A compact arc-based ILP formulation for the pickup and delivery problem with divisible pickups and deliveries

Jargalsaikhan, Bolor; Romeijnders, Ward; Roodbergen, K.J.

Published in:
Transportation Science

DOI:
10.1287/trsc.2020.1016

Publication date:
2021

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Compact Arc-Based ILP Formulation for the Pickup and Delivery Problem with Divisible Pickups and Deliveries

Bolor Jargalsaikhan, Ward Romeijnders, Kees Jan Roodbergen

To cite this article:
Bolor Jargalsaikhan, Ward Romeijnders, Kees Jan Roodbergen (2021) A Compact Arc-Based ILP Formulation for the Pickup and Delivery Problem with Divisible Pickups and Deliveries. Transportation Science 55(2):336-352. https://doi.org/10.1287/trsc.2020.1016

http://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 55, No. 2, March–April 2021, pp. 336–352
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

A Compact Arc-Based ILP Formulation for the Pickup and Delivery Problem with Divisible Pickups and Deliveries

Bolor Jargalsaikhan,[a] Ward Romeijnders,[a] Kees Jan Roodbergen[a]

[a] Department of Operations, Faculty of Economics and Business, University of Groningen, 9747 AE Groningen, Netherlands

Contact: bolor.jargalsaikhan@gmail.com (BJ); w.romeijnders@rug.nl, https://orcid.org/0000-0002-6918-5635 (WR); k.j.roodbergen@rug.nl, https://orcid.org/0000-0002-4083-2106 (KJR)

Received: July 11, 2019
Revised: October 21, 2019; May 13, 2020; August 4, 2020
Accepted: August 17, 2020
Published Online in Articles in Advance: January 28, 2021

https://doi.org/10.1287/trsc.2020.1016

Abstract. We consider the capacitated single vehicle one-to-one pickup and delivery problem with divisible pickups and deliveries (PDPDPD). In this problem, we do not make the standard assumption of one-to-one pickup and delivery problems (PDPs) that each location has only one transportation request. Instead we assume there are multiple requests per location that may be performed individually. This may result in multiple visits to a location. We provide a new compact arc-based integer linear programming (ILP) formulation for the PDPDPD by deriving time-consistency constraints that identify the order in which selected outgoing arcs from a node are actually traversed. The formulation can also easily be applied to the one-to-one PDP by restricting the number of times that a node can be visited. Numerical results on standard one-to-one PDP test instances from the literature show that our compact formulation is almost competitive with tailor-made solution methods for the one-to-one PDP. Moreover, we observe that significant cost savings of up to 15% on average may be obtained by allowing divisible pickups and deliveries in one-to-one PDPs. It turns out that divisible pickups and deliveries are not only beneficial when the vehicle capacity is small, but also when this capacity is unrestrictive.

Funding: This work was partially funded by the Netherlands Organisation for Scientific Research and TKI Dinalog [Grants 438-13-216 (Jargalsaikhan) and 439-16-612 (Roodbergen)].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2020.1016.

Keywords: integer linear program • pickup and delivery problem • divisible pickups and deliveries

1. Introduction

We consider the capacitated single-vehicle one-to-one pickup and delivery problem with divisible pickups and deliveries (PDPDPD or in short $(PD)^3$). This problem is related to the standard multicommodity one-to-one pickup and delivery problem (PDP), in which several transportation requests, each corresponding to a unique commodity with an origin and a destination, have to be carried out at minimal costs using a single capacitated vehicle. The difference from the standard one-to-one PDP, however, is that in the $(PD)^3$ we do not assume that each location is visited exactly once. Instead we allow multiple visits to the same location to separately deal with transportation requests that have the same origin but a different destination or the same destination but a different origin. However, the loads of the transportation requests cannot be split. This is, for example, relevant in offshore logistics when technicians need to be transported between offshore platforms using a helicopter with limited capacity. Allowing divisible pickups and deliveries in such problems may lead to significant cost savings compared with when these requests have to be combined.

We propose a new integer linear programming (ILP) formulation for the $(PD)^3$. Our formulation is arc-based, with binary flow variables representing the paths from the starting depot to the origin locations of each transportation request and from their origin locations to their destination locations, along which single-vehicle traverses. The challenge in deriving a correct formulation is that it is not clear which of the selected outgoing arcs is traversed first based on the binary flow variables. More importantly, for any given set of selected arcs it is unclear whether there exists an Euler path from the starting depot to ending depot that can fulfill all transportation requests. To address this issue, we introduce a new set of constraints that guarantees that all selected arcs are time consistent. Intuitively, this new set of constraints identifies the traversed sequence of the selected outgoing arcs from a node and only allows solutions for which all transportation requests can be carried out. The obtained ILP formulation can also easily be applied to the standard one-to-one PDP by restricting the number of times that a node may be visited.

The main contributions of this paper are as follows.

- We derive a novel compact arc-based ILP formulation for the one-to-one PDP with divisible pickups

and deliveries, $(PD)^3$, by deriving time-consistency constraints between arcs.

- Our ILP formulation does not use any artificial node duplication. This enabled us to be the first to exactly solve instances of the $(PD)^3$ of significant size: we solve instances involving up to 25 nodes or 15 transportation requests. This is significantly larger than the previous record of three nodes.
- When solving standard one-to-one PDPs, our compact formulation for the $(PD)^3$ is almost competitive with solution methods that were tailor-made for the standard one-to-one PDP.
- We show that cost savings of up to 15% on average may be obtained by allowing divisible pickups and deliveries in one-to-one PDPs. These savings are not only obtained when the vehicle capacity is small, but also when the vehicle capacity is unrestrictive.

The remainder of the paper is organized as follows. We first provide a summary of the relevant literature in Section 2. Then, we derive our new compact ILP formulations for the one-to-one PDP and the $(PD)^3$ in Section 3. In particular, we prove that both ILP formulations are correct. Afterward, we present numerical experiments and insights in Section 4. Finally, we conclude our paper with a discussion in Section 5.

2. Literature Review

In this section, we review the routing literature related to our problem. To keep this literature review concise, we mainly focus on exact solution methods. We distinguish three categories of routing problems: (1) standard vehicle routing problems, which consider only transportation requests for loads that originate at the depot and must be delivered to a number of locations; (2) one-to-many-to-one PDPs, which consider transportation requests for loads that originate at the depot and must be delivered to locations and loads that originate at locations and must be delivered to the depot; (3) one-to-one PDPs, which consider transportation requests between pairs of locations, while the vehicle starts and ends empty at the depot. For a general overview of PDPs, we refer to the survey papers of Berbeglia et al. (2007) and Parragh, Doerner, and Hartl (2008). We study a variant of the one-to-one PDP in this paper.

For each of the three categories of routing problems, numerous variants exist. We focus here on those variants that allow for multiple visits to locations. Specifically, we discuss routing problems with multiple location visits as resulting from split loads or from divisible pickups and deliveries. Typically, the distinction between the two is considered to be as follows. For *split loads* each location can serve as a pickup or delivery point (not both) for one commodity (e.g., Haddad et al. 2018). That is, everything that is to be picked up from (delivered to) a location has the same destination (origin). The model can determine the quantity to be picked up or delivered in each visit. For *divisible pickups and deliveries*, each location can serve as a pickup and/or delivery point for multiple commodities (e.g., Nagy et al. 2015). That is, every location may require transportation of loads to and/or from multiple other locations. The model can determine which commodity is picked up or delivered in which visit. Our problem classifies as a problem with divisible pickups and deliveries.

Split loads in the context of standard vehicle routing problems imply that multiple vehicles may be used to deliver the total requested load to a location, that is, the requested load may be split over a number of vehicles. This is known as the split delivery vehicle routing problem (SDVRP; see Dror and Trudeau 1989). As a consequence, locations can be visited more than once, which may result in considerable cost savings (Archetti, Savelsbergh, and Speranza 2006). If loads exceed the vehicle capacity, then they are only considered split if a location is visited *more often than required*. In contrast, in problems with divisible pickups and deliveries we assume that the loads of all transportation requests are below the vehicle capacity. Thus, in a one-to-many-to-one PDP with divisible pickups and deliveries, each customer location is visited at most twice (i.e., once for the pickup and once for the delivery; see Nagy et al. 2015). Bruck and Iori (2017) consider a similar problem and propose an exact solution method based on a nonelementary ILP formulation of the problem. Salazar-González and Santos-Hernández (2015) present a branch-and-cut algorithm to address the one-to-many-to-one PDP with split loads, albeit with one rather than two commodities. A heuristic for the one-to-many-to-one PDP with split loads is presented in Lai et al. (2015), which is restricted to an incomplete graph because of the specifics of the presented case study.

For standard routing problems with split loads, any pair of vehicle routes in an optimal solution can have at most one customer location in common (Dror and Trudeau 1989), so that in the optimal solution any arc between customer locations will be traversed at most once. This property aids in strongly reducing the solution space. For one-to-many-to-one PDPs with divisible pickups and deliveries, however, a comparably strong reduction of the solution space appears not to exist (Nagy et al. 2015). In optimal solutions to a one-to-many-to-one PDP with divisible pickups and deliveries, arcs may therefore be traversed multiple times (Nagy et al. 2015). This also holds for our $(PD)^3$, the one-to-one PDP with divisible pickups and deliveries (see Section 3.2).

A direct approach to address divisible pickups and deliveries is to artificially duplicate the nodes in the graph. Each node in the resulting extended graph then

corresponds to the origin or destination of a single transportation request only. With this approach, our $(PD)^3$ would reduce to a standard one-to-one PDP on this extended graph. The issue with this approach, however, is that state-of-the-art exact methods for the standard one-to-one PDP can only solve instances with up to 30 nodes and 15 transportation requests (Hernández-Pérez and Salazar-González 2009, Gouveia and Ruthmair 2015). A one-to-many-to-one PDP with divisible pickup and deliveries having N nodes would at most have $2N$ nodes in the extended graph. This approach is taken in, for example, Nagy et al. (2015). In contrast, a one-to-one PDP with divisible pickups and deliveries, like ours, may have up to $2N(N-1)$ nodes in the extended graph. This means that only very small problem instances of our $(PD)^3$ can be solved exactly using this approach. In fact, the only known exact solutions for the $(PD)^3$ are due to Psaraftis (2011) for instances with up to three nodes.

The standard one-to-one PDP (Hernández-Pérez and Salazar-González 2009, Gouveia and Ruthmair 2015, Letchford and Salazar-González 2016) and the dial-a-ride problem (Cordeau 2006; Baldacci, Bartolini, and Mingozzi 2011) are strongly related and assume each location can be visited at most once. Locations may be visited multiple times in the one-to-one PDP with split loads (Öncan et al. 2011, Şahin et al. 2013, Haddad et al. 2018) and the dial-a-ride problem with split loads (Parragh, de Sousa, and Almada-Lobo 2015). It is assumed that each origin (destination) is linked to exactly one transportation request and one destination (origin). In contrast, in the $(PD)^3$ we consider that pickups from one location may need to be delivered to many other locations and that deliveries for one location may need to be pickup up from many other locations. We do not allow loads to be split if they have the same origin and destination, but we do allow loads to be divided among multiple pickups (deliveries) if they have a different destination (origin). To our knowledge, there is no literature that presents an exact method that can solve the $(PD)^3$ for instances of significant size. Psaraftis (2011) presents a dynamic programming approach for the $(PD)^3$, solving instances with up to three nodes and six requests. Heuristic methods are deployed in Nowak, Ergun, and White (2008) and Nowak, Ergun, and White (2009), where both split loads and divisible pickups and deliveries are considered. Also Nowak, Hewitt, and White (2012) consider both split loads and divisible pickups and deliveries. However, the assumption that all pickups in a route must be made before any delivery is made strongly reduces the solution space compared with our setting that builds on the standard one-to-one PDP.

Table 1 summarizes the main differences between our work and the literature discussed above that employ exact solution methods. In this table, we use DPD and SL to refer to divisible pickups and deliveries and to split loads, respectively. We use P,D,PD and P,D to indicate whether a node in the model may serve as both pickup and delivery location or not. In the latter case, the model requires node duplication to model locations that have both a pickup and a delivery. For example, Nagy et al. (2015) study the one-to-many-to-one PDP with divisible pickups and delivery, and hence, this paper could potentially be classified as "max requests per node" = 2 and "max node visits" = 2. However, the article uses node duplication, which reduces the problem to "max requests per node" = 1 and "max node visits" = 1 with twice as many nodes. Our paper allows "max requests per node" and "max node visits" to take any value. Only three papers fall broadly into the same category as ours; however, these papers are distinctly different

Table 1. Main Characteristics of Our Paper Compared with the Literature

Reference	Max requests per node	Max node visits	Location types	Problem type
Cordeau (2006)	1	1	P,D	—
Baldacci et al. (2011)	1	1	P,D	—
Nagy et al. (2015)	1	1	P,D	DPD
Öncan et al. (2011)	1	2	P,D	SL
Parragh, de Sousa, and Almada-Lobo (2015)	1	∞	P,D	SL
Haddad et al. (2018)	1	∞	P,D	SL
Hernández-Pérez and Salazar-González (2009)	m	1	P,D,PD	—
Gouveia and Ruthmair (2015)	m	1	P,D,PD	—
Letchford and Salazar-González (2016)	m	1	P,D,PD	—
Psaraftis (2011)	m	m	P,D,PD	DPD
Nowak, Hewitt, and White (2012)	m	∞	P,D,PD	SL+DPD
Bruck and Iori (2017)	2	2	P,D,PD	DPD
This paper	m	m	P,D,PD	DPD

Note. DPD, divisible pickups and deliveries; SL, split loads.

(also discussed above). First, the method of Psaraftis (2011) only solves problems with up to three nodes. Second, Nowak, Hewitt, and White (2012) study a strongly restricted version of the one-to-one PDP. Third, Bruck and Iori (2017) study the one-to-many-to-one PDP, whereas we study the one-to-one PDP.

3. Problem Formulation

In this section, we present our novel compact ILP formulation for the $(PD)^3$. This formulation can also be used to solve the one-to-one PDP, which we discuss first in Section 3.1. The formulation for the $(PD)^3$ is discussed in Section 3.2.

Let $G = (\bar{V}, A)$ denote a complete directed graph with nodes $\bar{V} = \{0, 1, \ldots, N\}$ and arcs A. Here, node 0 represents the starting depot, node N the ending depot, and the remaining nodes $V = \bar{V}\backslash\{0, N\}$ represent pickup and delivery locations. The problem is to transport several commodities $k \in K$ from their origin $o_k \in V$ to their destination $d_k \in V$ using a single vehicle at minimal costs. The vehicle has to start and end at the starting and ending depots, respectively. Moreover, the capacity of the single vehicle is Q, and each commodity $k \in K$ has a weight $0 \le q_k \le Q$. We assume that, once a commodity is picked up at its origin, it stays in the vehicle until it is delivered at its destination. Finally, the costs of traversing arc $a = (i, j) \in A$ are denoted $c_a \ge 0$ and represent the costs of traversing the shortest path from node i to node j.

In the $(PD)^3$ of Section 3.2, we allow divisible pickups and deliveries by not restricting the number of times a node is visited, whereas in the one-to-one PDP of Section 3.1, every location must be visited exactly once. In all ILP formulations in this paper, we use the binary variables x_a, y_a^k, and z_a^k for all arcs $a \in A$ and commodities $k \in K$. They have the following interpretation:

- x_a represents whether the vehicle traverses $a \in A$,
- y_a^k represents whether $a \in A$ is traversed at any point in time before commodity $k \in K$ is picked up, and
- z_a^k represents whether $a \in A$ is traversed at any point in time after commodity $k \in K$ is picked up and before it is delivered.

For each commodity $k \in K$, the variables y_a^k and z_a^k define a path from the starting depot to the origin o_k and from the origin o_k to the destination d_k, respectively. The latter represents the path along which commodity k is transported. By imposing that $y_a^k + z_a^k \le x_a$ for all $k \in K$, we ensure that $x_a = 1$ if this arc a is on such a path.

3.1. One-to-One Pickup and Delivery Problem

In this subsection, we present a compact ILP formulation for the one-to-one PDP without divisible pickups and deliveries. We refer to this ILP formulation as (IP):

$$\min_{x,y,z} \quad \sum_{a\in A} c_a x_a$$

$$\text{s.t.} \quad \sum_{a:s(a)=0} x_a = 1, \quad \sum_{a:t(a)=0} x_a = 0, \tag{1}$$

$$\sum_{a:s(a)=N} x_a = 0, \quad \sum_{a:t(a)=N} x_a = 1, \tag{2}$$

$$\sum_{a:s(a)=v} x_a = \sum_{a:t(a)=v} x_a, \qquad v \in V, \tag{3}$$

$$y_a^k + z_a^k \le x_a, \qquad a \in A, k \in K, \tag{4}$$

$$\sum_{a:s(a)=0} y_a^k = 1, \qquad k \in K, \tag{5}$$

$$\sum_{k\in K} \sum_{a:t(a)=N} \left(y_a^k + z_a^k\right) = 0, \tag{6}$$

$$\sum_{a:t(a)=o_k} y_a^k - \sum_{a:s(a)=o_k} y_a^k = 1, \qquad k \in K, \tag{7}$$

$$\sum_{a:t(a)=v} y_a^k - \sum_{a:s(a)=v} y_a^k = 0, \qquad k \in K, v \in V\backslash\{o_k\}, \tag{8}$$

$$\sum_{a:t(a)=o_k} z_a^k - \sum_{a:s(a)=o_k} z_a^k = -1, \qquad k \in K, \tag{9}$$

$$\sum_{a:t(a)=d_k} z_a^k - \sum_{a:s(a)=d_k} z_a^k = 1, \qquad k \in K, \tag{10}$$

$$\sum_{a:t(a)=v} z_a^k - \sum_{a:s(a)=v} z_a^k = 0, \qquad k \in K, v \in V\backslash\{o_k, d_k\}, \tag{11}$$

$$\sum_{k\in K} q_k z_a^k \le Q x_a, \qquad a \in A, \tag{12}$$

$$\sum_{a:t(a)=v} x_a = 1, \qquad v \in V, \tag{13}$$

$$x_a, y_a^k, z_a^k \in \{0, 1\}, \qquad a \in A, k \in K.$$

Here, $s(a)$ and $t(a)$ represent the starting and ending nodes of arc a, respectively. Constraints (1) ensure that we exit the starting depot once and do not enter it again. Constraints (2) make sure that we enter the ending depot once and do not leave it. Constraints (3) ensure that the number of times a node $v \in V$ is entered equals the number of times it is exited. Constraints (4) say that we can only send flow along arc $a \in A$ if arc a is actually traversed. Constraints (5) ensure that all commodities still have to be picked up when we exit the starting depot. Similarly, constraint (6) makes sure that once we enter the ending depot all commodities are delivered. Constraints (7)–(11) represent "flow continuity" and "pickup and delivery restrictions" on y and z. They ensure for every $k \in K$ that the y^k and z^k variables represent a path from the depot to the origin o_k and from the origin o_k to the destination d_k, respectively. For the z^k variables, this is because (9) and (10) guarantee that the z^k-path starts

and ends at o_k and d_k, respectively, and (11) implies that if the z^k-path enters any other node $v \in V\backslash\{o_k, d_k\}$, then the z^k-path also has to leave that node v. Similarly for the y^k variables, constraints (5) and (7) guarantee that the y^k-path starts and ends at the depot and o_k, respectively, and (8) implies that if the y^k-path enters any node $v \in V\backslash\{o_k\}$, then it also leaves it. Finally, constraints (12) specify the capacity constraints, and constraints (13) guarantee that every node is visited exactly once.

Proposition 1. *The one-to-one PDP is correctly formulated by* (IP).

Proof. Because every node is visited exactly once, because of constraints (13), it suffices to show that the x variables do not admit any subtours. Suppose toward a contradiction that the origin node o_k of commodity $k \in K$ is part of a subtour. Then, there cannot exist a y^k-path from the starting depot to o_k, violating constraints (5)–(8). Similarly, a destination node d_k cannot be part of any subtour. We conclude that the x variables do not admit any subtours. Hence, the one-to-one PDP is correctly formulated by (IP). □

Remark 1. The one-to-one PDP is closely related to the traveling salesman problem with pickup and delivery (TSPPDP; see, e.g., Dumitrescu et al. 2010). In fact, the TSPPDP equals the one-to-one PDP without capacity constraints. Thus, a compact ILP formulation for the TSPPDP is obtained by considering (IP) without capacity constraints (12).

3.2. One-to-One Pickup and Delivery Problem with Divisible Pickups and Deliveries

Next, we consider $(PD)^3$, the one-to-one PDP *with divisible pickups and deliveries*. Thus, contrary to the previous section, we do not restrict the number of times each node is visited. Example 1 below illustrates the potential benefits of doing so. Note that we assume that the load of a commodity $k \in K$ cannot be split. Moreover, we assume that any arc $a \in A$ in the graph can be traversed at most once. This latter assumption is not restrictive because we allow adding multiple arcs between the same pair of nodes in A. However, the user has to decide how many arcs to include when constructing the ILP formulation. Evidently, increasing this number of arcs may result in a lower objective value, but at the cost of increased computation times.

An upper bound on the maximum number of required parallel arcs is obtained as follows. Let α_v denote the number of transportation requests with their origin at location $v \in V$, and let β_v denote the number of transportation requests with their destination at location $v \in V$. Then, including at most $\min\{\alpha_{v_1} + \beta_{v_1}, \alpha_{v_2} + \beta_{v_2}\}$ arcs from node v_1 to every adjacent node v_2 suffices to achieve the lowest possible objective value. Typically, however, lower values suffice to take advantage of the divisible pickup and deliveries.

Example 1. In this example, we illustrate the potential benefits of allowing divisible pickups and deliveries. We consider a problem instance (see Figures 1 and 2) with $N = 7$ and capacity $Q = 2$. There are four commodities that need to be transported: commodities 1–4 with origin-destination pairs (1, 2), (1, 6), (4, 3), and (4, 5), respectively. All commodities have weight 1.

Figure 1 depicts an optimal solution under the assumption that every node is visited at most once. Observe that, in this solution, neither the deliveries of commodities to nodes 2 and 3 nor those to nodes 5 and 6 are combined. This is because nodes 1 and 4 can only be visited once. If we visit, for example, node 1, then we have to pick up *all* commodities at node 1, and because the vehicle capacity Q is restricted to 2, we have to deliver all commodities corresponding to node 1 before we can visit node 4.

If we instead allow divisible pickups and deliveries, then we may visit node 1 multiple times, and thus, we can pick up only a single commodity, for example, commodity 1, during the first visit to node 1. In this way, there is *sufficient remaining capacity* in the vehicle to also pick up commodity 3 at node 4, and deliver these commodities together to nodes 2 and 3. The resulting optimal solution is depicted in Figure 2. Indeed, both nodes 1 and 4 are visited twice: the first time to pick up commodities for nodes 2 and 3, and the second time to pick up commodities for nodes 5 and 6.

A natural candidate for a compact ILP formulation for the $(PD)^3$ is (IP) without constraints (13), relaxing the constraints that every node is visited exactly once. However, it turns out that, without additional constraints on (x, y, z), infeasible routes may be constructed in this formulation (see Example 2). In particular, the fact that nodes may be visited multiple times increases the complexity of the problem: based on the variables y_a^k and z_a^k it is not necessarily clear

Figure 1. Optimal Solution If All Nodes Are Visited at Most Once

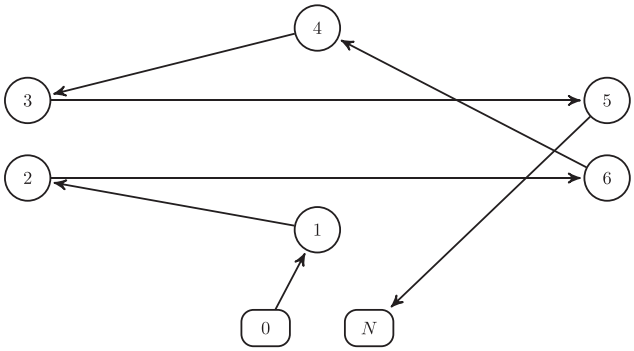
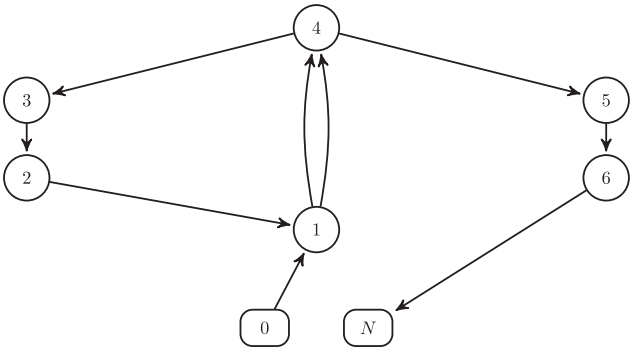


Figure 2. Optimal Solution If Nodes May Be Visited Multiple Times

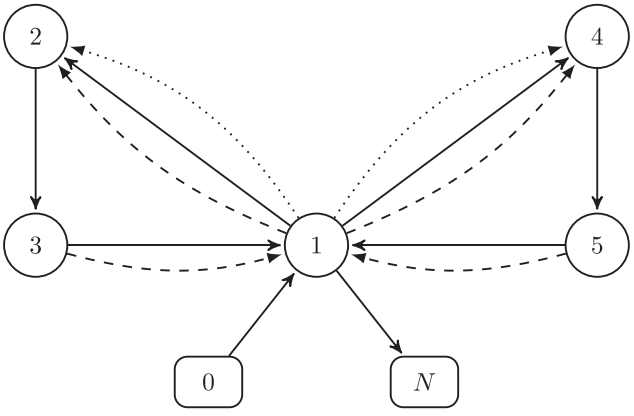


which of the multiple outgoing arcs from a node should actually be traversed first.

Example 2. Consider a similar problem as in Example 1 with $N = 6$, but with sufficient capacity Q. There are four commodities that need to be transported: commodities 1–4 with origin-destination pairs $(1,2)$, $(1,4)$, $(3,4)$, and $(5,2)$, respectively.

In Figure 3, we have displayed possible paths along which the commodities are transported from their origins to their destinations: the first two commodities are transported directly (dotted lines) from node 1 to 2 and from node 1 to 4, respectively, and the last two commodities are transported indirectly via node 1 (dashed lines) from node 3 to 4 and from node 5 to 2, respectively. The solid lines, representing the route that the vehicle traverses (with $x_a = 1$ for arcs on this route), cover all z-paths of all commodities. The problem, however, is that node 1 is visited multiple times, so that we do not know which outgoing arc, $(1,2)$ or $(1,4)$, is visited first. More importantly, a careful inspection shows that in either case we are not able to deliver all commodities. Indeed, if arc $(1,2)$ is traversed first, then commodity 4 cannot be transported from node 5 to 2, and if arc $(1,4)$ is traversed first, then the commodity 3 cannot be transported from node 3 to 4. This illustrates that additional constraints on

Figure 3. Inconsistency Between Paths



(x, y, z) are required to guarantee that feasible routes are obtained.

To address the issues illustrated in Example 2, we introduce the concept of *time consistency* in Section 3.2.1, and we derive appropriate linear constraints on y_a^k and z_a^k using this concept. Next, we use these linear constraints to construct a correct compact ILP formulation for the $(PD)^3$ in Section 3.2.2. In Section 3.2.3, we show that we do not need all time-consistency constraints to obtain a correct formulation.

3.2.1. Time Consistency. In this section, we assume that we are given binary values for x_a, y_a^k, and z_a^k, satisfying (1)–(12) of (IP). For every arc $a \in A$, we define the sets $O_a, D_a \subset K$ as

- $O_a := \{k \in K : y_a^k = 1\}$,
- $D_a := \{k \in K : y_a^k + z_a^k = 1\}$.

That is, O_a contains the commodities $k \in K$ that are not yet picked up when arc $a \in A$ is traversed, whereas D_a contains the commodities $k \in K$ that are not yet delivered when arc $a \in A$ is traversed. The set D_a includes both commodities $k \in K$ that are not yet picked up and commodities $k \in K$ that are picked up but not yet delivered. Thus, $O_a \subseteq D_a$ for every arc $a \in A$.

Intuitively, it is clear that if $O_{a_1} \supset O_{a_2}$ and $D_{a_1} \supset D_{a_2}$, then arc a_1 needs to be traversed before arc a_2 because more commodities still need to be picked up and fewer commodities still need to be delivered when traversing arc a_1 compared with when traversing arc a_2. We use this intuition to construct a formal precedence relation between two arcs a_1 and a_2, where $a_1 \preceq a_2$ will have the interpretation that a_1 can be traversed before a_2. In case $a_1 \not\preceq a_2$ and $a_2 \not\preceq a_1$, then we call arcs a_1 and a_2 *time inconsistent* because neither arc a_1 can be traversed before a_2 nor a_2 before a_1. We show in Theorem 1 that we can exclude time inconsistency between two arcs by using appropriate linear constraints on y_a^k and z_a^k. First, however, we introduce our precedence relation.

Definition 1. For every pair of arcs $a_1, a_2 \in A$ we say that a_1 *precedes* a_2, denoted $a_1 \preceq a_2$, if and only if $O_{a_1} \supseteq O_{a_2}$ and $D_{a_1} \supseteq D_{a_2}$. If $a_1 \preceq a_2$ and $a_2 \preceq a_1$, then we write $a_1 \simeq a_2$. Moreover, we say that a_1 *strictly precedes* a_2, denoted $a_1 \prec a_2$, if $a_1 \preceq a_2$ and $a_1 \neq a_2$. That is, if $O_{a_1} \supseteq O_{a_2}$ and $D_{a_1} \supseteq D_{a_2}$, and at least one of the two inclusions is strict.

The interpretation of $a_1 \prec a_2$ is that a_1 has to be traversed before a_2. For example, if $O_{a_1} \supset O_{a_2}$, then this is true because there exists a commodity k that is already picked up when traversing a_2 but not when traversing a_1. Similarly, a_1 has to be traversed before a_2 if $D_{a_1} \supset D_{a_2}$, and thus, a commodity k has already been delivered when traversing a_2 but not when traversing a_1. The interpretation of $a_1 \simeq a_2$ is that $O_{a_1} = O_{a_2}$

and $D_{a_1} = D_{a_2}$, and thus, exactly the same commodities have to be picked up and have to be delivered when traversing arc a_1 as when traversing arc a_2. Such arcs a_1 and a_2 may be traversed in arbitrary order. Finally, $a_1 \preceq a_2$ implies that either $a_1 \prec a_2$ or $a_1 \simeq a_2$. In either case, arc a_1 can be traversed before arc a_2.

Definition 2. For every pair of arcs $a_1, a_2 \in A$, we call a_1 and a_2 *time consistent* if and only if $a_1 \preceq a_2$ or $a_2 \preceq a_1$. That is, if either a_1 precedes a_2 or a_2 precedes a_1. We call a_1 and a_2 *time inconsistent* if $a_1 \not\preceq a_2$ and $a_2 \not\preceq a_1$.

Remark 2. Note that, when $x_a = 0$, constraints (4) imply that $y_a^k = z_a^k = 0$ for all $k \in K$, and thus, $O_a = D_a = \emptyset$. Thus, if $x_a = 0$, then by definition arc a is time consistent with any other arc.

If $a_1 \not\preceq a_2$ for two arcs $a_1, a_2 \in A$, then arc a_1 cannot be traversed before a_2. Hence, if a_1 and a_2 are time inconsistent, that is, $a_1 \not\preceq a_2$ and $a_2 \not\preceq a_1$, then there does not exist a feasible pickup and delivery route corresponding to (x, y, z). Hence, at the least, we require that all arcs are pairwise time consistent.

Example 3. In the simple problem of Example 2, it turns out that the arcs $(1,2)$ and $(1,4)$ are not time consistent for the (x, y, z) solution considered there. In Figures 4 and 5, we see the y-path (dashed) and z-path (solid) of commodity 3 and commodity 4, respectively. Hence, $O_{(1,2)} = \{3\}$ and $O_{(1,4)} = \{4\}$. Clearly, $O_{(1,2)} \not\subseteq O_{(1,4)}$ and $O_{(1,4)} \not\subseteq O_{(1,2)}$, and thus, $(1,2)$ and $(1,4)$ are not time consistent. Intuitively, $O_{(1,2)} = \{3\}$ implies that when traversing arc $(1,2)$ commodity 4 is already picked up, but commodity 3 is not, whereas $O_{(1,4)} = \{4\}$ implies that when traversing arc $(1,4)$ commodity 3 is already picked up, but commodity 4 is not. This is an infeasible solution for the (PD)[3].

In the remainder of this section, we derive linear constraints on y_a^k and z_a^k to guarantee that a_1 and a_2 are time consistent. For notational convenience we introduce the auxiliary variable $\bar{z}_a^k$ for all $k \in K$ and $a \in A$, defined as

$$\bar{z}_a^k := y_a^k + z_a^k, \quad k \in K, a \in A.$$

This binary variable equals 1 if commodity k still needs to be picked up or delivered when arc $a \in A$ is traversed. In other words, $\bar{z}_a^k = 1$ if and only if $k \in D_a$.

First we model when a_1 precedes a_2.

Figure 4. Path of Commodity 3 with Origin-Destination Pair $(3,4)$

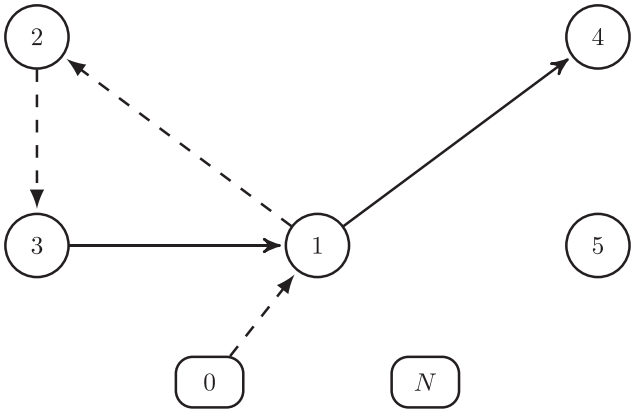
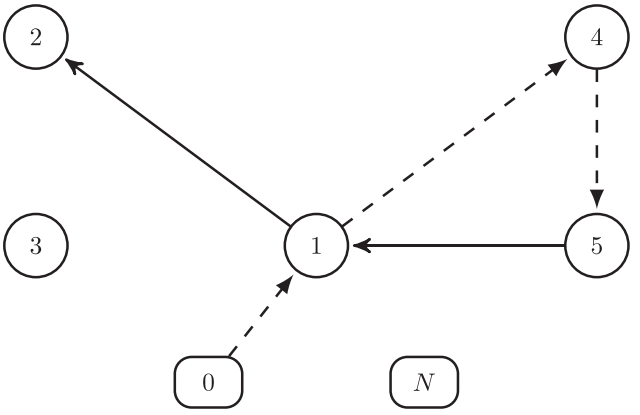


Figure 5. Path of Commodity 4 with Origin-Destination Pair $(5,2)$



Lemma 1. *Let two arcs $a_1, a_2 \in A$ be given. Then, $a_1 \preceq a_2$ if and only if $y_{a_1}^k \geq y_{a_2}^k$ and $\bar{z}_{a_1}^k \geq \bar{z}_{a_2}^k$ for all $k \in K$.*

Proof. This follows directly from the definitions of $a_1 \preceq a_2$, O_a, and D_a. □

Thus, by Lemma 1 we can guarantee that a_1 precedes a_2 by imposing the constraints $y_{a_1}^k \geq y_{a_2}^k$ and $\bar{z}_{a_1}^k \geq \bar{z}_{a_2}^k$ for all $k \in K$. In our optimization problem, however, we do not know beforehand whether a_1 precedes a_2 or a_2 precedes a_1, but only that they need to be time consistent. This means that to be able to use the constraints in an ILP we have to rewrite the disjunction

$$\left(\bigwedge_{k\in K}\left(\left(y_{a_1}^k \geq y_{a_2}^k\right) \wedge \left(\bar{z}_{a_1}^k \geq \bar{z}_{a_2}^k\right)\right)\right) \vee \left(\bigwedge_{k\in K}\left(\left(y_{a_2}^k \geq y_{a_1}^k\right) \wedge \left(\bar{z}_{a_2}^k \geq \bar{z}_{a_1}^k\right)\right)\right) \tag{14}$$

as a conjunction. To do so, we first introduce the following auxiliary lemma.

Lemma 2. *Let $u_1^k, u_2^k, w_1^k, w_2^k$ be binary variables for all $k \in K$. Then,*

$$u_1^k \geq u_2^k \ \forall k \in K \quad \text{or} \quad w_2^k \geq w_1^k \ \forall k \in K,$$
$$\Leftrightarrow u_2^{k_1} + w_1^{k_2} \leq 1 + u_1^{k_1} + w_2^{k_2} \quad \forall k_1, k_2 \in K.$$

Proof. Note that ($u_1^k \geq u_2^k \ \forall k \in K$ or $w_2^k \geq w_1^k \ \forall k \in K$) is equivalent to the disjunction

$$\left(\bigwedge_{k\in K}(u_1^k \geq u_2^k)\right) \bigvee \left(\bigwedge_{k\in K}(w_2^k \geq w_1^k)\right), \tag{15}$$

which can be rewritten as the conjunction $\wedge_{k_1,k_2\in K}((u_1^{k_1} \geq u_2^{k_1}) \vee (w_2^{k_2} \geq w_1^{k_2}))$. To see this, observe that if the disjunction in (15) holds, then either $u_1^k \geq u_2^k$ for all $k \in K$ or $w_2^k \geq w_1^k$ for all $k \in K$, and thus, $(u_1^{k_1} \geq u_2^{k_1}) \vee (w_2^{k_2} \geq w_1^{k_2})$ holds for all $k_1, k_2 \in K$. On the other hand, if the conjunction holds, then it is not possible that the disjunction does not hold, because otherwise there would exist $k_1 \in K$ and $k_2 \in K$ such that both $u_1^{k_1} \not\geq u_2^{k_1}$ and $w_2^{k_2} \not\geq w_1^{k_2}$, contradicting that the conjunction is true.

We conclude the proof by observing that, for every $k_1, k_2 \in K$ in the conjunction,

$$\begin{aligned} & u_1^{k_1} \geq u_2^{k_1} \quad \text{or} \quad w_2^{k_2} \geq w_1^{k_2}, \\ \Leftrightarrow\ & u_2^{k_1} + w_1^{k_2} \leq 1 + u_1^{k_1} + w_2^{k_2}, \end{aligned}$$

because $u_1^{k_1}, u_2^{k_1}, w_1^{k_2}$ and $w_2^{k_2}$ are binary variables. □

Now we are ready to define the constraints necessary to impose time consistency on any two arcs.

Theorem 1. *Let x_a, y_a^k, and z_a^k for $k \in K$ and $a \in A$ satisfy constraints (1)–(12) of (IP). Suppose that $a_1, a_2 \in A$ are given. Then, a_1 and a_2 are time consistent, that is, $a_1 \preceq a_2$ or $a_2 \preceq a_1$, if and only if for all $k_1, k_2 \in K$:*

i. $y_{a_2}^{k_1} + y_{a_1}^{k_2} \leq 1 + y_{a_1}^{k_1} + y_{a_2}^{k_2}$,
ii. $y_{a_2}^{k_1} + \bar{z}_{a_1}^{k_2} \leq 1 + y_{a_1}^{k_1} + \bar{z}_{a_2}^{k_2}$,
iii. $\bar{z}_{a_2}^{k_1} + y_{a_1}^{k_2} \leq 1 + \bar{z}_{a_1}^{k_1} + y_{a_2}^{k_2}$,
iv. $\bar{z}_{a_2}^{k_1} + \bar{z}_{a_1}^{k_2} \leq 1 + \bar{z}_{a_1}^{k_1} + \bar{z}_{a_2}^{k_2}$.

Proof. Because two arcs a_1 and a_2 are time consistent if and only if their corresponding y and z variables satisfy the disjunction in (14), we rewrite this disjunction as the conjunction of

$$\left(\bigwedge_{k\in K}\left(y_{a_1}^k \geq y_{a_2}^k\right)\right) \bigvee \left(\bigwedge_{k\in K}\left(y_{a_2}^k \geq y_{a_1}^k\right)\right), \tag{16}$$

$$\left(\bigwedge_{k\in K}\left(y_{a_1}^k \geq y_{a_2}^k\right)\right) \bigvee \left(\bigwedge_{k\in K}\left(\bar{z}_{a_2}^k \geq \bar{z}_{a_1}^k\right)\right), \tag{17}$$

$$\left(\bigwedge_{k\in K}\left(\bar{z}_{a_1}^k \geq \bar{z}_{a_2}^k\right)\right) \bigvee \left(\bigwedge_{k\in K}\left(y_{a_2}^k \geq y_{a_1}^k\right)\right), \tag{18}$$

and

$$\left(\bigwedge_{k\in K}\left(\bar{z}_{a_1}^k \geq \bar{z}_{a_2}^k\right)\right) \bigvee \left(\bigwedge_{k\in K}\left(\bar{z}_{a_2}^k \geq \bar{z}_{a_1}^k\right)\right). \tag{19}$$

The result follows directly by applying Lemma 2 to (16)–(19) separately. For example, applying Lemma 2 with $u_1^k := y_{a_1}^k, u_2^k := y_{a_2}^k, w_1^k := y_{a_1}^k$, and $w_2^k := y_{a_2}^k$ to (16) yields the constraints in (i). Similarly, the constraints in (ii)–(iv) are obtained from (17)–(19), respectively. □

Remark 3. In terms of O_a and D_a, the set of constraints in Theorem 1 can be equivalently stated as

i. $O_{a_1} \subseteq O_{a_2}$ or $O_{a_1} \supseteq O_{a_2}$,
ii. $O_{a_1} \subseteq O_{a_2}$ or $D_{a_1} \supseteq D_{a_2}$,
iii. $D_{a_1} \subseteq D_{a_2}$ or $O_{a_1} \supseteq O_{a_2}$,
iv. $D_{a_1} \subseteq D_{a_2}$ or $D_{a_1} \supseteq D_{a_2}$.

Observe that constraints (i)–(iv) in Theorem 1 are defined for all $a_1, a_2 \in A$ and for all $k_1, k_2 \in K$, meaning that in total there are $4|A|^2|K|^2$ such constraints. This number is significantly large already for medium-sized problems. However, we show in Section 3.2.3 that we do not need all these constraints to guarantee that all $a_1, a_2 \in A$ are time consistent. Moreover, in practice not all constraints may be relevant. Hence, in Section 4.3, we also explain how we iteratively add them to our problem within a branch-and-bound framework.

3.2.2. A Compact ILP Formulation for the (PD)^3. Armed with the time-consistency constraints of Theorem 1, we are ready to define a new compact ILP formulation for the (PD)^3, which we label $(IP)^3$:

$$\begin{aligned} \min_{x,y,z} \quad & \sum_{a\in A} c_a x_a \\ \text{s.t.} \quad & (1)\text{--}(12), \\ & y_{a_2}^{k_1} + y_{a_1}^{k_2} \leq 1 + y_{a_1}^{k_1} + y_{a_2}^{k_2}, \\ & \qquad a_1, a_2 \in A, k_1, k_2 \in K, && (20) \\ & y_{a_2}^{k_1} + \bar{z}_{a_1}^{k_2} \leq 1 + y_{a_1}^{k_1} + \bar{z}_{a_2}^{k_2} \\ & \qquad a_1, a_2 \in A, k_1, k_2 \in K, && (21) \\ & \bar{z}_{a_2}^{k_1} + y_{a_1}^{k_2} \leq 1 + \bar{z}_{a_1}^{k_1} + y_{a_2}^{k_2} \\ & \qquad a_1, a_2 \in A, k_1, k_2 \in K, && (22) \\ & \bar{z}_{a_2}^{k_1} + \bar{z}_{a_1}^{k_2} \leq 1 + \bar{z}_{a_1}^{k_1} + \bar{z}_{a_2}^{k_2} \\ & \qquad a_1, a_2 \in A, k_1, k_2 \in K, && (23) \\ & x_a, y_a^k, z_a^k \in \{0,1\} \quad a \in A, k \in K. \end{aligned}$$

Observe that the above formulation is equal to that for the one-to-one PDP without constraints (13) but with time-consistency constraints (20)–(23) added. In the remainder of this section we show that $(IP)^3$ is a correct formulation of the (PD)^3.

The time-consistency constraints in (20)–(23) guarantee that all arcs $a \in A$ with $x_a = 1$ are pairwise time consistent, so that they can be ordered using the precedence relation of Definition 1. In fact, this precedence relation defines a *total order* on all arcs $a \in A$. For example, transitivity holds because

$$a_1 \preceq a_2 \text{ and } a_2 \preceq a_3 \Rightarrow a_1 \preceq a_3.$$

This implies that there exists a function τ that assigns a value to each arc $a \in A$ with $x_a = 1$ such that $\tau(a_1) < \tau(a_2)$ if and only if $a_1 \prec a_2$, and $\tau(a_1) = \tau(a_2)$ if and only if $a_1 \simeq a_2$.

If the value $\tau(a)$ of every arc $a \in A$ with $x_a = 1$ is unique, so that $\tau(a_1) < \cdots < \tau(a_L)$, then $a_1 \prec \cdots \prec a_L$, and thus, the order in which arcs need to be traversed is completely determined by the function τ. For this reason, we call τ an *order function*. Moreover, if $a_1 \prec \cdots \prec a_L$, then we assume that this order function τ is such that $\tau(a) = l$ if and only if arc a is the lth arc that needs to traversed. That is, $\tau(a_1) = 1$, $\tau(a_2) = 2, \ldots, \tau(a_L) = L$.

In general, however, it is possible that $a_i \simeq a_j$ so that $\tau(a_i) = \tau(a_j)$, and thus, arcs a_i and a_j may be traversed in arbitrary order. In this case, the order function τ defines the order in which *group of arcs*, with a_i and a_j in the same group if $\tau(a_i) = \tau(a_j)$, have to be traversed. Within groups, however, arcs may be traversed in arbitrary order. We assume that the order function τ is such that $\tau(a) = l$ if and only if arc a is in the lth group that needs to be traversed. Without loss of generality we assume that there are L such groups.

Throughout this paper we refer to τ as the order function consistent with the precedence relation of Definition 1.

Definition 3. Let τ be the order function defined on the arcs $a \in A$ with $x_a = 1$ that is consistent with the precedence relation of Definition 1. Then, for every $l = 1, \ldots, L$, we define

$$A_l := \{a \in A : x_a = 1 \text{ and } \tau(a) = l\}.$$

That is, A_l represents the lth group of arcs that need to be traversed. Moreover, because the y_a^k and z_a^k values are the same for all arcs $a \in A_l$ we define O_l and D_l as

- $O_l = O_a$ for all $a \in A_l$,
- $D_l = D_a$ for all $a \in A_l$.

It should be clear that the sets O_l and D_l completely specify the order in which pickups and deliveries are carried out. Indeed, $O_1 = K$ and if $k \notin O_2$, then this implies that commodity k is picked up first. In general, the time consistency of the arcs imposes the following order on the sets O_l and D_l for $l = 1, \ldots, L$.

Lemma 3. *For $l_1, l_2 = 1, \ldots, L$ with $l_1 \le l_2$, we have*

$$O_{l_1} \supseteq O_{l_2} \text{ and } D_{l_1} \supseteq D_{l_2}.$$

Moreover, if $l_1 < l_2$, then at least one of the inclusions is strict.

If every set A_l, $l = 1, \ldots, L$, consists of a single arc a_l, then $\tau(a_l) = l$ for every $l = 1, \ldots, L$, and thus, a_l is the lth arc that is traversed. Hence, the only pickup and delivery route corresponding to (x, y, z) that is possibly feasible in practice is to traverse $a_1, \ldots, a_L$ in this order. To prove that this route is feasible we need to show that a_1 starts at the starting depot, a_L ends at the ending depot, and $t(a_l) = s(a_{l+1})$ for all $l = 1, \ldots, L-1$. In other words, we need to show that there exist nodes $v_l \in V$, $l = 1, \ldots, L-1$, that connect a_l and a_{l+1}, that is, $v_l = t(a_l)$ and $v_l = s(a_{l+1})$.

In general, however, the arc set A_l may consist of multiple arcs, which can be traversed in arbitrary order. Nevertheless, we show that there exist nodes $v_l \in \bar{V}$, $l = 0, \ldots, L$, with $v_0 = 0$ and $v_L = N$, such that for every $l = 1, \ldots, L$, there exists a path P_l, traversing arcs in A_l only, starting at node v_{l-1} and ending at node v_l. To prove this, we introduce $\delta_l(v)$, the difference between the number of outgoing and incoming arcs $a \in A_l$ from $v \in \bar{V}$, and we analyze properties of $\delta_l(v)$.

Definition 4. Let $l \in \{1, \ldots, L\}$ and $v \in \bar{V}$ be given. Then, we define $\delta_l(v)$ as

$$\delta_l(v) := \sum_{a \in A_l : s(a) = v} x_a - \sum_{a \in A_l : t(a) = v} x_a.$$

Observe that we can interpret $\sum_{\lambda=1}^{l} \delta_\lambda(v)$ as the difference between the number of outgoing and incoming arcs at v over all arcs a with $\tau(a) \le l$. It turns out that, for every $l = 1, \ldots, L$, there is a unique node $v_l \in \bar{V}$ with $v_l \neq 0$ for which $\sum_{\lambda=1}^{l} \delta_\lambda(v_l) = -1$. In case the arc set A_l consists of a single arc a_l, this node v_l equals $t(a_l)$. In general, this node v_l is the ending node of path P_l.

Lemma 4. *For every $l = 1, \ldots, L$, there exists a node $v_l \in \bar{V}$ with $v_l \neq 0$ such that*

$$\sum_{\lambda=1}^{l} \delta_\lambda(v) = \begin{cases} 1, & \text{if } v = 0, \\ 0, & \text{if } v \neq 0, v_l, \\ -1, & \text{if } v = v_l. \end{cases}$$

Proof. Let $l = 1, \ldots, L-1$ be given. Then, by Lemma 3, we have $O_l \supseteq O_{l+1}$ and $D_l \supseteq D_{l+1}$, and at least one of these inclusions is strict. Suppose that there exists a commodity $k \in O_l$ such that $k \notin O_{l+1}$. Then, again by Lemma 3,

$$\begin{cases} k \in O_\lambda, & \text{if } \lambda \le l, \\ k \notin O_\lambda, & \text{if } \lambda \ge l+1. \end{cases}$$

Hence, *commodity k is an identifier for $\bigcup_{\lambda=1}^{l} A_\lambda$* in the sense that $y_a^k = 1$ if and only if $a \in \bigcup_{\lambda=1}^{l} A_\lambda$, and thus $\tau(a) \le l$. Because y_a^k satisfies the flow-conservation constraints (6)–(8), it follows that the arcs in $\bigcup_{\lambda=1}^{l} A_\lambda$ define a path from the starting depot zero to the origin o_k of commodity k and possibly several additional cycles. Hence,

$$\sum_{\lambda=1}^{l} \delta_\lambda(v) = \begin{cases} 1, & \text{if } v = 0, \\ 0, & \text{if } v \neq 0, o_k, \\ -1, & \text{if } v = o_k. \end{cases} \tag{24}$$

By similar arguments we can show that if $k \in D_l \backslash O_l$ but $k \notin D_{l+1}$, then (24) holds with o_k replaced by d_k. Moreover, (24) may hold for multiple commodities k_1 and k_2, but only if $o_{k_1} = o_{k_2}$. Hence, in general there exists a unique node $v_l \in V$ with $v_l \neq 0$ such that

$$\sum_{\lambda=1}^{l} \delta_l(v) = \begin{cases} 1, & \text{if } v = 0, \\ 0, & \text{if } v \neq 0, v_l, \\ -1, & \text{if } v = v_l. \end{cases} \tag{25}$$

Finally, note that, for $l = L$, the set $\bigcup_{\lambda=1}^{l} A_\lambda$ contains all arcs $a \in A$ with $x_a = 1$. By constraints (1)–(3), the indegree of every node $v \in \bar{V}$ equals its outdegree except for $v = 0$ and $v = N$. Hence, the equalities in (25) also hold for $l = L$, with $v_L = N$. □

Now we are ready to prove that $(IP)^3$ is a correct ILP formulation of the $(PD)^3$.

Theorem 2. *Let (x, y, z) be a feasible solution of $(IP)^3$. Then, there exists a practically feasible route, carrying out all pickups and deliveries subject to the capacity constraints, that only traverses arcs $a \in A$ with $x_a = 1$.*

Proof. Let (x, y, z) be a feasible solution to $(IP)^3$, and consider the order function τ consistent with the precedence relation of Definition 1, taking values in $\{1, \ldots, L\}$. By Lemma 4, there exist nodes $v_l \in \bar{V}$, $l = 0, \ldots, L$, with $v_0 = 0$ and $v_L = N$, such that for every $l = 1, \ldots, L$

$$\sum_{\lambda=1}^{l} \delta_\lambda(v) = \begin{cases} 1, & \text{if } v = 0, \\ 0, & \text{if } v \neq 0, v_l, \\ -1, & \text{if } v = v_l. \end{cases}$$

Because

$$\delta_l(v) = \sum_{\lambda=1}^{l} \delta_\lambda(v) - \sum_{\lambda=1}^{l-1} \delta_\lambda(v), \qquad l = 1, \ldots, L,\ v \in \bar{V},$$

this implies that, for every $l = 1, \ldots, L$, either $\delta_l(v) = 0$ for all $v \in \bar{V}$ if $v_{l-1} = v_l$, or

$$\delta_l(v) = \begin{cases} 1, & \text{if } v = v_{l-1}, \\ 0, & \text{if } v \neq v_{l-1}, v_l, \\ -1, & \text{if } v = v_l, \end{cases} \qquad \text{if } v_{l-1} \neq v_l.$$

In both cases, it follows from Euler path theory that there exists a path P_l from v_{l-1} to v_l traversing arcs in A_l only. Note that this path does not necessarily have to traverse all arcs in A_l.

By successively traversing paths P_l, starting with $l = 1$ until $l = L$, we construct a path P from the starting depot to the ending depot consisting of arcs with $x_a = 1$ only. Moreover, the path P is consistent with the order function τ in the sense that arcs with lower value are always traversed before arcs with a higher value. This immediately proves that each commodity is actually picked up before it is delivered, and thus, the path P corresponds to a practically feasible route traversing only arcs with $x_a = 1$. □

In the proof of Theorem 2, we do not show that the practical route corresponding to a feasible solution (x, y, z) in $(IP)^3$ actually traverses *all* arcs with $x_a = 1$. This is also not necessarily the case. However, the practical route corresponding to any *optimal* solution (x, y, z) in $(IP)^3$ does traverse all arcs with $x_a = 1$, at least if $c_a > 0$ for all $a \in A$.

3.2.3. Reduced Number of Time-Consistency Constraints.

In this section, we show that we do not need all the time-consistency constraints in (20)–(23) of $(IP)^3$ to obtain a correct formulation of the $(PD)^3$. In fact, we show that we only need constraints (20)–(23) for each $a_1, a_2 \in A$ with either $s(a_1) = s(a_2)$ or $t(a_1) = t(a_2)$. That is, we only need to make sure that, for all nodes $v \in V$, all outgoing arcs from this node v are pairwise time consistent and all incoming arcs are pairwise time consistent. This yields the following formulation $(IP_R)^3$:

$$\min_{x,y,z} \quad \sum_{a \in A} c_a x_a$$

$$\text{s.t.} \quad (1)\text{--}(12),$$

$$y_{a_2}^{k_1} + y_{a_1}^{k_2} \leq 1 + y_{a_1}^{k_1} + y_{a_2}^{k_2} \quad a_1, a_2 \in A \text{ with } st(a_1) = st(a_2), k_1, k_2 \in K, \tag{26}$$

$$y_{a_2}^{k_1} + \bar{z}_{a_1}^{k_2} \leq 1 + y_{a_1}^{k_1} + \bar{z}_{a_2}^{k_2} \quad a_1, a_2 \in A \text{ with } st(a_1) = st(a_2), k_1, k_2 \in K, \tag{27}$$

$$\bar{z}_{a_2}^{k_1} + y_{a_1}^{k_2} \leq 1 + \bar{z}_{a_1}^{k_1} + y_{a_2}^{k_2} \quad a_1, a_2 \in A \text{ with } st(a_1) = st(a_2), k_1, k_2 \in K, \tag{28}$$

$$\bar{z}_{a_2}^{k_1} + \bar{z}_{a_1}^{k_2} \leq 1 + \bar{z}_{a_1}^{k_1} + \bar{z}_{a_2}^{k_2} \quad a_1, a_2 \in A \text{ with } st(a_1) = st(a_2), k_1, k_2 \in K, \tag{29}$$

$$x_a, y_a^k, z_a^k \in \{0, 1\} \qquad a \in A, k \in K,$$

where $\text{st}(a_1) = \text{st}(a_2)$ if and only if $s(a_1) = s(a_2)$ or $t(a_1) = t(a_2)$. To show that the above formulation is correct, we show that, for any feasible (x, y, z) in $(IP_R)^3$, we can construct a feasible solution $(\hat{x}, \hat{y}, \hat{z})$ to the original formulation $(IP)^3$ with $\hat{x}_a \leq x_a$ for all $a \in A$. The reverse is obviously true because $(IP_R)^3$ is a relaxation of $(IP)^3$. Before we discuss how to construct $(\hat{x}, \hat{y}, \hat{z})$ based on (x, y, z), we first derive a time-consistency property of a feasible solution (x, y, z) of $(IP_R)^3$. It turns out that not only arcs $a_1, a_2 \in A$ with $s(a_1) = s(a_2)$ or $t(a_1) = t(a_2)$ are time consistent, but also arcs with $t(a_1) = s(a_2)$.

To prove this, let $v \in V$ be given, and let $a_1^{\text{in}}, \ldots, a_m^{\text{in}}$ and $a_1^{\text{out}}, \ldots, a_m^{\text{out}}$ denote the incoming arcs and outgoing arcs with $x_a = 1$ at v, respectively. Note that the number of incoming arcs equals the number of outgoing arcs. Moreover, because the outgoing arcs $a_1^{\text{out}}, \ldots, a_m^{\text{out}}$ of a node $v \in V$ are pairwise time consistent, there exists a total order among those arcs, so that without loss of generality we have $a_1^{\text{out}} \preceq \cdots \preceq a_m^{\text{out}}$. Similarly, there exists a total order among incoming arcs: $a_1^{\text{in}} \preceq \cdots \preceq a_m^{\text{in}}$. The next lemma, however, shows that also $a_i^{\text{in}} \preceq a_i^{\text{out}}$ for all $i = 1, \ldots, m$.

Lemma 5. *Let (x, y, z) be a solution to $(IP_R)^3$, and let $a_1^{in} \preceq \cdots \preceq a_m^{in}$ and $a_1^{out} \preceq \cdots \preceq a_m^{out}$ denote the incoming and outgoing arcs at some node $v \in V$. Then,*

$$a_i^{in} \preceq a_i^{out}, \quad \text{for all } i = 1, \ldots, m.$$

Proof. Let node $v \in V$ be given with incoming and outgoing arcs a_i^{in} and a_i^{out}, $i = 1, \ldots, m$, respectively. Let $i = 1, \ldots, m$ be given, and assume toward a contradiction that $a_i^{\text{in}} \not\preceq a_i^{\text{out}}$. This implies that there exists a commodity $k \in K$ such that either

i. $k \in O_{a_i^{\text{out}}}$ and $k \notin O_{a_i^{\text{in}}}$, or
ii. $k \in D_{a_i^{\text{out}}} \setminus O_{a_i^{\text{out}}}$ and $k \notin D_{a_i^{\text{in}}}$.

In the first case, this implies that $k \in O_{a_j^{\text{out}}}$ for all $j \le i$ and $k \notin O_{a_j^{\text{in}}}$ for all $j \ge i$, because $a_1^{\text{in}} \preceq \cdots \preceq a_m^{\text{in}}$ and $a_1^{\text{out}} \preceq \cdots \preceq a_m^{\text{out}}$. However, this contradicts a constraint in either (7) or (8) because it implies that $\sum_{a:s(a)=v} y_a^k \ge i$ whereas $\sum_{a:t(a)=v} y_a^k \le i - 1$, so that

$$\sum_{a:t(a)=v} y_a^k - \sum_{a:s(a)=v} y_a^k \le -1.$$

The second case is not possible either by similar arguments. Hence, we conclude that $a_i^{\text{in}} \preceq a_i^{\text{out}}$ for all $i = 1, \ldots, m$. □

Lemma 5 shows that we can link any incoming arc a_i^{in} of a node $v \in V$ in a natural way to an outgoing arc a_i^{out} with $a_i^{\text{in}} \preceq a_i^{\text{out}}$. In fact, in this way we can assign a unique outgoing arc to every incoming arc at every node. Starting with the single outgoing arc a_1 from the starting depot, this allows us to construct a path of time-consistent arcs $a_1 \preceq \cdots \preceq a_r$ to the ending depot, by iteratively traversing the unique outgoing arc a_i corresponding to the incoming arc a_{i-1}. The path has to end at the ending depot because for every node $v \in V$ the number of incoming arcs equals the number of outgoing arcs.

In general, it is not necessary that the path $a_1 \preceq \cdots \preceq a_r$ traverses all arcs with $x_a = 1$. This is why we construct a solution $(\hat{x}, \hat{y}, \hat{z})$ that is feasible in $(IP)^3$ with $\hat{x}_a = 1$ if and only if $a \in \{a_1, \ldots, a_r\}$. The key observation used to construct $(\hat{x}, \hat{y}, \hat{z})$ is that the unique correspondence between incoming and outgoing arcs for (x, y, z) defines a path $a_1 \preceq \cdots \preceq a_r$ and several cycles. However, for every cycle $C = \{a_1, \ldots, a_c, a_1\}$, we must have $a_1 \preceq \cdots \preceq a_c \preceq a_1$, and thus, $a_1 \simeq \cdots \simeq a_c$. Hence, it is not surprising that these cycles turn out to be irrelevant for the practically feasible route corresponding to (x, y, z).

Theorem 3. *Let (x, y, z) be a solution to formulation $(IP_R)^3$. Then there exists a corresponding solution $(\hat{x}, \hat{y}, \hat{z})$ with $\hat{x}_a \le x_a$ for all $a \in A$ that is feasible in the original formulation $(IP)^3$.*

Proof. Consider the path $a_1 \preceq \cdots \preceq a_r$ from starting depot to ending depot defined by the unique correspondence between incoming and outgoing arcs from nodes, induced by Lemma 5. The remaining arcs $a \in A$ with $x_a = 1$ define several cycles of the form $C := \{a_1, \ldots, a_c, a_1\}$. Consider one such cycle. We show that the solution $(\tilde{x}, \tilde{y}, \tilde{z})$, defined as

$$\begin{aligned} \tilde{x}_a = \tilde{y}_a^k = \tilde{z}_a^k = 0 \quad & \forall a \in C, \forall k \in K, \\ \tilde{x}_a = x_a, \tilde{y}_a^k = y_a^k, \tilde{z}_a^k = z_a^k \quad & \forall a \notin C, \forall k \in K, \end{aligned}$$

is also feasible in $(IP_R)^3$. We do so by arguing that each constraint of $(IP_R)^3$ holds for $(\tilde{x}, \tilde{y}, \tilde{z})$.

Constraints (1) and (2) are not affected because the arcs in the cycle cannot be adjacent to the starting or ending depots. Obviously, constraints (3) are still satisfied, because for every node $v \in V$

$$\sum_{a \in C: s(a)=v} x_a = \sum_{a \in C: t(a)=v} x_a.$$

Constraints (4) are true by construction for every $a \in C$ and $a \notin C$, and in (5) and (6), similarly to as in (1) and (2), arcs adjacent to the starting and ending depot are not affected. Next, because

$$\begin{aligned} \sum_{a:t(a)=v} y_a^k - \sum_{a:s(a)=v} y_a^k = & \left(\sum_{a \notin C: t(a)=v} y_a^k - \sum_{a \notin C: s(a)=v} y_a^k \right) \\ & + \left(\sum_{a \in C: t(a)=v} y_a^k - \sum_{a \in C: s(a)=v} y_a^k \right) \end{aligned}$$

and the latter term equals zero, constraints (7) and (8) also hold for $(\tilde{x}, \tilde{y}, \tilde{z})$. An identical argument applies to (9)–(11). Constraints (12) hold by construction for all $a \in C$ and $a \notin C$, and finally, the time-consistency constraints (26)–(29) hold because if $a_1 \in C$, then $\tilde{y}_{a_1}^{k_1} = \tilde{y}_{a_1}^{k_2} = 0$, so that the first constraint reduces to $\tilde{y}_{a_2}^{k_1} \le 1 + \tilde{y}_{a_2}^{k_2}$, which always holds. Similar arguments apply to the other constraints and $a_2 \in C$. Finally, the binary restrictions on $(\tilde{x}, \tilde{y}, \tilde{z})$ are trivially satisfied.

By applying the above arguments to each cycle, it follows immediately that $(\hat{x}, \hat{y}, \hat{z})$ defined as

$$\begin{aligned} \hat{x}_a = x_a = 1, \ \hat{y}_a^k = y_a^k, \ \hat{z}_a^k = z_a^k, \quad & a \in \{a_1, \ldots, a_r\}, \ k \in K, \\ x_a = y_a^k = z_a^k = 0, \quad & \text{otherwise}, \end{aligned}$$

is feasible in $(IP_R)^3$. Moreover, because all arcs with $\hat{x}_a = 1$ are pairwise time consistent, we conclude that $(\hat{x}, \hat{y}, \hat{z})$ is also feasible for $(IP)^3$. □

We conclude that we do not have to add all time-consistency constraints (20)–(23) to our model of $(PD)^3$, but only those for adjacent arcs $a_1, a_2 \in A$ for which either $s(a_1) = s(a_2)$ or $t(a_1) = t(a_2)$.

4. Computational Results

The ILP formulations described in Section 3 for the one-to-one PDP and the $(PD)^3$ are implemented in C++ and solved using CPLEX 12.8 on an Intel Xeon E5 2680v3 CPU (2.5 GHz). We made use of four parallel running threads on a single core, a setup nowadays available on any common desktop machine. In the remainder of this section, we first describe the test instances from the literature that we use for our numerical experiments. Next, we investigate the performance of our ILP formulation (IP) for the one-to-one PDP, and we compare our results with those from Hernández-Pérez and Salazar-González (2009) and Gouveia and Ruthmair (2015). Finally, we turn to the $(PD)^3$ and present numerical results for divisible pickups and deliveries. Detailed results of our numerical experiments are available in the online appendix.

4.1. Test Instances

Because there are no test instances available in the literature for exact solution methods for the $(PD)^3$, we use existing test instances from Hernández-Pérez and Salazar-González (2009) for the one-to-one PDP. To our knowledge, these are the only test instances that can be transformed into $(PD)^3$ test instances by allowing multiple visits to each node.

In our numerical experiments, we consider two classes of test sets proposed by Hernández-Pérez and Salazar-González (2009): class 2 and class 3. The difference between these two classes is that in class 3 each node is either the origin or destination of a single commodity, whereas in class 2 commodities may have the same origin or destination. Because there can only be divided pickups and deliveries in problems of class 2, we only test the $(PD)^3$ on such instances. The instances of both classes are used to test our ILP formulation for the one-to-one PDP.

For completeness, we describe how test instances of class 2 and class 3 are constructed by Hernández-Pérez and Salazar-González (2009). Both are based on a complete graph $G = (\bar{V}, A)$, with a single copy of each arc, where the starting and ending depot is located at the origin $(0, 0)$, and the locations of $|V|$ nodes are independently and uniformly generated from the square $[-500, 500] \times [-500, 500]$. The travel cost c_{ij} between nodes i and j is the rounded Euclidean distance between these two nodes. Next, the origins and destinations of $|K|$ commodities are iteratively determined. For each commodity $k \in K$ in class 2, a random arc a_k is selected, with $o_k := s(a_k)$ and $d_k := t(a_k)$, such that the set of arcs a_k remains cycle free. For class 3 instances, random arcs a_k are selected such that the arcs a_k remain unconnected. Finally, for both instance classes the weights q_k of commodities $k \in K$ are independently and discrete uniformly generated from $\{1, \ldots, 5\}$.

We assume that in all test instances for the one-to-one PDP, we have to visit each node exactly once, also the nodes that are not the origin or destination of any commodity. This is in line with the numerical experiments of Hernández-Pérez and Salazar-González (2009) and Gouveia and Ruthmair (2015). In the corresponding $(PD)^3$ instances, we assume that each node is visited at least once.

4.2. One-to-One Pickup and Delivery Problem

In this section, we consider the one-to-one PDP without divisible pickups and deliveries and present numerical results obtained by solving test instances from class 2 and class 3 using our compact ILP formulation (IP). In Tables 2 and 3, the first four columns correspond to the instance set name, the number $|\bar{V}|$ of nodes, the number $|K|$ of commodities, and the vehicle capacity Q. For each setting of $(|\bar{V}|, |K|, Q)$, we solve the 10 test instances constructed by Hernández-Pérez and Salazar-González (2009). The remaining columns correspond to the number of instances where the infeasibility of the instance was proved (# Inf.), the number of instances where the algorithms did not find an optimal solution within the time limit (Unsolved within 2 h), and the average computation time over all instances that are not proven to be infeasible (Time). The columns with IP correspond to our results obtained by solving (IP), the columns with BE correspond to results obtained by the Benders decomposition approach from Hernández-Pérez and Salazar-González (2009), and the columns C/L/PL correspond to results obtained by different branch-and-cut methods from Gouveia and Ruthmair (2015). The results for BE and C/L/PL are all taken from table 5 in Gouveia and Ruthmair (2015). Moreover, the BE results are obtained using a personal computer with Intel Pentium 3.0 GHz and CPLEX 10.2, whereas the C/L/PL results are obtained using a single core of an Intel Xeon E5540 or E5649, both 2.53 GHz. All reported solution methods have a time limit of two hours.

In Tables 2 and 3 we do not report results for instances with $|K| = 5$, that is, with only five commodities, because all such instances are solved to optimality within 10 seconds on average. Moreover, for some test instances not all existing solution methods from the literature have been applied. In these cases we report "—".

Table 2. Results of Solving (*IP*) on the Test Instances of Class 3

				# Inf.		Unsolved within 2 h					Time				
Set	$\|\bar{V}\|$	$\|K\|$	Q	IP	C/L/PL	IP	BE	C	L	PL	IP	BE	C	L	PL
m10Q5	22	10	5	0	0	0	0	0	0	0	2	2	2	1	93
m10Q10	22	10	10	0	0	0	0	0	0	7	218	87	165	612	5,670
m10Q15	22	10	15	0	0	0	0	0	2	7	209	62	30	1,741	5,122
m10Q20	22	10	20	0	0	0	0	0	0	5	22	2	2	328	4,128
m10Q25	22	10	25	0	0	0	0	0	1	4	14	1	2	1,099	4,086
m10Q30	22	10	30	0	0	0	0	0	1	4	9	1	2	1,294	4,379
m10Q500	22	10	500	0	—	0	0	—	—	—	11	1	—	—	—
m15Q5	32	15	5	0	0	1	2	1	1	8	874	2,006	2,529	1,053	5,922
m15Q10	32	15	10	0	0	9	9	9	9	10	7,119	6,523	6,493	6,908	7,200
m15Q15	32	15	15	0	0	9	5	4	9	10	6,720	4,124	3,284	6,595	7,200
m15Q20	32	15	20	0	0	7	0	0	9	10	5,757	918	269	7,033	7,200
m15Q25	32	15	25	0	0	2	0	0	8	10	3,561	118	40	5,971	7,200
m15Q30	32	15	30	0	0	1	0	0	9	10	2,649	101	43	6,482	7,200
m15Q500	32	15	500	0	—	0	0	—	—	—	1,905	99	—	—	—

For instances of class 3, we observe in Table 2 that we can solve instances with up to 32 nodes and 15 commodities. In fact, we easily solve all instances with $|\bar{V}| = 22$ and $|K| = 10$, and the majority of the instances with $|\bar{V}| = 32$, $|K| = 15$, and larger capacity values Q. For smaller values of Q, such as in m15Q10 and m15Q15, our ILP formulation cannot solve 9 out of 10 instances within the time limit. However, such problems with limited capacity cannot be solved by the other solution methods either. Overall, we conclude that our ILP formulation is almost competitive with existing methods from the literature: BE and C solve a slightly larger number of class 3 instances and typically require less computation time. Nevertheless, the overall performance of (IP) is surprising, because our ILP formulation is a compact formulation using an off-the-shelf solver that can easily be extended to the $(PD)^3$, whereas the other methods are tailor-made for the one-to-one PDP.

Our conclusions are confirmed in Table 3, where we show the results for test instances of class 2. In fact, we are able to solve a test instance from the instance set n20m10Q10 that both Hernández-Pérez and Salazar-González (2009) and Gouveia and Ruthmair (2015) cannot solve. (We have to remark, however, that in an independent and simultaneously conducted study Castro, Cire, and Beck (2020) are able to solve this and the majority of class 2 and class 3 instances for the one-to-one PDP.) Additionally, our ILP formulation (IP) correctly identifies in all cases whether an instance is infeasible. Contrary to instances of class 3, instances of class 2 may be infeasible because of the total capacity necessary to transport different commodities located at the same node. Indeed, when Q is smallest, that is, $Q = 10$, then the test instances are infeasible most often.

4.3. One-to-One Pickup and Delivery Problem with Divisible Pickups and Deliveries

In this section, we present numerical results for the $(PD)^3$, obtained by solving test instances from class 2. We use our compact formulation $(IP_R)^3$ of Section 3.2.3, which has a reduced number of constraints compared with $(IP)^3$. However, because the number of time-consistency constraints (26)–(29) in $(IP_R)^3$ remains large, that is, of order $O(|\bar{V}|^3|K|^2)$, we have implemented them as lazy constraints in CPLEX. Every time an incumbent solution is found, a single violated constraint from (26)–(29) is added within the lazy constraint callback. The rationale for using these lazy constraints is that in the optimal solution most of them are not active. In this way, we try to include only those constraints that we actually need. Finally, we observed that the heuristics within CPLEX's branch-and-bound procedure typically find incumbent solutions that do not satisfy the relaxed time-consistency constraints. Because this forces us to add a lazy constraint that is likely to be irrelevant in the optimal solution, we have set CPLEX's MIP emphasis parameter to three, reducing CPLEX's effort on these heuristics. This significantly improved CPLEX's performance on our test instances.

The results are presented in Table 4. Similarly to the tables of the previous section, the first four columns correspond to the instance set name, the number $|\bar{V}|$ of nodes, the number $|K|$ of commodities, and the vehicle capacity Q. For each setting of $(|\bar{V}|, |K|, Q)$, we solve 10 test instances from class 2. These are the exact same instances as in Table 3, except that we now allow more than one visit to a node. In the next two columns, we report how many instances were unsolved after a time limit of 7,200 seconds (2 h t.l.) and 10,800 seconds (3 h t.l.), respectively. The results in the final columns,

Table 3. Results of Solving (*IP*) on the Test Instances of Class 2

				# Inf.		Unsolved within 2 h					Time				
Set	$\|\bar{V}\|$	$\|K\|$	Q	IP	C/L/PL	IP	BE	C	L	PL	IP	BE	C	L	PL
n10m10Q10	11	10	10	7	7	0	0	0	0	0	0	0	0	0	0
n10m10Q15	11	10	15	1	1	0	0	0	0	0	0	0	0	0	0
n10m10Q20	11	10	20	0	0	0	0	0	0	0	0	0	0	0	0
n10m10Q25	11	10	25	0	0	0	—	0	0	0	0	—	0	0	0
n10m10Q30	11	10	30	0	0	0	—	0	0	0	0	—	0	0	0
n10m10Q500	11	10	500	0	—	0	0	—	—	—	0	0	—	—	—
n10m15Q10	11	15	10	10	10	0	—	0	0	0	—	—	—	—	—
n10m15Q15	11	15	15	9	9	0	0	0	0	0	0	0	0	0	0
n10m15Q20	11	15	20	6	6	0	0	0	0	0	0	0	0	0	0
n10m15Q25	11	15	25	4	4	0	0	0	0	0	0	0	0	0	0
n10m15Q30	11	15	30	2	2	0	0	0	0	0	0	0	0	0	0
n10m15Q500	11	15	500	0	—	0	0	—	—	—	0	0	—	—	—
n15m10Q10	16	10	10	7	7	0	1	0	0	0	7	1,801	1	4	74
n15m10Q15	16	10	15	1	1	0	0	0	0	0	9	0	1	3	36
n15m10Q20	16	10	20	0	0	0	0	0	0	0	6	0	0	6	48
n15m10Q25	16	10	25	0	0	0	0	0	0	0	4	0	0	7	74
n15m10Q30	16	10	30	0	0	0	—	0	0	0	9	0	0	7	67
n15m10Q500	16	10	500	0	—	0	0	—	—	—	8	0	—	—	—
n15m15Q10	16	15	10	10	10	0	—	0	0	0	—	—	—	—	—
n15m15Q15	16	15	15	4	4	0	0	0	0	0	83	2	1	3	21
n15m15Q20	16	15	20	2	2	0	0	0	0	0	157	1	0	0	4
n15m15Q25	16	15	25	0	0	0	0	0	0	0	15	0	0	0	5
n15m15Q30	16	15	30	0	0	0	0	0	0	0	12	0	0	0	6
n15m15Q500	16	15	500	0	—	0	0	—	—	—	11	0	—	—	—
n20m10Q10	21	10	10	2	2	**1**	2	2	2	3	1,540	1,832	1,806	1,854	3,475
n20m10Q15	21	10	15	0	0	1	0	0	0	3	1,068	67	31	374	3,138
n20m10Q20	21	10	20	0	0	1	0	0	0	2	758	53	1	156	1,853
n20m10Q25	21	10	25	0	0	0	0	0	0	2	353	53	1	231	1,770
n20m10Q30	21	10	30	0	0	0	—	0	0	2	346	—	1	212	1,835
n20m10Q500	21	10	500	0	—	0	0	—	—	—	343	53	—	—	—
n20m15Q10	21	15	10	8	8	2	—	2	2	2	7,200	—	7,200	7,200	7,200
n20m15Q15	21	15	15	0	0	8	7	4	5	8	6,530	5,305	3,399	3,684	6,304
n20m15Q20	21	15	20	0	0	6	4	1	3	6	5,417	3,073	910	2,239	4,864
n20m15Q25	21	15	25	0	0	3	0	0	0	4	3,051	172	6	532	3,397
n20m15Q30	21	15	30	0	0	3	0	0	0	2	2,716	114	2	278	2,555
n20m15Q500	21	15	500	0	—	2	0	—	—	—	2,336	117	—	—	—
n25m10Q10	26	10	10	1	1	3	3	2	4	8	3,691	3,684	2,004	3,916	6,545
n25m10Q15	26	10	15	0	0	0	0	0	0	10	1,235	137	67	1,577	7,200
n25m10Q20	26	10	20	0	0	0	0	0	2	9	500	14	5	1,980	6,631
n25m10Q25	26	10	25	0	0	0	0	0	3	8	382	14	5	2,367	6,466
n25m10Q30	26	10	30	0	0	0	—	0	1	8	352	—	4	2,146	6,697
n25m10Q500	26	10	500	0	—	0	0	—	—	—	319	14	—	—	—
n25m15Q10	26	15	10	3	3	7	—	7	7	7	7,200	—	7,200	7,200	7,200
n25m15Q15	26	15	15	0	0	9	8	4	7	10	7,091	5,786	3,167	5,333	7,200
n25m15Q20	26	15	20	0	0	7	5	1	6	8	5,559	3,804	1,385	4,787	6,520
n25m15Q25	26	15	25	0	0	6	1	0	3	9	5,425	1,387	59	3,545	6,864
n25m15Q30	26	15	30	0	0	4	0	0	4	10	5,253	565	14	3,388	7,200
n25m15Q500	26	15	500	0	—	3	0	—	—	—	3,464	372	—	—	—

however, are all obtained using the latter time limit. In the first of those columns, we report the number of instances (# DPD) in which the optimal solution contains divided pickups and deliveries. Next, we present the average computation time of the procedure (Time) in seconds, and the average number of lazy cuts (Cuts) used in CPLEX's branch-and-bound procedure, over all instances that have been solved to optimality. Moreover, we also report the average number of lazy cuts (Cuts t.l.) over *all* instances, thus including those in which CPLEX reached its three-hour time limit. Finally, we report the number of instances (# Ins) that were feasible for the one-to-one PDP *and* solved for both the PDP and $(PD)^3$. These are the instances for which we are able to compare their objective values and, thus, compute the savings obtained from allowing divisible pickups and deliveries. The average savings over those instances are

Table 4. Results of Solving $(IP_R)^3$ on the Test Instances of Class 2

Set	$\lvert\bar{V}\rvert$	$\lvert K\rvert$	Q	2h t.l.	3h t.l.	# DPD	Time	Cuts	Cuts t.l.	# Ins	% Savings
n10m10Q10	11	10	10	0	0	10	76	350	350	3	8.1
n10m10Q15	11	10	15	0	0	9	28	121	121	9	11.9
n10m10Q20	11	10	20	0	0	7	14	75	75	10	9.3
n10m10Q25	11	10	25	0	0	6	12	75	75	10	8.1
n10m10Q30	11	10	30	0	0	6	11	71	71	10	8.1
n10m10Q500	11	10	500	0	0	6	9	66	66	10	8.1
n10m15Q10	11	15	10	5	5	5	1,717	1,474	2,188	0	—
n10m15Q15	11	15	15	4	3	7	2,579	1,479	2,028	1	11.9
n10m15Q20	11	15	20	0	0	9	1,183	1,044	1,044	4	13.6
n10m15Q25	11	15	25	1	0	9	910	611	611	6	15.5
n10m15Q30	11	15	30	0	0	9	260	535	535	8	14.4
n10m15Q500	11	15	500	0	0	9	105	359	359	10	12.8
n15m10Q10	16	10	10	2	1	9	1,851	547	706	3	8.5
n15m10Q15	16	10	15	1	1	7	603	456	729	8	5.6
n15m10Q20	16	10	20	0	0	7	631	368	368	10	6.2
n15m10Q25	16	10	25	0	0	8	186	203	203	10	5.8
n15m10Q30	16	10	30	0	0	7	225	315	315	10	6.1
n15m10Q500	16	10	500	0	0	8	155	279	279	10	6.1
n15m15Q10	16	15	10	10	10	—	—	—	1,601	0	—
n15m15Q15	16	15	15	4	4	6	3,111	763	1,350	2	12.3
n15m15Q20	16	15	20	3	2	8	2,787	852	1,222	6	11.6
n15m15Q25	16	15	25	3	3	6	1,429	523	1,104	7	8.6
n15m15Q30	16	15	30	2	2	8	1,202	638	1,103	8	8.3
n15m15Q500	16	15	500	2	2	8	852	596	1,170	8	8.5
n20m10Q10	21	10	10	5	5	2	1,708	325	725	5	4.7
n20m10Q15	21	10	15	3	2	4	1,895	397	820	8	4.3
n20m10Q20	21	10	20	1	1	1	475	86	290	9	2.2
n20m10Q25	21	10	25	1	1	2	214	73	321	9	2.2
n20m10Q30	21	10	30	1	1	2	287	83	337	9	2.2
n20m10Q500	21	10	500	1	1	2	164	64	320	9	2.2
n20m15Q10	21	15	10	10	10	—	—	—	586	0	—
n20m15Q15	21	15	15	10	10	—	—	—	652	0	—
n20m15Q20	21	15	20	9	9	1	5,899	372	761	1	7.4
n20m15Q25	21	15	25	8	6	4	6,767	686	970	4	7.7
n20m15Q30	21	15	30	6	5	5	3,246	316	843	4	9.2
n20m15Q500	21	15	500	5	3	6	4,256	731	1,177	7	8.5
n25m10Q10	26	10	10	8	8	1	3,367	71	355	1	0.0
n25m10Q15	26	10	15	3	2	5	5,512	436	545	8	4.3
n25m10Q20	26	10	20	2	1	4	3,603	221	302	9	2.8
n25m10Q25	26	10	25	1	1	5	2,475	227	384	9	3.2
n25m10Q30	26	10	30	1	1	5	1,901	268	474	9	3.2
n25m10Q500	26	10	500	1	1	5	1,779	277	470	9	3.2
n25m15Q10	26	15	10	10	10	—	—	—	4	0	—
n25m15Q15	26	15	15	10	10	—	—	—	2	0	—
n25m15Q20	26	15	20	10	10	—	—	—	6	0	—
n25m15Q25	26	15	25	10	10	—	—	—	54	0	—
n25m15Q30	26	15	30	10	10	—	—	—	148	0	—
n25m15Q500	26	15	500	9	8	2	8,194	364	663	2	4.0

reported in the final column (% Savings). Recall that for the $(PD)^3$ instances we assume that every node is visited at least once, also if it does not correspond to the origin or destination of any commodity. Because we assume in the corresponding PDP instances that each node is visited exactly once, the reduction in optimal objective value for the $(PD)^3$ compared with the corresponding PDP instances is exactly the savings obtained by allowing divisible pickups and deliveries.

From Table 4, we observe that for many instances it is beneficial to allow divisible pickups and deliveries. This is not only the case when the capacity Q is small, and thus, dividing pickups and deliveries may be required to obtain a feasible solution, but also when Q is large, that is, when $Q = 500$. For example, for n10m15Q500 we observe that 9 out of 10 instances have an optimal solution with divided pickups and deliveries, whereas all 10 corresponding PDP instances in Table 3 are feasible. On the other hand, for

n10m10Q10 we observe that 7 out of 10 PDP instances are infeasible, whereas all $(PD)^3$ instances can be solved.

The fact that divisible pickups and deliveries may also lead to savings when capacity is not a restriction is caused by the pickup and delivery nature of our problem. In contrast, for example, for the capacitated SDVRP, there are no savings if the capacity is large enough. In this case, all customers can be served in a single tour in which every node is visited exactly once. The fundamental difference with the $(PD)^3$, however, is that commodities in the SDVRP have the same destination, that is, the depot, whereas in the $(PD)^3$ the origins and destinations of commodities are different. This explains why we obtain positive savings for our problem, even if $Q = 500$.

When considering the average number of time-consistency constraints required in CPLEX's branch-and-bound procedure, we observe that we only need a very small percentage of the total number of available constraints. For example, for the instances with $|\bar{V}| = 11$ and $|K| = 10$, there are approximately $4 \cdot 10^5$ constraints, whereas only 60–350 are needed. In general, the required number of constraints does not exceed 3,500, whereas the number of time-consistency constraints in $(IP_R)^3$ is of order $O(|\bar{V}|^3|K|^2)$.

As expected, the average number of time-consistency constraints required for the instances that are solved to optimality (Cuts) is consistently below the average number of time-consistency constraints required for *all* instances (Cuts t.l.). Indeed, the unsolved harder instances require more time-consistency constraints. Additionally, we observe that, for $|\bar{V}| = 11$ and 16, typically more time-consistency constraints are required for instances with small capacity, that is, $Q = 10$, than for instances with large capacity, that is, $Q = 500$. This may be explained by the fact that, in instances with lower capacity, nodes may have to be visited more often than in instances with higher capacity, and thus, more time-consistency constraints may be required. For $|V| = 26$ and $|K| = 15$, this effect seems to be reversed, but the number of time-consistency constraints are so small that we expect that CPLEX does not get the opportunity to add many of the required constraints before the time limit in these instances.

We observe that we are able to solve $(PD)^3$ instances of almost the same size as for the one-to-one PDP. Only for $|\bar{V}| = 26$ and $|K| = 15$ can we solve almost none of the instances. We also observe that our compact formulation has difficulties solving instances with small capacity, in particular, with $Q = 10$. For example, for n15m15Q10 and n20m15Q10 none of the instances are solved. This does not come as a surprise, because our compact ILP formulation *and* existing approaches in the literature also could not solve (slightly larger) one-to-one PDP instances with small capacity (see Table 3). However, for the uncapacitated $(PD)^3$, that is, for $Q = 500$, we *can* solve the majority of the instances, even with $|\bar{V}| = 21$ and $|K| = 15$ or $|\bar{V}| = 26$ and $|K| = 10$.

Finally, we discuss the average savings obtained by allowing divisible pickups and deliveries. As can be observed from Table 4, they are significant and can be as large as 15%. As a general rule, we see that savings are typically larger if the number of nodes $|\bar{V}|$ is smaller or the number of commodities $|K|$ is larger. In these cases, there are more commodities per node, increasing the opportunities for dividing pickups and deliveries. In particular, for $|\bar{V}| = 11$ and $|K| = 15$, the average savings are largest, and for $|\bar{V}| = 21$ or 26 and $|K| = 10$, they are smallest.

5. Discussion

We consider the capacitated single vehicle one-to-one pickup and delivery problem with divisible pickups and deliveries (PDPDPD, or in short $(PD)^3$). In this problem, nodes are allowed to be visited multiple times. We derive a novel compact arc-based ILP formulation for the $(PD)^3$. To the best of our knowledge, it is the first nontrivial ILP formulation for this problem. Moreover, the formulation can be easily applied to the standard one-to-one PDP by restricting the number of times that a node can be visited.

Numerical experiments on one-to-one PDP test instances show that our ILP formulation using an off-the-shelf solver is almost competitive with tailor-made solution methods from the literature for the standard one-to-one PDP. Moreover, for the $(PD)^3$ we solve instances with up to 21 nodes and 15 transportation requests using our formulation. We observe cost savings of up to 15% by allowing divisible pickups and deliveries in one-to-one PDPs.

In our numerical experiments, we have observed that our ILP formulation has difficulties solving problem instances in which the vehicle capacity is relatively small. That is why a direction for future research is to derive tight capacity-based cuts for the $(PD)^3$.

Acknowledgments

The authors are grateful to Gerard Sierksma and Moshe Dror for many beneficial discussions. Moreover, they thank two anonymous referees for their suggestions on previous versions of this paper. Finally, the authors thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high-performance computing cluster.

References

Archetti C, Savelsbergh MWP, Speranza MG (2006) Worst-case analysis for split delivery vehicle routing problems. *Transportation Sci.* 40(2):226–234.

Baldacci R, Bartolini E, Mingozzi A (2011) An exact algorithm for the pickup and delivery problem with time windows. *Oper. Res.* 59(2):414–426.

Berbeglia G, Cordeau J-F, Gribkovskaia I, Laporte G (2007) Static pickup and delivery problems: A classification scheme and survey. *TOP* 15:1–31.

Bruck BP, Iori M (2017) Non-elementary formulations for single vehicle routing problems with pickups and deliveries. *Oper. Res.* 65(6):1597–1614.

Castro MP, Cire AA, Beck JC (2020) An MDD-based Lagrangian approach to the multicommodity pickup-and-delivery TSP. *INFORMS J. Comput.* 32(2):263–278.

Cordeau J-F (2006) A branch-and-cut algorithm for the dial-a-ride problem. *Oper. Res.* 54(3):573–586.

Dror M, Trudeau P (1989) Savings by split delivery routing. *Transportation Sci.* 23(2):141–145.

Dumitrescu I, Ropke S, Cordeau JF, Laporte G (2010) The traveling salesman problem with pickup and delivery: Polyhedral results and a branch-and-cut algorithm. *Math. Programming* 121:269–305.

Gouveia L, Ruthmair M (2015) Load-dependent and precedence-based models for pickup and delivery problems. *Comput. Oper. Res.* 63:56–71.

Haddad MN, Martinelli R, Vidal T, Martins S, Ochi LS, Souza MJF, Hartl R (2018) Large neighborhood-based metaheuristic and branch-and-price for the pickup and delivery problem with split loads. *Eur. J. Oper. Res.* 270(3):1014–1027.

Hernández-Pérez H, Salazar-González JJ (2009) The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *Eur. J. Oper. Res.* 196(3):987–995.

Lai M, Battarra M, Di Francesco M, Zuddas P (2015) An adaptive guidance meta-heuristic for the vehicle routing problem with splits and clustered backhauls. *J. Oper. Res. Soc.* 66:1222–1235.

Letchford AN, Salazar-González JJ (2016) Stronger multi-commodity flow formulations of the (capacitated) sequential ordering problem. *Eur. J. Oper. Res.* 251(1):74–84.

Nagy G, Wassan NA, Speranza MG, Archetti C (2015) The vehicle routing problem with divisible deliveries and pickups. *Transportation Sci.* 49(2):271–294.

Nowak M, Ergun Ö, White CC III (2008) Pickup and delivery with split loads. *Transportation Sci.* 42(1):32–43.

Nowak M, Ergun O, White CC III (2009) An empirical study on the benefit of split loads with the pickup and delivery problem. *Eur. J. Oper. Res.* 198(3):734–740.

Nowak M, Hewitt M, White CC III (2012) Precedence constrained pickup and delivery with split load. *Internat. J. Logist. Res. Appl.* 15(1):1–14.

Öncan T, Aksu DT, Şahin G, Şahin M (2011) A branch and cut algorithm for the multi-vehicle one-to-one pickup and delivery problem with split loads, *IEEE Internat. Conf. Indust. Engrg Engrg Management* (IEEE, Piscataway, NJ), 1864–1868.

Parragh SN, de Sousa JP, Almada-Lobo B (2015) The dial-a-ride problem with split requests and profits. *Transportation Sci.* 49(2): 311–334.

Parragh SN, Doerner K, Hartl RF (2008) A survey on pickup and delivery problems. *J. Betriebswirtschaft* 58:81–117.

Psaraftis H (2011) A multi-commodity, capacitated pickup and delivery problem: The single and two-vehicle cases. *Eur. J. Oper. Res.* 215(3):572–580.

Şahin M, Çavuşlar G, Öncan T, Şahin G, Aksu DT (2013) An efficient heuristic for the multi-vehicle one-to-one pickup and delivery problem with split loads. *Transportation Res. Part C: Emerging Technol.* 27:169–188.

Salazar-González JJ, Santos-Hernández B (2015) The split-demand one-commodity pickup-and-delivery travelling salesman problem. *Transportation Res. Part B: Methodological* 75:58–73.