solution. It is based on model proposed in Sarin et al. (2005) for *The Traveling Salesman Problem with Precedence Constraints*. Using these variables, PDPI can be formulated as the following integer program:

$$\text{minimize} \quad \sum_{(i,j)\in A_G} c_{i,j} x_{i,j}$$

$$\text{subject to} \quad \sum_{i|(i,j)\in A_G} x_{i,j} = 1 \quad \forall j \in V_G \setminus \{s\} \tag{6}$$

$$\sum_{j|(i,j)\in A_G} x_{i,j} = 1 \qquad \forall i \in V_G \setminus \{t\} \tag{7}$$

$$y_{i,j} + y_{j,i} = 1 \qquad \forall i, j \in V_G \setminus \{s,t\} \tag{8}$$

$$x_{j,i} + y_{i,j} + y_{j,k} + y_{k,i} \leq 2 \qquad \forall i, j, k \in V_G \setminus \{s,t\} \tag{9}$$

$$x_{i,j} \leq y_{i,j} \qquad \forall (i,j) \in V_G \tag{10}$$

$$y_{i,i+n} = 1 \qquad \forall i \in P \tag{11}$$

$$\begin{aligned} y_{i,j} &= y_{i+n,j} && \forall i, j \in P, (i,j) \in E_I \\ x_{i,j} &\in \{0,1\} && (i,j) \in A_G \\ y_{i,j} &\in \{0,1\} && i, j \in V_G \setminus \{s,t\} \end{aligned} \tag{12}$$

The objective function minimizes the total routing cost. Constraints (6) and (7) guarantee that each vertex is visited exactly once. Constraints (8) and (9) are the well known constraints which determine a linear ordering among the vertices. Note that (9) is reinforced as was suggested in Sarin et al. (2005). Constraints (10) establish the relationship between variable $x_{i,j}$ and $y_{i,j}$, meaning that if j is the successor of i, then i is somewhere before j in the route. Constraints (11) compel a delivery vertex to be on the route after its corresponding pickup vertex. Finally, constraints (12) manage the incompatibility between two pickup points: if j is after i, then the delivery vertex of i must be previously visited.

The last model, **ITVMi** (Inside Task Variable Model), is a new proposal based on the addition of new binary variables $z_{i,j}$ to the classical model for TSP introduced in Dantzig et al. (1954). Variable $z_{i,j}$ values 1 iff j is in the path from i to its delivery $i+n$.

Using these variables, PDPI can be formulated as the following integer program:

$$\text{minimize} \quad \sum_{(i,j)\in A_G} c_{i,j} x_{i,j}$$

$$\text{subject to} \quad \sum_{i|(i,j)\in A_G} x_{i,j} = 1 \qquad \forall j \in V_G \setminus \{s\} \tag{13}$$

$$\sum_{j|(i,j)\in A_G} x_{i,j} = 1 \qquad \forall i \in V_G \setminus \{t\} \tag{14}$$

$$X(S) \leq |S| - 1 \qquad \forall S \subset V_G,\ 2 \leq |S| \tag{15}$$

$$x_{i,j} \leq z_{i,j} \qquad \begin{aligned} &i \in P,\ j \in (P \cup D) \\ &j \neq i, i+n \end{aligned} \tag{16}$$

$$x_{s,j} \leq 1 - z_{i,j} \qquad i \in P,\ j \in P,\ j \neq i \tag{17}$$

$$x_{j,t} \leq 1 - z_{i,j} \qquad i \in P,\ j \in D,\ j \neq i+n \tag{18}$$

$$z_{i,j} + x_{j,k} + x_{k,j} \leq 1 + z_{i,k} \qquad \begin{aligned} &i \in P, j, k \in (P \cup D) \\ &j, k \neq i, i+n, j \neq k \end{aligned} \tag{19}$$

$$\begin{aligned} z_{i,j} &= z_{i,j+n} = 0 && i, j \in P, (i,j) \in E_I \\ x_{i,j} &\in \{0,1\} && (i,j) \in A_G \\ z_{i,j} &\in \{0,1\} && i \in P, j \in (P \cup D), j \neq i, i+n \end{aligned} \tag{20}$$

Objective function and constraints (13), (14), (15) come from model in Dantzig et al. (1954) for TSP. The next proposition proves that constraints (13) to (20) guarantee precedences between a pickup vertex and its delivery one and incompatibility constraints.

Proposition 2. *Consider H a Hamiltonian path, starting at s and ending in t. Constraints (13) to (20) are inconsistent if and only if some precedence between a pickup vertex and its delivery one is not satisfied or some incompatibility is not complied.*

Proof. Consider H a Hamiltonian path that satisfies all the constraints. Then, the following properties are satisfied for all $i \in P$ and $j \in (P \cup D)$.

1. $z_{i,j} = 0$ if j is placed before i along H.
   We prove it by induction on the distance between j and s.
   Base case: $x_{s,j} = 1$. Then, j is a pickup vertex and constraints (17) imply $z_{i,j} = 0$. Induction step: $x_{k,j} = 1$ for $k \neq s, i$. By inductive hypothesis $z_{i,k} = 0$, then (19) imply $z_{i,j} = 0$.
2. $z_{i,j} = 0$ if j is placed after $i+n$ along H.
   We prove it by induction on the distance between j and t.
   Base case: $x_{j,t} = 1$. Then, j is a delivery vertex and constraints (18) imply $z_{i,j} = 0$. Induction step: $x_{j,k} = 1$ for $k \neq t, i$. By inductive hypothesis $z_{i,k} = 0$, then (19) imply $z_{i,j} = 0$.
3. $z_{i,j} = 1$ if j is placed after i along H, provided that $i+n$ is not set between i and j.
   We proceed by induction on the distance from i to j. Base case: $x_{i,j} = 1$. Then, constraint (16) imply $z_{i,j} = 1$. Induction step: $x_{k,j} = 1$ $(k \neq i+n)$. By inductive hypothesis $z_{i,k} = 1$, then constraints (19) imply $z_{i,j} = 1$ (changing role between j and k)

Suppose that H does not satisfy a precedence between a pickup vertex i and its delivery $i+n$. By constraints (13), there exists a last vertex $j+n \neq i+n$ which immediately precedes t. Then $j+n$ is placed after $i+n$, after i and $i+n$ is not set between i and $j+n$. Therefore, the last two properties lead to a contradiction.

Suppose that H does not satisfy an incompatibility between i and j. By constraints (20), $z_{i,j} = z_{j,i} = 0$. But the last property leads to a contradiction.

Moreover, if H is a Hamiltonian path, starting at s and ending in t that satisfies all the precedences and the incompatibilities, then we set (x, z) such that $x_{i,j} = 1$ if i is the immediate predecessor of j, 0 otherwise and $z_{i,j} = 1$ if j is between i and $i+n$ and 0 otherwise. We must see that (x, z) satisfies constraints (13) to (20). Constraints (13) to (15) are satisfied since H is a Hamiltonian path, starting at s and ending in t.

If $x_{i,j} = 1$, then j is placed immediately after i. Moreover, H satisfies the precedence between i and $i+n$, then j is placed between i and $i+n$. We have set $z_{i,j} = 1$ in this case, so constraints (16) are satisfied.

If $x_{s,j} = 1$, then j is the first visited vertex and it is not between any pickup and its corresponding delivery. We set $z_{i,j} = 0$ in this case, so constraints (17) are satisfied. If $x_{j,t} = 1$, then j is the last visited vertex and it is not between any pickup and its delivery. We set $z_{i,j} = 0$ in this case, so constraints (18) are satisfied.

Suppose $z_{i,j} = 1$, $x_{j,k} = 1$. According to the rule to set values to these variables, it means that j is the immediate predecessor of k ($x_{j,k} = 1$, $x_{k,j} = 0$) and j is placed between i and $i+n$ ($z_{i,j} = 1$). Therefore, k is also placed between i and $i+n$ and $z_{i,k} = 1$. So, constraints (19) are satisfied.

The path fulfills the incompatibility requests; therefore, the values of z set to 1 correspond to compatible vertices. Otherwise they are set to 0. Hence constraints (20) are satisfied. □

5. Experimental evaluation of the models

The models are tested in the context of a Branch and Cut approach using CPLEX 12.7 on an Intel Core i7 7700, 3.60Ghz and 16

Table 1
Results on Set1 and $n = 10$ for B&C Cplex default based on the three models.

p	PDPInc1			PDPInc2			ITVMi		
%	UInst	t(s)	#B&Bn	UInst	t(s)	#B&Bn	UInst	t(s)	#B&Bn
10	1	669.96	56094.56	0	15.56	648.88	0	86.88	23489.04
25	7	3251.36	135016.08	0	15.68	592.28	0	37.08	12180.6
50	17	6296.40	137011.16	0	6.60	224.08	0	1.88	1155.2
70	19	6019.96	136854.40	0	2.60	95.88	0	0.00	109.24

Table 2
Results on Set2 and $n = 10$ for B&C Cplex default based on the three models.

p	PDPInc1			PDPInc2			ITVMi		
%	UInst	t(s)	#B&Bn	UInst	t(s)	#B&Bn	UInst	t(s)	#B&Bn
10	0	137.36	7353.28	0	1.56	35.84	0	1.76	619.32
25	0	1587.24	47019.60	0	2.28	61.92	0	1.40	694.20
50	0	3726.92	72955.64	0	1.84	50.60	0	0.08	169.56
70	1	4332.20	83968.00	0	1.16	40.08	0	0.00	53.68

Gb of RAM. Exponential inequalities in model **PDPInc1** are managed as cutting planes by solving maximum flow subproblems as in Dumitrescu et al. (2010). A time limit of 7200 seconds is imposed and all CPLEX solver parameters are at their default values.

We use two sets of random instances with $n = 10$ and $n = 15$ tasks (22 and 32 vertices, respectively). Set1 corresponds to 10 instances (5 for $n = 10$ and 5 for $n = 15$) where the points are randomly generated within a grid square of $[0, 1000] \times [0, 1000]$. The first n points are the pickup ones, the following n are the delivery ones, and the last two points correspond to the origin and the destination, respectively. The cost is the Euclidean distance between two locations. Instances of Set2 are also randomly generated as those of Set1, but costs are randomly generated in [1000; 1500].

Now, for every instance and every percentage p% in the set {10%, 25%, 50%, 70%}, we generated 5 different instances adding random incompatibility graphs with p% of density. In conclusion, we have built 400 instances.

Notice that the instances in Set1 are Euclidean, while those in Set2 are random cost instances. In the last ones, costs were generated at random in [1000; 1500] to control that they were not very dissimilar. According to our computational experience, instances with very dissimilar costs are easy to solve. As the models do not take advantage of Euclidean condition, it is worth testing them in both types of instances and experimenting if the algorithms differ in effectiveness.

Let us consider first the results in Tables 1 and 2 on instances with $n = 10$. We report the number of not solved instances within the time limit (UInst), average CPU time in seconds (t(s)) and average of the size of the search tree for each model (#B&Bn). Results correspond to the average across the corresponding 25 instances for each density.

The tables show the poor performance of model **PDPInc1**. This model is the only one that could not solve all the instances within the time limit. Moreover, it requires significantly greater CPU time than the other two models on solved instances. This is mainly due to the huge number of nodes in the search tree that must be explored to find the optimal solution.

Regarding **PDPInc2** and **ITVMi** models, there is not a clear winner. **ITVMi** performs better on instances with medium and high percentages of incompatibility while **PDPInc1** shows better efficiency on low percentage of incompatibility.

It can be observed that, according to the relationship between CPU time and number of nodes in the search tree, LP relaxations of model **ITVMi** are faster to solve than LP relaxations of model **PDPInc2**.

Table 3
Results on Set1 and $n = 15$ for B&C Cplex default on **PDPInc2** and **ITVMi**.

p	PDPInc2			ITVMi		
%	UInst	t(s)	#B&Bn	UInst	t(s)	#B&Bn
10	15	5340.56	22480.96	24	6970.32	239058.52
25	14	5507.60	23308.44	24	7143.40	299014.44
50	8	3885.56	18163.84	4	2373.64	207984.12
70	1	989.56	5665.36	0	21.80	5661.52

Table 4
Results on Set2 and $n = 15$ for B&C Cplex default on **PDPInc2** and **ITVMi**.

p	PDPInc2			ITVMi		
%	UInst	t(s)	#B&Bn	UInst	t(s)	#B&Bn
10	0	445.84	1583.08	5	2692.40	144086.08
25	0	687.20	2644.40	2	1891.40	145188.44
50	0	438.56	1905.36	0	63.76	9500.60
70	0	220.88	1088.84	0	2.96	860.76

Next, Tables 3 and 4 show results on instances with $n = 15$. Model **PDPInc1** was excluded from the comparison due to the poor performance on $n = 10$. As expected, the instances are more difficult to solve and both models fail many times. However, the conclusions are similar: the larger the density, the better the **ITVMi** performance.

One possible explanation for this behavior could be related to how density affects the models. Model **PDPInc2** and model **ITVMi** share variables $x_{i,j}$. In addition, **PDPInc2** uses $n(2n-1) + 2n(n-1)$ variables $y_{i,j}$ and **ITVMi** uses $2n(n-1)$ variables $z_{i,j}$. To represent incompatibility, **PDPInc2** includes constraints $y_{i,j} = y_{i+n,j}$ for all $(ij) \in A_I$ ($2|A_I|$ constraints) and **ITVMi** includes constraints $z_{i,j} = z_{i,j+n} = 0$ ($4|A_I|$ constraints). As the number of incompatibilities grows, the proportion of fixed variables in **ITVMi** over the total of variables is more significant than in **PDPInc2**. It substantially reduced the Branch and Bound tree. Since the **ITVMi** relaxations are faster than **PDPInc2** relaxations, the algorithm based on **ITVMi** outperforms the one based on **PDPInc2** as density increases.

Even though both models tend to be more effective in not Euclidean instances than in Euclidean instances, similar conclusions about their performance and how it is related to the percentage of incompatibility can be stated for both type of instances.

As we mentioned before, LP relaxations of **ITVMi** can be solved faster than LP relaxations of **PDPInc2**. To give a deeper insight into LP relaxations, onTables 5–8 we report average percentage gap

Table 5
Set1 $n = 10$ – LP relaxations of model PDPInc2 and ITVMi.

p	AvBestSol	PDPInc2		ITVMi	
%		Gap	t(s)	Gap	t(s)
10	5103.76	14.28	0.08	15.40	0.00
25	5605.36	13.83	0.20	16.78	0.00
50	6597.80	10.65	0.24	14.13	0.00
70	7103.88	6.14	0.44	9.28	0.00

Table 6
Set2 $n = 10$ – LP relaxations of model PDPInc2 and ITVMi.

p	AvBestSol	PDPInc2		ITVMi	
%		Gap	t(s)	Gap	t(s)
10	22342.52	0.47	0.00	0.68	0.00
25	22580.80	0.82	0.04	1.11	0.00
50	22976.04	0.83	0.04	1.15	0.00
70	23333.84	0.67	0.12	1.00	0.00

Table 7
Set1 $n = 15$ – LP relaxations of model PDPInc2 and ITVMi.

p	AvBestSol	PDPInc2		ITVMi	
%		Gap	t(s)	Gap	t(s)
10	6327.56	20.50	5.68	23.24	0.00
25	7210.64	22.71	7.24	26.71	0.00
50	8535.52	21.69	7.16	25.86	0.00
70	9295.92	14.53	7.68	19.04	0.00

Table 8
Set2 $n = 15$ – LP relaxations of model PDPInc2 and ITVMi.

p	AvBestSol	PDPInc2		ITVMi	
%		Gap	t(s)	Gap	t(s)
10	32177.64	0.76	5.08	0.93	0.00
25	32481.88	1.09	6.08	1.30	0.00
50	33028.68	1.29	6.36	1.59	0.00
70	33552.88	1.23	6.08	1.56	0.00

(Gap) and average resolution time of the initial relaxation (t(s)). Gap is calculated as $100(BestSol - z^*)/BestSol$ where z^* is the optimal value of the initial linear relaxation and *BestSol* the best known solution for the instance. Moreover, we also report average on best known solutions (AvBestSol).

It can observed that model **PDPInc2** has a lower percentage gap than **ITVMi**; however, it is much more time-consuming.

Considering these observations, we think that it would be worthwhile to strengthen the relaxation of model **ITVMi** with cutting planes. In this way, we hope to reduce relaxation gaps and the size of the search tree without increasing too much the time required for solving the relaxations in each node and to develop a competitive algorithm.

The next section is devoted to the study of P_{ITVMi}, the polytope associated to model **ITVMi** (convex hull of all integer feasible solutions), to infer valid inequalities that could be useful as cutting planes to improve the overall performance of a Branch and Cut algorithm.

6. Polyhedral analysis

The aim of this section is to present polyhedral properties of P_{ITVMi}. Notice that, for all $(i, j) \in E_I$, variables $x_{i,j}, x_{j,i}, x_{i,j+n}, x_{j,i+n}, x_{j+n,i+n}, x_{i+n,j+n}$ and $z_{i,j}, z_{j,i}, z_{i,j+n}, z_{j,i+n}$ are set to 0 and therefore they can be removed from the formulation.

6.1. Minimal equation system

The first step in our polyhedral study is the characterization of the minimal equation system for P_{ITVMi}.

Proposition 3. *Consider $i \in P$, $N_I(i) = \{j \in P : (i, j) \in E_I\}$ and $d_I(i) = |N_I(i)|$. The following equations constitute the minimal equation system of P_{ITVMi}:*

$$\sum_{i|(i,j)\in A_G} x_{i,j} = 1 \qquad \forall j \in V_G \setminus \{s\} \tag{21}$$

$$\sum_{j|(i,j)\in A_G} x_{i,j} = 1 \qquad \forall i \in V_G \setminus \{s,t\},\ i \notin \{k : d_I(k) = n-1\} \tag{22}$$

$$z_{i,j} + z_{j,i} = z_{i,j+n} + z_{j,i+n} \qquad \forall (i,j) \notin E_I,\ i < j \tag{23}$$

$$x_{i,j} = z_{i,j} \qquad \forall (i,j) \notin E_I,\ d_I(i) = n-2 \tag{24}$$

$$x_{j+n,i+n} = z_{i,j+n} \qquad \forall (i,j) \notin E_I,\ d_I(i) = n-2,\ (d_I(j) < n-2 \vee i < j) \tag{25}$$

$$z_{i,j} + x_{i,j+n} = z_{i,j+n} + x_{j,i+n} \qquad \forall (i,j) \notin E_I,\ i < j,\ d_I(i) < n-2,\ d_I(j) < n-2,\ N_I(i) \cup N_I(j) = P \setminus \{i,j\} \tag{26}$$

$$x_{i,j} + z_{i,k+n} = x_{k+n,i+n} + z_{i,j} \qquad \forall (j,k) \in E_I,\ j < k,\ N_I(i) = P \setminus \{i,j,k\} \tag{27}$$

Proof. We will focus on showing that all feasible solutions satisfy the equations. Details of proof this is a minimal equation system can be seen in Factorovich (2019).

By definition of ITVMi model, it follows that Eqs. (21) and (22) are valid.

Let us consider $i, j \in V_G \setminus \{s, t\}$, with $(i, j) \notin E_I$. In all feasible Hamiltonian paths, it happens one of the following configurations:

1. $s \rightsquigarrow i \rightsquigarrow j \rightsquigarrow i+n \rightsquigarrow j+n \rightsquigarrow t$ $(z_{i,j} = z_{j,i+n} = 1;\ z_{j,i} = z_{i,j+n} = 0)$
2. $s \rightsquigarrow i \rightsquigarrow j \rightsquigarrow j+n \rightsquigarrow i+n \rightsquigarrow t$ $(z_{i,j} = z_{i,j+n} = 1;\ z_{j,i} = z_{j,i+n} = 0)$
3. $s \rightsquigarrow j \rightsquigarrow i \rightsquigarrow j+n \rightsquigarrow i+n \rightsquigarrow t$ $(z_{j,i} = z_{i,j+n} = 1;\ z_{i,j} = z_{j,i+n} = 0)$
4. $s \rightsquigarrow j \rightsquigarrow i \rightsquigarrow i+n \rightsquigarrow j+n \rightsquigarrow t$ $(z_{j,i} = z_{j,i+n} = 1;\ z_{i,j} = z_{i,j+n} = 0)$
5. $s \rightsquigarrow j \rightsquigarrow j+n \rightsquigarrow i \rightsquigarrow i+n \rightsquigarrow t$ $(z_{i,j} = z_{j,i+n} = z_{j,i} = z_{i,j+n} = 0)$
6. $s \rightsquigarrow i \rightsquigarrow i+n \rightsquigarrow j \rightsquigarrow j+n \rightsquigarrow t$ $(z_{i,j} = z_{j,i+n} = z_{j,i} = z_{i,j+n} = 0)$

Considering all possible configurations, it is easy to see that (23) are valid for all feasible Hamiltonian path.

If $d_I(i) = n-2$, validity of (24) and (25) follows from the fact that the only vertices that could be on the way form i to $i+n$ are j and/or $j+n$.

Eqs. (26) are trivially satisfied for all Hamiltonian paths with configuration 2, 4, 5 or 6. Moreover, if $N_I(i) \cup N_I(j) = P \setminus \{i, j\}$, we can assure that in all feasible solution with configuration 1 or 3, there are no vertices on the way from j to $i+n$ or from i to $j+n$, respectively. Then, Eqs. (26) are valid.

Let us finally analyze Eqs. (27). In all feasible Hamiltonian path, it takes place one of the following possible situations:

1. neither j nor $k+n$ is on the way from i to $i+n$ $(z_{i,j} = z_{i,k+n} = x_{i,j} = x_{k+n,i+n} = 0)$
2. j is placed on the way from i to $i+n$ and $k+n$ is not on the way from i to $i+n$ $(z_{i,j} = 1, z_{i,k+n} = 0$ and $x_{k+n,i+n} = 0)$

3. $k+n$ is placed on the way from i to $i+n$ and j is not on the way from i to $i+n$ ($z_{i,k+n}=1, z_{i,j}=0$ and $x_{i,j}=0$)
4. both j and $k+n$ are on the way from i to $i+n$ ($z_{i,j}=z_{i,k+n}=1$).

Case 1 makes Eqs. (27) trivially satisfied. To prove the validity of Eqs. (27) for case 2, we have to see that the immediate successor of i must be j. Given that $N_I(i)=P\setminus\{i,j,k\}$, the immediate successor could be k or j. If k were the immediate successor, then j and k would be in the vehicle at the same time, which is not feasible since j and k are incompatible. Similar arguments can be applied in case 3 by analyzing the possible immediate predecessors of $i+n$.

Finally, in case 4, the only two feasible paths are:

$s \rightsquigarrow i \rightsquigarrow k \rightsquigarrow k+n \rightsquigarrow j \rightsquigarrow j+n \rightsquigarrow i+n \rightsquigarrow t$
$(z_{i,k+n}=z_{i,j}=1;\ x_{i,j}=x_{k+n,i+n}=0)$

$s \rightsquigarrow i \rightsquigarrow j \rightsquigarrow j+n \rightsquigarrow k \rightsquigarrow k+n \rightsquigarrow i+n \rightsquigarrow t \qquad (z_{i,j}=z_{i,k+n}=1)$.
Since $N_I(i)=P\setminus\{i,j,k\}$, then no other vertex can be on the way from i to $i+n$. This implies that $x_{i,j}=x_{k+n,i+n}=1$.

In both cases, it is easy to see that Eqs. (27) are valid. □

Corollary 1. *The dimension of P_{ITVMi} is $\frac{11}{2}n^2-\frac{13}{2}n-1-9|E_I|+n_1-n_2-n_3-n_4-n_5$ where n_1 is the number of universal vertices in V_I, n_2 the number of vertices in V_I with degree equal to $n-2$, n_3 the number of vertices in V_I with degree equal to $n-2$ and its unique compatible vertex satisfies (25), n_4 the number of pairs of compatible vertices in V_I that verify (26) and n_5 the number of triples in V_I that comply condition (27).*

Proof. The model has $4n^2-n-6|E_I|$ variables $x_{i,j}$ and $2n^2-2n-4|E_I|$ variables $z_{i,j}$. There are $2n+1$ equations in (21), $2n-n_1$ in (22), $\frac{n(n-1)}{2}-|E_I|$ in (23), n_2 in (24), n_3 in (25), n_4 in (26) and n_5 in (27). Therefore,

$$\begin{aligned}\dim(P_{ITVMi}) &= 6n^2-3n-10|E_I| - \text{range of the minimal equation system}\\ &= 6n^2-3n-10|E_I| \\ &\quad -(2n+1)-(2n-n_1)-\left(\frac{n(n-1)}{2}-|E_I|\right)\\ &\quad -n_2-n_3-n_4-n_5\\ &= 6n^2-3n-10|E_I| \\ &\quad -\frac{1}{2}n^2-2n-2n+\frac{1}{2}n-1+n_1+|E_I|-n_2-n_3-n_4-n_5\\ &= \frac{11}{2}n^2-\frac{13}{2}n-1-9|E_I|+n_1-n_2-n_3-n_4-n_5\end{aligned}$$

□

6.2. Polynomial-size valid inequalities

Next we present several polynomial-size families of valid inequalities. The first group focuses on the relationship between two pickup points and their corresponding delivery points.

Proposition 4. *The following inequalities are valid for P_{ITVMi}:*

$$x_{i,i+n}+x_{i,j+n}+z_{i,j}\le 1 \quad \forall i,j\in P, j\neq i, (i,j)\notin E_I \tag{28}$$

$$x_{i,i+n}+x_{j,i+n}+z_{i,j+n}\le 1 \quad \forall i,j\in P, i\neq j, (i,j)\notin E_I \tag{29}$$

$$z_{i,j}+z_{j,i}+x_{j+n,i}+x_{i+n,j}\le 1 \quad \forall i,j\in P, j\neq i, (i,j)\notin E_I \tag{30}$$

$$x_{j+n,i}+x_{i+n,j}\le 1 \quad \forall i,j\in P, j\neq i, (i,j)\in E_I \tag{31}$$

$$x_{j,i+n}\le z_{i,j} \quad \forall i\in P, j\in P\cup D\setminus\{i,i+n\} \tag{32}$$

Proof. Let us see one by one:

Inequality (28): By (22), $x_{i,i+n}+x_{i,j+n}\le 1$. If $x_{i,i+n}=1$ then there are not any pickup vertices between i and $i+n$. Therefore, $z_{i,j}=0, \forall j\in P$ and $x_{i,i+n}+z_{i,j}\le 1$. If $x_{i,j+n}=1$ then, by the precedence constraint between any pickup vertex and its delivery one, we can conclude that $z_{i,j}=0$. Then, $x_{i,j+n}+z_{i,j}\le 1$.

Inequality (29): The proof runs on similar arguments to those used before using that $x_{i,i+n}+x_{j,i+n}\le 1$ by (21).

Inequality (30): We can state that $z_{i,j}+z_{j,i}\le 1$ since j can not be simultaneously placed before and after i. Moreover $x_{j+n,i}+x_{i+n,j}\le 1$ by the precedence constraint between a pickup vertex and its delivery one. Now, the analysis continues like in (28).

Inequality (31): It follows from the precedence constraint between a pickup vertex and its delivery one.

Inequality (32): If $i+n$ follows j, from the precedence between i and $i+n$, then j is placed between i and $i+n$.

□

The second group is based on the transitive property that follows from the order that three compatible pickup points are visited along a Hamitonian path.

Proposition 5. *The following inequalities are valid for P_{ITVMi}:*

$$z_{i,j}+z_{j,k}+z_{k,i}+x_{i,k}+x_{i+n,j}+x_{j+n,k}+x_{k+n,i}\le 2$$
$$\forall i,j,k\in P, i\neq j\neq k, (i,j),(j,k),(k,i)\notin E_I \tag{33}$$

$$z_{i,j}+z_{j,k}+z_{k,i}+x_{j,i}+x_{i+n,j}+x_{j+n,k}+x_{k+n,i}\le 2$$
$$\forall i,j,k\in P, i\neq j\neq k, (i,j),(j,k),(k,i)\notin E_I \tag{34}$$

$$z_{i,j}+z_{j,k}+z_{k,i}+x_{k,j}+x_{i+n,j}+x_{j+n,k}+x_{k+n,i}\le 2$$
$$\forall i,j,k\in P, i\neq j\neq k, (i,j),(j,k),(k,i)\notin E_I \tag{35}$$

Proof. Let us analize Inequality (33). By transitivity, if i goes before j and j goes before k, then k can not be placed before i. Then $z_{i,j}+z_{j,k}+z_{k,i}\le 2$. With similar arguments and the precedence constraints between each pickup vertex and its delivery one, we conclude that $x_{i+n,j}+x_{j+n,k}+x_{k+n,i}\le 2$. Moreover, $z_{i,j}+x_{i+n,j}\le 1$ since j can not be simultaneously placed before and after $i+n$. Similarly, $z_{j,k}+x_{j+n,k}\le 1$ and $z_{k,i}+x_{k+n,i}\le 1$. If $z_{i,j}+z_{j,k}=2$, then $x_{i+n,j}=x_{j+n,k}=0$ and $k+n$ is not a predecessor of i, then $x_{k+n,i}=0$. Therefore, the inequality $z_{i,j}+z_{j,k}+z_{k,i}+x_{i+n,j}+x_{j+n,k}+x_{k+n,i}\le 2$ is valid. If $x_{i+n,j}+x_{j+n,k}=2$, then $z_{i,j}=z_{j,k}=0$ and i is not placed before $k+n$ then $z_{k,i}=0$. Then, the inequality $z_{i,j}+z_{j,k}+z_{k,i}+x_{i+n,j}+x_{j+n,k}+x_{k+n,i}\le 2$ is valid. Other cases are symmetric. If $x_{i,k}=0$, it follows that the inequality (33) is valid. If $x_{i,k}=1$, then $z_{k,i}=x_{k+n,i}=x_{j+n,k}=0$ and $z_{i,j}+z_{j,k}+x_{i+n,j}\le 1$. It implies that the inequality (33) is valid.

Validity of Inequalities (34) and (35) runs on similar arguments. □

6.3. Exponential-size valid inequalities

Let us consider now exponential-size inequality families. The first two are based on properties of the incompatibility undirected graph I.

6.3.1. Coloring incompatibility subtour inequality

The following inequality stands on the minimum number of sub-paths in which an induced Hamiltonian path is split. It has a strong connection with the relationship we established between PDPI and the vertex coloring problem in Section 3.

Given $S\subset P\cup D$, we define $\Pi(S)=\{i\in P: i+n\in S\}$ and $\mathrm{P}(S)=(S\setminus\Pi(S))\cap P$ (set of pickup vertices in S such that $i+n\notin S$).

Proposition 6. *Let us consider* $S \subseteq V_G \setminus \{s\}$, $t \in S$, $|E_{I_{[P(S)]}}| \geq 1$, *where* $I_{[P(S)]}$ *is the induced graph whose vertex set is* $P(S)$. *The following inequality is valid for* P_{ITVMi}:

$$X(S) \leq |S| - \chi(I_{[P(S)]}) - 1 \tag{36}$$

where $\chi(I_{[P(S)]})$ *is the chromatic number of the induced subgraph* $I_{[P(S)]}$. *We name it* ***Coloring incompatibility subtour inequality***.

Proof. Let H be a Hamiltonian path and suppose that there exists S such that $X(S) > |S| - \chi(I_{[P(S)]}) - 1$. Then, the induced graph $H_{[S]}$ has at most $\chi(I_{[P(S)]})$ disjunctive subpaths. Given that for all $v \in P(S)$ it is satisfied that $v + n \notin S$, then we can assure that there are no $v \in P(S)$ that belong to the same subpath that t. Therefore, in any $\chi(I_{[P(S)]})$ coloring of $I_{[P(S)]}$, there are two incompatible vertices p and q that belong to the same subpath which are assigned different colors. Nevertheless, by definition of $P(S)$, we know that $\{p+n, q+n\} \cap S = \emptyset$, then p and q are in the vehicle at the same time along H. A contradiction holds. □

6.3.2. Independent set incompatibility inequalities

The following inequality is based on the maximum number of a given set of pickup points or their corresponding delivery points that could be present at the same time in the vehicle.

Proposition 7. *Consider* $i \in P$ *and* $S \subseteq P \setminus (N_I(i) \cup \{i\})$. *The following inequalities are valid for* P_{ITVMi}:

$$\sum_{k \in S} (x_{i,k} + z_{k,i}) \leq \alpha(I_{[S]}) \tag{37}$$

$$\sum_{k \in S} (x_{i+n,k} + z_{k,i+n}) \leq \alpha(I_{[S]}) \tag{38}$$

$$\sum_{k \in S} (x_{k+n,i} + z_{k,i}) \leq \alpha(I_{[S]}) \tag{39}$$

$$\sum_{k \in S} (x_{k+n,i+n} + z_{k,i+n}) \leq \alpha(I_{[S]}) \tag{40}$$

where $\alpha(I_{[S]})$ *is the stability number of the induced subgraph* $I_{[S]}$. *We name them* ***Independent set incompatibility inequalities***.

Proof. The number of pickup vertices in S that could be in the vehicle at the same time is at most $\alpha(I_{[S]})$. Then, $\sum_{k \in S} z_{k,i} \leq \alpha(I_{[S]})$. Moreover, (22) implies that $\sum_{k \in S} x_{i,k}$ and $\sum_{k \in S} x_{i+n,k}$ are ≤ 1 and (21) implies that $\sum_{k \in S} x_{k+n,i}$ and $\sum_{k \in S} x_{k+n,i+n}$ are ≤ 1. Let us analyze the inequality when they are combined.

- **Inequality** (37): If $x_{i,k_1} = 1$, then $z_{k_1,i} = 0$ and the pickup vertices in S such that i could be placed between them and their deliveries must be compatible with k_1. So, $\Sigma_{k \in S} z_{k,i}$ is lower than or equal to $\alpha(I_{[S]}) - 1$ and the inequality is valid.
- **Inequality** (38): The argument is similar to the previous one, but based on $i+n$ position instead of i position.
- **Inequality** (39): If $x_{k_1+n,i} = 1$, by the precedence constraint between k_1 and $k_1 + n$, the pickup vertices in S that go through i before visiting their deliveries must be compatible with k_1. So, $\sum_{k \in S} z_{k,i}$ is lower than or equal to $\alpha(I_{[S]}) - 1$ and the inequality is valid.
- **Inequality** (40): The argument is similar to the previous one, but based on $i+n$ position instead of i position. □

6.3.3. 2-Generalized Order constraints

The following inequalities are a generalization of the generalized order constraints presented in Ruland and Rodin (1997).

Proposition 8. *Consider* $S_1 \subset V_G$, $S_2 \subset V_G$ *such that there exist* $i, j \in P$ *with* $\{i, j+n\} \subseteq S_1$, $\{i, j+n\} \cap S_2 = \emptyset$, $\{i+n, j\} \subseteq S_2$, $\{i+n, j\} \cap S_1 = \emptyset$. *The following inequality is valid for* P_{ITVMi}:

$$X(S_1) + X(S_2) \leq |S_1| + |S_2| - 3 \tag{41}$$

We name them ***2-Generalized Order constraints***.

Proof. If $S_1 \cap S_2 = \emptyset$ the inequality reduces to the generalized order constraints with $m = 2$ presented in Ruland and Rodin (1997). Suppose that $T = S_1 \cap S_2$ with $T \neq \emptyset$. Without loss of generality, we assume that

$$X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \leq X(T : S_1 \setminus T) + X(S_1 \setminus T : T)$$

where $X(Q_1 : Q_2)$ is defined as $\sum_{\substack{i \in Q_1 \\ j \in Q_2}} x_{i,j}$ for any two vertex sets.

Then,

$$\begin{aligned} X(S_1) + X(S_2) = {} & X(S_1) + X(S_2 \setminus T) + X(T) + X(T : S_2 \setminus T) \\ & + X(S_2 \setminus T : T) \end{aligned}$$

Since $S_1 \cap (S_2 \setminus T) = \emptyset$,we know that the generalized order constraint introduced in Ruland and Rodin (1997) holds for S_1 and $S_2 \setminus T$. Then

$$\begin{aligned} & X(S_1) + X(S_2 \setminus T) + X(T) + X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \\ & \leq |S_1| + |S_2 \setminus T| - 3 + X(T) + X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \\ & = |S_1| + |S_2| - |T| - 3 + X(T) + X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \end{aligned}$$

By considering (13) and (14), we know that:

$$\begin{aligned} |T| + |T| & \geq \sum_{i \in T} \sum_{j} x_{i,j} + \sum_{i \in T} \sum_{j} x_{j,i} \\ & = \sum_{i \in T} \left(\sum_{j \in T} x_{i,j} + \sum_{j \in S_2 \setminus T} x_{i,j} + \sum_{j \in S_1 \setminus T} x_{i,j} + \sum_{j \notin S_1 \cup S_2} x_{i,j} \right) \\ & \quad + \sum_{i \in T} \left(\sum_{j \in T} x_{j,i} + \sum_{j \in S_2 \setminus T} x_{j,i} + \sum_{j \in S_1 \setminus T} x_{j,i} + \sum_{j \notin S_1 \cup S_2} x_{j,i} \right) \\ & \geq 2X(T) + X(T : S_2 \setminus T) + X(T : S_1 \setminus T) \\ & \quad + X(S_2 \setminus T : T) + X(S_1 \setminus T : T) \end{aligned}$$

Then,

$$X(T) + X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \leq |T|$$

or

$$X(T) + X(T : S_1 \setminus T) + X(S_1 \setminus T : T) \leq |T|$$

Since, w.l.o.g, we assume that

$$X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \leq X(T : S_1 \setminus T) + X(S_1 \setminus T : T),$$

it follows that

$$\begin{aligned} X(S_1) + X(S_2) = {} & |S_1| + |S_2| - |T| - 3 + X(T) \\ & + X(T : S_2 \setminus T) + X(S_2 \setminus T : T) \\ \leq {} & |S_1| + |S_2| - |T| - 3 + |T| = |S_1| + |S_2| - 3 \end{aligned}$$

□

6.3.4. Reinforced subtour elimination constraint

Next we present two valid inequalities that arise from strengthening the subtour elimination constraint.

Proposition 9. *Consider $S \subset V_G$ and $H = \{i_1, \dots, i_m\} \subset P$, such that $\{i_1, i_1+n, \dots i_m, i_m+n\} \subseteq S$ and $j \in (P \cup D) \backslash S$. The following inequalities are valid for P_{ITVMi}:*

$$X(S) + \sum_{a=1}^{m} z_{i_a,j} \leq |S| + \alpha(H) - 2 \tag{42}$$

*If $\alpha(H) = 1$, the inequality reinforces the corresponding subtour elimination constraint. For this particular case, we name them **Reinforced-1 subtour elimination constraint**.*

Proof. By incompatibility constraints, we know that $\sum_{a=1}^{m} z_{i_a,j} \leq \alpha(H)$. By subtour constraint, $X(S) \leq |S| - 1$ is valid. If $X(S) \leq |S| - 2$, the inequality is trivially valid. If $X(S) = |S| - 1$, then there is a simple path including all vertices in S and does not include j. Therefore $\sum_{a=1}^{m} z_{i_a,j} = 0$ and the inequality follows. □

Proposition 10. *Consider $S \subset V_G$. The following inequalities are valid for P_{ITVMi}:*

$$X(S) + \sum_{\substack{i \in S \\ i+n \in S}} \sum_{\substack{j \in S \\ j+n \notin S}} x_{i,j+n} \leq |S| - 1 \tag{43}$$

*We name them **Reinforced-2 subtour elimination constraint**.*

Proof. By subtour constraint, we know that $X(S) \leq |S| - 1$. If $X(S) = |S| - 1$ then there is a simple path including all vertices in S and only the last vertex in the path can be followed by a vertex not in S. Therefore, $\sum_{i, i+n \in S} \sum_{j \in S, j+n \notin S} x_{i,j+n} = 0$ otherwise the precedence between the last vertex and its delivery is not satisfied. In other cases, if there are k subpaths including vertices in S then $X(S) = |S| - k$ and only from the last vertex of the first $k-1$ subpaths can follow a vertex not in S. Therefore, the inequality is valid. □

This inequality is a reinforced version of *LSEC* inequality (Lifted subtour elimination constraint) that was introduced for The Traveling Salesman Problem with Pickup and Delivery in Dumitrescu et al. (2010).

6.4. Other inequalities

Moreover, all inequalities introduced in Dumitrescu et al. (2010) are also valid for P_{ITVMi}. We consider those that, according to the reported computational experience, show good performance toward closing the integrality gap:

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

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

  is valid for P_{ITVMi}.
- **Π and σ Inequalities**.
  For any $S \subset V \backslash \{t\}$, the Π-inequality

  $$x(S \setminus \Pi(S) : \bar{S} \setminus \Pi(S)) \geq 1$$

  is valid for P_{ITVMi}
  σ-inequalities are obtained by exchanging the roles of pickup and delivery vertices in the Π-inequalities.
- **Precedence constraint (*PC*)** These inequalities come from the original model proposed in Ruland and Rodin (1997) for the Pick and Delivery problem we describe in Section 3. Let $S \subset (V_G \setminus \{s, i+n\})$ such that $\{i, t\} \subseteq S$, then

  $$X(S) \leq |S| - 2$$

  is valid for P_{ITVMi}.

7. Solution methodology

Next, we describe the different components we propose to include in a Branch and Cut algorithm.

7.1. Separation algorithms

Given a fractional LP relaxation solution in a node of the search tree, separation algorithms try to find valid inequalities violated by the actual fractional solution in order to be added and therefore strengthen the LP relaxation.

Polynomial size families are tackled through direct enumeration.

Next we describe the algorithm for each exponential-size family.

- **Coloring incompatibility subtour inequality** (36)
  We separate these inequalities by using a heuristic procedure. First, we find 5-holes (C_5) and cliques of 2, 3 and 4 vertices (K_2, K_3, K_4) in the incompatibility graph I. Each of these structures is used to be extended to a candidate set S. In this way, we can assure that $\chi(I_{[P(S)]})$ is greater than 2 and, in most of cases, close to $|S|$ which gives more chances to find a violated inequality. The extension of each structure is made by solving a minimum cut problem of V_G enforcing that t belongs to one of the two partition sets, and s and the corresponding deliveries of vertices in the structure, to the other partition set. Finally, the chromatic number of $I_{[P(S)]}$ is calculated for the candidate set S.
  It is worth mentioning that the size of the instances that we consider allows us to address the computation of χ exactly through DSATUR algorithm (Brélaz, 1979). In Algorithm 1 we present the sketch of the procedure.

Algorithm 1 Separation algorithm of Inequalities (36).

Input: X^* fractional solution of LP relaxation
Output: A set C of inequalities (36) which are violated by X^*

```
Find 5-hole, K2, K3, K4 subgraphs of I
Structures = set of C5, K2, K3, K4 subgraphs of I
Set C = ∅
for estr ∈ Structures do
    Set S_estr = {i ∈ P : i ∈ estr} ∪ {t} and T = {i + n : i ∈ estr} ∪ {s}
    S = minimum cut such that S_estr ⊂ S and S ∩ T = ∅
    Compute χ(I_[P(S)])
    if Σ_{(i,j)∈S} x*_{i,j} > |S| − χ(I_[P(S)]) − 1 then
        Include Σ_{i,j∈S} x_{i,j} ≤ |S| − χ(I_[P(S)]) − 1 in C
    end if
end for
return C
```

- **Independent incompatibility inequality** (37)
  We develop separation algorithms for two particular cases.
  The first one considers S as a complete subgraph. Therefore, $\alpha(I_{[S]}) = 1$. We use a heuristic approach which looks for cliques by applying a greedy procedure. For each $i \in P$, we find, among

the compatible pickup vertices, the one which maximizes $x^*_{i,k} + z^*_{i,k}$ and we initialize a clique with it. Then, the clique is sequentially increased into a bigger clique in I looking for another adjacent vertex with $x^*_{i,k} + z^*_{i,k}$ maximum value. The procedure makes several trials limited by an input parameter (maxClique). In Algorithm 2 we present the sketch of the procedure. The second separation algorithm deals with a particular case where $\alpha(I_{[S]}) = 2$. Given $i \in P$ and starting from S equal to a 5-hole (C_5) compatible with i, S is extended by adding vertices without increasing $\alpha(I_{[S]}) = 2$. For each vertex v in C_5, the algorithm initializes two complete subgraphs K_1, K_2 with the other four vertices in C_5. Then the algorithm attempts to extend each complete subgraph with vertices which are compatible with i and incompatible with v or incompatible with all vertices in the other complete subgraph. Formally, if $j \in K_1$ then $j \in N_I(v) \cup \cap_{k \in K_2} N_I(k) \setminus N_I(i)$ and $j \in K_2$ then $j \in N_I(v) \cup \cap_{k \in K_1} N_I(k) \setminus N_I(i)$. We consider $S = \{v\} \cup K_1 \cup K_2$, for which is guaranteed that $\alpha(I_{[S]}) = 2$. In Algorithm 3 we present the sketch of the procedure.

Algorithm 2 Separation algorithm of Inequalities (37) for $\alpha(I_{[S]}) = 1$.

Input: (x^*, z^*) fractional solution of LP relaxation
Output: A set C of inequalities (37) which are violated by X^*

Set $C = \emptyset$
for $i \in P$ **do**
  for h=1 to maxClique **do**
    Set $Candidate[h]$ = $\underset{\substack{j:(i,j)\notin E_I \\ j \neq Candidate[h'] \ \forall h' < h}}{\arg\max} (x^*_{i,j} + z^*_{j,i})$
    Initialize $Clique$ with $Candidate[h]$
    while $M > 0$ **do**
      Set $k = \underset{\substack{j:(i,j)\notin E_I; Clique \subseteq \Gamma(j) \\ j \neq Candidate[h'] \ \forall h' < h}}{\arg\max} (x^*_{i,j} + z^*_{j,i})$
      $M = x^*_{i,k} + z^*_{k,i}$
      if $M > 0$ **then**
        Include k in $Clique$
      end if
    end while
    if $\sum_{k \in Clique_i} (x^*_{i,k} + z^*_{k,i}) > 1$ **then**
      Include $\sum_{k \in Clique_i} (x_{i,k} + z_{k,i}) \leq 1$ in C
    end if
  end for
end for
Return C

Algorithm 3 Separation algorithm of Inequalities (37) with $\alpha(I_{[S]}) = 2$.

Input: (x^*, z^*) fractional solution of LP relaxation
Output: A set C of inequalities (37) which are violated by X^*.

Find 5-holes of I
$Cycles$ = set of C_5
for $i \in P \cup D$ **do**
  for $cycle \in Cycles$ **do**
    if $cycle \cap N_I(i) = \emptyset$ **then**
      for $v \in cycle$ **do**
        $\{k_1, k_2\} = N_I(v) \cap cycle$
        $K_1 = \{k_1\} \cup \{k \in cycle, k \in N_I(v) \cap N_I(k_1)\}$
        $K_2 = \{k_2\} \cup \{k \in cycle, k \in N_I(v) \cap N_I(k_2)\}$
        for $j \in P \setminus cycle$, $j \notin N_I(i)$ **do**
          if $K_1 \subset N_I(j)$ **then**
            Add j to K_1
          else
            if $K_2 \subset N_I(j)$ **then**
              Add j to K_2
            end if
          end if
        end for
        repeat
          Change = FALSE
          for $j \in K_1$ **do**
            if $j \notin N_I(v)$, $j \notin cycle$, and exists $u \in K_2$ such that $j \notin N_I(u)$ **then**
              Remove j from K_1
              Change = TRUE
            end if
          end for
          for $j \in K_2$ **do**
            if $j \notin N_I(v)$, $j \notin cycle$, and exists $u \in K_1$ such that $j \notin N_I(u)$ **then**
              Remove j from K_2
              Change = TRUE
            end if
          end for
        until Change=FALSE
        if $\sum_{k \in (\{v\} \cup K_1 \cup K_2)} x^*_{i,k} + z^*_{k,i} > 2$ **then**
          Include $\sum_{k \in (\{v\} \cup K_1 \cup K_2)} x^*_{i,k} + z^*_{k,i} \leq 2$ in C
        end if
      end for
    end if
  end for
end for
Return C

- **Independent incompatibility inequalities** (38), (39) and (40).
  The procedure is similar to the one above in the way the clique is built, just replacing the value $x^*_{i,k} + z^*_{i,k}$ by $x^*_{i+n,k} + z^*_{k,i+n}$, $x^*_{k+n,i} + z^*_{k,i}$ or $x^*_{k+n,i+n} + z^*_{k,i+n}$, respectively.
- **Generalized Order inequality** (41)
  These inequalities are separated by the algorithm proposed in Dumitrescu et al. (2010), but we do not require that $S_1 \cap S_2 = \emptyset$.
- **Reinforced-1 subtour elimination inequality** (42)
  We consider these inequalities for the particular case that H is a complete subgraph. Therefore, $\alpha(H) = 1$.
  When a subtour elimination constraint is found, for each $j \in (P \cup D) \setminus S$, a complete subgraph K in I compatible with j, such that vertices in K and their corresponding deliveries belong to S, is searched for in a greedy way following a decreasing order of $z^*_{i,j}$. The subtour elimination inequality is reinforced with $\sum_{i \in K} z_{k,j}$.
- **Reinforced-2 subtour elimination inequality** (43)
  When a subtour elimination constraint is found, the inequality is reinforced by adding $\sum_{i \in S} \sum_{j+n \notin S} x_{i,j+n}$.
- **GLSEC,** PC, Π and σ **inequalities**
  These inequalities are separated by the algorithms proposed in Dumitrescu et al. (2010).

7.2. Primal heuristic

After solving the LP relaxation at a node of the search tree, we apply a heuristic procedure which tries to use the fractional solution of the LP relaxation to accomplish an improved integer solu-

tion. The primal heuristic consists of a greedy constructive heuristic followed by a local search improvement phase.

7.2.1. Initial solution: a constructive heuristic

The algorithm builds a path from s to t by incrementally inserting a pickup vertex and its corresponding delivery. Our approach follows a greedy strategy. Given (x^*, z^*) a fractional solution, for each pickup vertex i, we calculate $\sum_{j \in P \cup D, j \neq i} z^*_{i,j}$ value and sort the pickup vertices in decreasing order according to this value. The rationale is that the larger this value, the greater the number of vertices that are on the way between a pickup vertex and its delivery vertex. Once a pickup and its delivery are chosen, we have to decide where to insert them. Let us define $SP_i^{k,l}$ as the partial solution built so far and set i and $i+n$ in feasible positions k and l $(k < l)$, respectively, and $\overline{SP_i^{k,l}}$ as the partial binary vector x which represents the partial path. The algorithm selects the positions k and l $(k < l)$ which minimize $||\overline{SP_i^{k,l}} - X^*||_1$. The algorithm finishes when a Hamiltonian path is obtained.

7.2.2. Local search improvement phase

The procedure looks for improving the initial solution by exploring neighboring ones. Given a feasible Hamiltonian path, the neighborhood is created with the following movements:

- a pickup vertex i is shifted to any previous locations until reaching a delivery vertex corresponding to an incompatible pickup vertex or the origin vertex.
- a pickup vertex i is shifted to any later locations until reaching its delivery $i+n$.
- a delivery vertex $i+n$ is shifted to any previous locations until reaching its pickup vertex i.
- a delivery vertex $i+n$ is shifted to any later locations until reaching a pickup vertex incompatible with i or the destination vertex.

All Hamiltonian paths generated by these movements are feasible since precedence and incompatibility constraints are guaranteed. The neighborhood is thoroughly explored looking for the best solution that improves the actual incumbent solution.

7.3. Branching and search strategy

The idea behind our branching strategy is to incrementally construct partial paths (one starting in s and the other ending in t) by choosing the next vertex to visit in the partial path starting in s or the first vertex to visit in the partial path ending in t. Among all feasible possibilities, the associated fractional variable which maximizes the pseudo-cost is chosen.

Best-bound search strategy provided by CPLEX is set to select the next node to be processed.

8. Computational experience

In this section we evaluate our algorithm by analyzing the effectiveness of the family cuts, the primal heuristic and the branching strategy. Default CPLEX Branch and Cut algorithm is used to compare and to appreciate the performance of our Branch and Cut algorithm. Given that instances with $n = 10$ do not present a challenge since they are resolved by standard CPLEX configuration in non-significant time, we consider $n = 15$ and generate instances with $n = 20$ for 10%, 25%, 50% and 70% percentage of density to report on interesting instances.

Table 9
Acronyms valid inequalities.

R1SEC	Reinforced-1 subtour elimination constraints
R2SEC	Reinforced-2 subtour elimination constraints
Prec	Precedence constraints
2-GOC	2-Generalized order constraints
IndepIn	Independent set incompatibility inequalities
GLSEC	Generalized lifted subtour elimination constraints
ColIn	Coloring incompatibility subtour constraints
$\Pi\sigma$	Π and σ inequalities
Pol	Polynomial size inequalities: (33), (34), (35)

8.1. Cutting planes

In the first experiment we evaluate the valid inequalities introduced in a previous section by analyzing how they reinforced LP relaxation bounds at the root node. We do not limit the number of cutting plane iterations nor the number of cuts added in each iteration except for Coloring incompatibility subtour inequalities. In preliminary tests we observed that the separation of these inequalities consumes high cpu time. This is to be expected, since the separation algorithm works with cliques of size 2, 3 and 4 and 5 holes. If the density of the incompatibility graph is high or medium, the number of these structures makes computationally prohibitive the exhaustive test with each of them. After an experimental evaluation, we decided to use at most 50 of each of these structures as base for the inequalities analyzed. In each iteration, these structures are chosen at random with the aim of diversifying the search space.

ITVMi* is defined as **ITVMi** model to which the minimal equation system as well as linear and quadratic size inequality families (28) to (32) have been added.

We test single configurations (just one family cuts), the one which applies all families and finally, the one which also includes the general purpose cuts provided by CPLEX. The columns in the tables should be interpreted as follows: RelITVMi and RelITVMi* report the integrality gap of the initial linear relaxation of **ITVMi** and **ITVMi*** models, respectively.

Columns R1SEC, R2SEC, Prec, 2-GOC, IndepIn, GLSEC, ColIn, $\Pi\sigma$ and Pol (acronyms are defined in Table 9) show the final integrality gap when the particular family of valid inequalities is applied as cutting planes on **ITVMi*** model.

Column AllCuts reports the final gap when using all cutting plane families and the last column AllCuts+Cplex shows the final gap when general purpose cuts provided by CPLEX are also included. In all cases the integrality gap is based on z^* which refers to the optimal value of the final LP relaxation after finishing the cutting plane procedure. Results correspond to the average across the corresponding 25 instances for each density.

The first comment is that the equalities and inequalities explicitly incorporated into the model obtain a reduction gap between 1% and 2% without increasing the resolution time. Regarding cutting planes, results in Tables 10, 12, 14 and 16 show that ColIn inequalities are, by far, the most effective cutting planes, although this involves longer computation time. The reduction of the gap is more remarkable as the density increases. This is an expected behaviour since structures in higher density incompatibility graphs tend to have a higher chromatic number, which result in stronger inequalities. Although to a lesser extent, $\Pi\sigma$ and 2-GOC also prove to be useful for low density graphs and lose effectiveness as the density increases. IndepIn inequalities are helpful for medium and high density incompatibility graphs since the search of complete subgraphs during the separation algorithm seems to be more effective as the density increases.

Table 10
Set1 and n = 15: Initial and final root percentage gap after adding different cutting plane families.

p%	RelITVMi	RelITVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	23.24	21.12	20.26	20.90	19.97	16.00	19.89	19.43	8.85	18.36	20.56	7.20	7.20
25	26.71	24.74	24.23	24.62	23.94	21.34	22.75	23.82	8.46	22.66	24.06	7.93	7.93
50	25.86	24.35	24.13	24.33	24.01	23.00	20.76	24.16	5.01	23.18	23.70	4.69	4.69
70	19.04	17.03	16.99	17.03	16.99	16.69	13.61	17.00	2.11	16.53	16.30	1.89	1.89

Table 11
Set1 and $n = 15$: Computational time for the root node after adding different cutting plane families.

p%	RelITVMi	RelITVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.04	0.00	0.00	2.52	2.52
25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.20	2.28
50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	0.00	0.00	2.08	2.08
70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	2.52	2.56

Table 12
Set1 and $n = 20$: Initial and final root percentage gap after adding different cutting plane families.

p%	RelITVMi	RelITVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	24.93	23.59	22.62	23.57	22.43	18.38	22.09	22.47	11.39	20.37	23.34	9.79	9.79
25	30.44	29.37	28.82	29.23	28.73	26.49	27.61	28.90	12.65	26.89	28.99	12.20	12.20
50	29.03	27.81	27.61	27.80	27.57	26.55	25.72	27.71	8.25	26.31	27.14	7.97	7.97
70	22.49	20.93	20.87	20.91	20.85	20.68	17.57	20.91	4.58	20.27	20.52	4.17	4.17

Table 13
Set1 and $n = 20$: Computational time for the root node after adding different cutting plane families.

p%	RelITVMi	RelITVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	0.00	0.00	0.00	0.00	0.00	1.64	0.00	0.16	1.36	1.48	0.00	8.92	8.88
25	0.00	0.00	0.00	0.00	0.00	1.44	0.00	0.00	1.52	1.28	0.00	8.60	8.48
50	0.00	0.00	0.00	0.00	0.00	0.88	0.00	0.00	2.76	0.72	0.00	9.24	9.28
70	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.00	6.72	0.24	0.00	20.76	20.80

Table 14
Set2 and $n = 15$: Initial and final root percentage gap after adding different cutting plane families.

p%	RelITVMi	RelITVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	0.93	0.84	0.84	0.84	0.83	0.83	0.83	0.84	0.81	0.76	0.82	0.73	0.73
25	1.30	1.20	1.20	1.20	1.20	1.20	1.17	1.29	1.13	1.15	1.19	1.07	1.07
50	1.59	1.48	1.48	1.48	1.48	1.47	1.29	1.48	1.21	1.43	1.46	1.05	1.06
70	1.56	1.39	1.39	1.39	1.39	1.39	1.16	1.39	0.87	1.37	1.36	0.72	0.72

Table 15
Set2 and $n = 15$: Computational time for the root node after adding different cutting plane families.

p%	RelITVMi	RelITVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.92
25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.88	0.92
50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.04	1.00
70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	1.52	1.52

In regard to the rest of the inequality families, the results show that they are capable of improving the lower bound but, none of them achieves a significant gap reduction.

When using all cutting plane families, the results show that this combination improves even more the final root percentage gap. We experiment with other cut combinations but we do not notice any difference that is worth reporting. To sum up, the inequalities enhance with one another, achieving an excellent reduction of the gap.

It is worth mentioning that, even though gaps of the initial linear relaxation are much lower in the instances of Set2, the conclusions about the performance of the inequalities are valid for both Set1 and Set2.

Tables 11, 13, 15 and 17 show the cpu time it takes to apply the cutting plane procedure. We can see that separation algorithm for ColIn is the most time consuming, followed by $\Pi\sigma$ and 2-GOC. It is not surprising that, usually, efficiency goes hand-in-hand with expense of cpu time.

The combination of all inequalities takes more cpu time but it is not so expensive if we take into account the significant reduction of the gap. Regarding the addition of cuts provided by CPLEX package, some minor loss or even no differences in the quality of the lower bound are observed. The results show that they are not worth considering.

As a final conclusion we can say that all inequalities have an impact on the reduction of the gap and, if considered all together, it results in an effective strategy to obtain a stronger relaxation.

It is worth mentioning that, in Section 5, we made a comparison of the models and it was observed that **PDPInc2** has better initial gaps than **ITVMi**, but at a high computational cost. When cutting planes are applied to **ITVMi***, we noticed that a significant reduction gap is reached in less cpu time than required by **PDPInc2**.

Table 16
Set2 and $n = 20$: Initial and final root percentage gap after adding different cutting plane families.

p%	RellTVMi	RellTVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	0.91	0.82	0.82	0.82	0.82	0.81	0.82	0.82	0.80	0.78	0.81	0.76	0.76
25	1.45	1.41	1.41	1.41	1.41	1.40	1.37	1.41	1.36	1.37	1.40	1.31	1.31
50	1.99	1.93	1.93	1.93	1.93	1.93	1.82	1.93	1.73	1.91	1.91	1.60	1.60
70	2.03	1.92	1.92	1.92	1.91	1.91	1.60	1.92	1.46	1.90	1.90	1.14	1.14

Table 17
Set2 and $n = 20$: Computational time for the root node after adding different cutting plane families.

p%	RellTVMi	RellTVMi*	R1SEC	R2SEC	Prec	2-GOC	IndepIn	GLSEC	ColIn	$\Pi\sigma$	Pol	AllCuts	AllCuts+Cplex
10	0.00	0.00	0.00	0.00	0.00	0.56	0.00	0.00	0.00	0.84	0.00	3.28	3.32
25	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.36	0.00	2.80	2.80
50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60	0.24	0.00	3.96	3.92
70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.56	0.00	0.00	8.76	8.72

Table 18
Set1 and $n = 15$: Cplex default vs BC.

p	ITVMi				BC			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	24	6.39	6970.32	239058.52	4	0.08	2048.32	27204.76
25	24	3.84	7143.40	299014.44	3	0.04	2028.76	42895.72
50	4	0.19	2373.64	207984.12	0	0.00	252.16	5387.36
70	0	0.00	21.80	5661.50	0	0.00	18.00	71.92

Table 19
Set1 and $n = 20$: Cplex default vs BC.

p	ITVMi				BC			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	25	38.67	7197.40	94279.28	23	8.26	6736.72	25417.08
25	25	29.10	7197.92	104493.12	25	8.88	7197.40	42656.00
50	24	6.57	6979.60	238938.72	16	1.40	5131.08	47645.60
70	5	0.18	3522.24	309666.32	0	0.00	309.48	2297.76

Table 20
Set2 and $n = 15$: Cplex default vs BC.

p	ITVMi				BC			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	4	0.06	2692.40	144086.08	0	0.00	441.84	12366.52
25	1	0.02	1891.40	145188.44	0	0.00	431.04	16208.00
50	0	0.00	63.76	9500.60	0	0.00	38.00	1023.72
70	0	0.00	2.96	860.76	0	0.00	17.96	101.40

This certainly proves that our goal of strengthening relaxation is satisfactorily fulfilled by the valid inequalities characterized.

8.2. Branch and cut

To provide a deep insight into the effectiveness of cutting planes, we experiment with them embedded in a Branch and Cut scheme. We name this implementation BC algorithm.

According to our computational experience with cutting planes we reported above, we decide to include AllCuts as the strategy for looking at cutting planes up to 20 nodes of the search tree. We limit to 20 rounds of cutting plane procedure. After 20 explored nodes and for the next 10 nodes, ColIn and 2-GOC are the only cuts applied.

The results are shown inTables 18–21. Column **UInst** refers to the number of unsolved instances within 7200s time limit, column **Gap** to the average percentage gap, column **t(s)** to the average CPU time and column **#B&Bn** to the average of explored nodes of the search tree. Gap refers to $100(BestSol - z^*)/BestSol$ where z^* is the best solution found and *BestSol*, the best known solution. Results correspond to the average across the corresponding 25 instances for each density.

As a general comment, the cutting planes allow to reach optimality in many more instances, both in Set1 and Set2 instances.

For $n = 15$ the hardest instances turned out to be those with density lower than 50. CPLEX could not reach optimality in most instances within the time limit. However, the addition of cutting planes improves the performance, solving almost all instances. The final percentage gap in unsolved instances is considerably reduced, with values well below 1%. Moreover, the average CPU time is reduced significantly, mainly due to the reduction in size of the search tree.

The tree size reduction factor grows as the density of the instances increases. For graphs of high densities, although this reduction is very significant, it is not always enough to substantially compensate for total time. The resolution time of CPLEX in these instances is very small and as expected, the incorporation of separation procedures of cutting planes in the nodes of the tree and resolving the LP-relaxations, although it is small, is relatively significant in the total time.

Table 21
Set2 and $n = 20$: Cplex default vs BC.

p	ITVMi				BC			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	25	2.75	7198.00	110268.16	25	0.39	7198.00	52505.72
25	25	3.02	7197.68	158859.72	25	0.57	7196.44	81311.08
50	20	0.40	6716.72	322044.48	6	0.03	4327.08	85963.04
70	0	0.00	205.60	20711.48	0	0.00	170.48	1845.92

Table 22
Set1 and $n = 15$: BC vs BC+Heu.

p	BC				BC+ Heu			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	4	0.08	2048.32	27204.76	4	0.00	1739.88	26132.56
25	3	0.04	2028.76	42895.72	1	0.00	1916.44	45084.88
50	0	0.00	252.16	5387.36	0	0.00	158.52	4191.88
70	0	0.00	18.00	71.92	0	0.00	17.20	72.72

Table 23
Set1 and $n = 20$: BC vs BC+Heu.

p	BC				BC + Heu			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	23	8.26	6736.72	25417.08	23	0.08	6684.76	30401.44
25	25	8.88	7197.40	42656.00	24	0.16	7030.64	45797.48
50	16	1.40	5131.08	47645.60	15	0.08	4970.52	47783.36
70	0	0.00	309.48	2297.76	0	0.00	265.64	2154.72

Table 24
Set2 and $n = 15$: BC vs BC+Heu.

p	BC				BC + Heu			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	0	0.00	441.84	12366.52	0	0.00	302.44	8906.28
25	0	0.00	431.04	16208.00	0	0.00	257.72	10917.76
50	0	0.00	38.00	1023.72	0	0.00	32.96	899.96
70	0	0.00	17.96	101.40	0	0.00	17.40	93.60

For $n = 20$, where instances of higher density are more difficult to solve for CPLEX, it can be seen that the cuts make a difference: more instances are resolved and the average time is reduced. Even though BC is also unable to reach optimality in most of the low density instances, the final percentage gap is considerably reduced in more than 70%.

When we chose the ITVMi model to develop on it a Branch and Cut algorithm, we based our decision on the fact that the relaxations were very quick to solve and that the total resolution time used went hand in hand with the large number of explored nodes. Convinced that if we succeeded in strengthening the relaxation without falling into prohibitive time of resolution, we could get a substantial decrease in the size of the search tree and, in this way, consume less cpu time. We succeeded in this goal. The computational experiments show that valid inequalities are key to obtaining an algorithm that surpasses CPLEX default algorithm.

In conclusion, a close observation of the table reveals that, thanks to the cutting planes, our algorithm dominates the Branch and Cut CPLEX default algorithm by the number of solved instances, the final gap and the computational time.

8.3. Primal heuristic

In the next experiments, we test the performance of our proposed primal heuristic. Cutting planes are managed according to the strategy we mentioned above and the primal heuristic is applied in all nodes of the search tree. The results are shown inTables 22–25.

We observe a moderate reduction in size of the tree as well as in CPU time required to reach optimality. In some set of instances (Set1, $n = 15$ and 25%, $n = 20$ and 50%), the primal heuristic helps to solve more instances. For unsolved instances, in all cases, the algorithm reaches the best solution known but needs more time to prove optimality.

The power of the heuristic is more significant in low density instances and decreases as the density increases. This type of behavior is expected since the neighborhood where the heuristic works is smaller as the incompatibility increases.

We can affirm that the heuristic satisfactorily fulfills two purposes. On the one hand, it helps to solve more instances to optimality and on the other hand, it provides quality solutions within the time limit. Both objectives were achieved improving, or at least not increasing, the total computational time.

8.4. Branching strategy

The last experiment compares the proposal branching strategy with the default strategy used by CPLEX. The cutting plane algorithm follows the scheme described above and the primal heuristic is applied after each LP relaxation. The results are shown inTables 26–29.

The branching strategy works for 10% of density of incompatibility. In the case of Set1 instances and $n = 15$, this strategy allows

Table 25
Set2 and $n = 20$: BC vs BC+Heu.

p	BC				BC + Heu			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	25	0.39	7197.04	52505.72	24	0.05	7089.40	54779.92
25	25	0.57	7196.44	81311.08	23	0.04	7109.56	83661.56
50	6	0.03	4327.08	85963.04	6	0.01	3683.52	74206.56
70	0	0.00	170.48	1845.92	0	0.00	142.48	1485.04

Table 26
Set1 and $n = 15$: BC+Heu vs BC+Heu+Branching.

p	BC + Heu				BC + Heu + Branching			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	4	0.00	1739.88	26132.56	0	0.00	552.52	10574.28
25	1	0.00	1916.44	45084.88	1	0.00	1347.72	28394.88
50	0	0.00	158.52	4191.88	0	0.00	298.64	7873.36
70	0	0.00	17.20	72.72	0	0.00	21.52	260.68

Table 27
Set1 and $n = 20$: BC+Heu vs BC+Heu+Branching.

p	BC + Heu				BC + Heu + Branching			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	23	0.08	6684.76	30401.44	22	0.20	6487.24	21490.84
25	24	0.16	7030.64	45797.48	23	0.70	6891.24	23896.84
50	15	0.08	4970.52	47783.36	15	0.22	5052.08	25615.44
70	0	0.00	265.64	2154.72	2	0.05	1152.16	13127.88

Table 28
Set2 and $n = 15$: BC+Heu vs BC+Heu+Branching.

p	BC + Heu				BC + Heu + Branching			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	0	0.00	302.44	8906.28	0	0.00	202.44	7204.32
25	0	0.00	257.72	10917.76	0	0.00	344.28	15665.32
50	0	0.00	32.96	899.96	0	0.00	106.80	5491.12
70	0	0.00	17.40	93.60	0	0.00	25.36	818.24

Table 29
Set2 and $n = 20$: BC+Heu vs BC+Heu+Branching.

p	BC + Heu				BC + Heu + Branching			
%	UInst	Gap	t(s)	#B&Bn	UInst	Gap	t(s)	#B&Bn
10	24	0.05	7089.40	54779.92	18	0.03	6434.12	45368.64
25	23	0.04	7109.56	83661.56	24	0.06	7162.44	71415.08
50	6	0.01	3683.52	74206.56	25	0.14	7197.36	72364.92
70	0	0.00	142.48	1485.04	4	0.02	2732.92	42528.32

us to solve all the instances and reduces the CPU time to one third of the original time. For Set2 instances, the strategy attains a decrease of the time in one third. For $n = 20$, the strategy helps to solve more instances both for Set1 and Set2.

For 25% of density of incompatibility, the strategy works for Set1 instances. For $n = 15$, the cpu time is reduced and, for $n = 20$, the algorithm could solve more instances. In the case of Set2 instances and $n = 15$, on average, the CPU time increases. But, when analyzing the instances one by one, we observe that this is mainly due to three instances in which the strategy uses more than twice the original time. In the rest of the instances, the results are mixed and there is not a clear winner. For $n = 20$, the strategy gets worse results.

For 50% and 70% of density, the strategy does not seem very efficient. Results show an insignificant or negative difference. One possible explanation is that the construction of partial paths is frequently found with the impossibility of extending them and the algorithm is forced to open and to explore more nodes.

9. Conclusions

The main contribution of this work is the introduction of a new generalization of the One-to-One Pick and Delivery problem. We propose three integer programming models and we have studied the polytope associated to one of them. We have characterized the minimal equation system and several new valid inequalities. The effectiveness of these inequalities has been demonstrated through computational experiments, showing an excellent reduction of the percentage gaps.

The combination of cutting planes along with the primal heuristic, which helps to find feasible solutions, contributes to achieve a successful Branch and Cut algorithm that is capable of solving instances that are out of the reach of CPLEX. Moreover, our algorithm overcomes CPLEX both in CPU time and final gap. We consider that our work is a promising scheme to be followed in practice for solving the Pick and Delivery Problem with Incompatibilities. There is still place for future work to increase the size

of solved instances. For example, the branching strategy could be improved and more valid inequality families could be characterized. Furthermore, given that Euclidean instances have proved to be more difficult than random cost instances, it would be interesting to exploit this condition to solve instances with a greater number of requests.

Acknowledgments

This research is partially supported by UBACyT Grant 20020170100484BA from Universidad de Buenos Aires, Argentina and FONCyT grant PICT-2016-2677 and PICT-2017-1826 from the Government of Argentina. The authors are grateful to the anonymous referees and the general editor for their careful reading and valuable comments, which helped to improve a previous version of the article.

References

Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J., 2006. The Traveling Salesman Problem: A Computational Study. Princeton University Press.
Battarra, M., J.Cordeau, Iori, M., 2014. Chapter 6: pickup-and-delivery problems for goods transportation. In: Toth, P., Vigo, D. (Eds.), Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, pp. 161–191.
Battarra, M., Monaci, M., Vigo, D., 2009. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. Comput. Oper. Res. 36 (11), 3041–3050.
Berbeglia, G., Cordeau, J., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP Official J. Span. Soc.Stat. Oper. Res. 15 (1), 1–31.
Brélaz, D., 1979. New methods to color the vertices of a graph. Commun. ACM 22, 251–256.
Cao, B., 1992. Transportation problem with nonlinear side constraints a branch and bound approach. Methods Models Oper. Res. 36, 185–197.
Cao, B., Uebe, G., 1995. Solving transportation problems with nonlinear side constraints with tabu search. Comput. Oper. Res. (6) 593–603.
Caramia, M., Guerriero, F., 2010. A milk collection problem with incompatibility constraints. INFORMS J. Appl. Anal. 40 (2), 130–143.
Ceselli, A., Righini, G., Salani, M., 2009. A column generation algorithm for a rich vehicle-routing problem. Transp. Sci. 43 (1), 56–69.
Colombi, M., Corberán, A., Mansini, R., I.Plana, Sanchis, J.M., 2017. The directed profitable rural postman problem with incompatibility constraints. Eur. J. Oper. Res. 261 (2), 549–562.
Cordeau, J., G.Laporte, 2007. The dial-a-ride problem: models and algorithms. Ann. Oper. Res. 153 (1), 29–46.
Cordeau, J., Laporte, G., Ropke, S., 2008. Recent models and algorithms for one–to-one pickup and delivery problems. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 327–357.
Cornillier, F., Boctor, F., Renaud, J., 2012. Heuristics for the multi-depot petrol station replenishment problem with time windows. Eur. J. Oper. Res. 220 (2), 361–369.
Dantzig, G.B., Fulkerson, D.R., Johnson, S.M., 1954. Solution of a large-scale traveling-salesman problem. Oper. Res. 3, 393–410.
Dumitrescu, I., Ropke, S., Cordeau, J., Laporte, G., 2010. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. Math. Programm. 121 (2), 269–305.
Factorovich, P., 2019. Problemas de recolección y entrega punto a punto. Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires Ph.D. thesis. https://digital.bl.fcen.uba.ar/collection/tesis/browse/CL4/6
Ficker, A.M.C., Spieksma, F.C.R., Woeginger, G.J., 2018. The transportation problem with conflicts. Ann. Oper. Res..
Gendreau, M., Laporte, G., Semet, F., 2004. Heuristics and lower bounds for the bin packing problem with conflicts. Comput. Oper. Res. 31 (3), 347–358.
Gendreau, M., Manerba, D., Mansini, R., 2016. The multi-vehicle traveling purchaser problem with pairwise incompatibility constraints and unitary demands: a branch-and-price approach. Eur. J. Oper. Res. 248 (1), 59–71.
Goossens, D., Spieksma, F.C.R., 2009. The transportation problem with exclusionary side constraints. 4OR Q. J. Oper. Res. 7 (1), 51–60.
Henkea, T., Speranza, M.G., Wäscher, G., 2019. A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. Ann. Oper. Res. 275, 321–338.
Ho, S.C., Szeto, W., Kuo, Y., Leung, J., Petering, M., Tou, T.W., 2018. A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. Part B 111, 395–421.
Hu, Z., Sheu, J., Zhao, L., Lu, C., 2015. A dynamic closed-loop vehicle routing problem with uncertainty and incompatible goods. Transp. Res. Part C 55, 273–297.
Iori, M., Martello, S., 2010. Routing problems with loading constraints. Top 18, 4–27.
Kowalczyk, D., Leus, R., 2017. An exact algorithm for parallel machine scheduling with conflicts. J. Schedul. 20 (4), 355–372.
Lahyani, R., Coelho, L.C., Khemakhem, M., Laporte, G., Semet, F., 2015. A multi-compartment vehicle routing problem arising in the collection of olive oil in tunisia. Omega 51, 1–10.
Letchford, A.N., Salazar-González, J.-J., 2016. Stronger multi-commodity flow formulations of the (capacitated) sequential ordering problem. Eur. J. Oper. Res. 251 (1), 74–84.
Manerba, D., Mansini, R., 2015. A branch-and-cut algorithm for the multi-vehicle traveling purchaser problem with pairwise incompatibility constraints. Networks 65 (2), 139–154.
Manerba, D., Mansini, R., 2016. The nurse routing problem with workload constraints and incompatible services. IFAC-PapersOnLine 49 (12), 1192–1197. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016
Molenbruch, Y., Braekers, K., Caris, A., 2017. Typology and literature review for dial-a-ride problems. Ann. Oper. Res. 259, 295–335.
ONU, 2007. UN recommendations on the transport of dangerous goods. Model regulations (fifteenth ed.). New York and Geneva.
Paredes-Belmar, G., Bronfman, A., Marianov, V., Latorre-Núñez, G., 2017. Hazardous materials collection with multiple-product loading. J. Clean. Prod. 141, 909–919.
Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008. A survey on pickup and delivery problems part ii : transportation between pickup and delivery locations. J. für Betriebswirtschaft 58 (1), 21–51.
Riera-Ledesma, J., Salazar-González, J., 2012. Solving school bus routing using the multiple vehicle traveling purchaser problem: a branch-and-cut approach. Comput. Oper. Res. 39 (1), 391–404.
Ruland, K.S., Rodin, E.Y., 1997. The pickup and delivery problem: faces and branch-and-cut algorithm. Comput. Math. Appl. 33 (12), 1–13.
Sadykov, R., Vanderbeck, F., 2012. Bin packing with conflicts: a generic branch-and-price algorithm. INFORMS J. Comput. 25 (2), 244–255.
Sarin, S.C., Sherali, H.D., Bhootra, A., 2005. New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. Oper. Res. Lett. 33 (1), 62–70.
Sun, M., 2002. The transportation problem with exclusionary side constraints and two branch-and-bound algorithms. Eur. J. Oper. Res. 140 (3), 629–647.
Vancroonenburg, W., Croce, F.D., Goossens, D., Spieksma, F., 2014. The red blue transportation problem. Eur. J. Oper. Res. 237 (3), 814–823.