Computers & Industrial Engineering 179 (2023) 109207

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

The multi-depot pickup and delivery problem with capacitated electric vehicles, transfers, and time windows

Cansu Agrali *, Seokcheon Lee

The School of Industrial Engineering, Purdue University, West Lafayette, IN, 47906, USA

ARTICLE INFO

Dataset link: this link

Keywords:
Transportation
Routing
Pickup and delivery problem with transfers
Electric vehicles
Metaheuristic

ABSTRACT

The Pickup and Delivery Problem has received increasing attention as a result of the recent growth of third-party delivery companies, and electric vehicles (EVs) are becoming a preferable choice for such large delivery systems due to their environmental benefits. The EVs, however, have limited energy capacity; thus, intra-route facilities are required to recharge them. These facilities can also be visited to transfer requests to other vehicles. In this study, we introduce a novel pickup and delivery problem with electric vehicles and transfers. The traditional constraint that each request should be handled by a single vehicle, is relaxed in this problem with transfers, and additionally, we incorporate other practical considerations such as multi-depots, time-windows, and EVs' battery and carrying capacity constraints. We develop a mixed-integer linear programming model encompassing all these constraints. To address the computational difficulty of the problem, we propose a hybrid heuristic combining Simulated Annealing (SA) and Large Neighborhood Search (LNS). Experimental results reveal that for small instances where CPLEX can find optimal solutions, our heuristic finds them about 90% faster.

1. Introduction

The *Pickup and Delivery Problem* (PDP) aims to schedule vehicles by assigning them to requests and determining their routes. Vehicles depart from their depots, serve requests by visiting nodes, and return to their origin nodes. Each request consists of a pickup and a delivery node. Pickup nodes must be visited before their associated delivery nodes. The problem arises in many applications such as third-party delivery companies, e.g., Doordash and Walmart+. Arslan, Agatz, Kroon, and Zuidwijk (2019) address the difficulties faced by logistic service providers in the era of online sales. The authors claim that providing cost-effective delivery systems is one of the fundamental challenges.

E-commerce revenue in the U.S. was $431.6 billion in 2020. The amount is expected to increase to $563.4 billion in the next five years (Statista, Research Department, 2021). One particular concern with this growth is its impact on the environment. Sundstrom and Binding (2012) state that recently electric vehicles (EVs) usage has been increased rapidly because of societal concerns about environmental issues and the availability of renewable power sources. Chen, Kockelman, and Hanna (2016) explain how the use of EVs and shared-vehicles might be an important factor to satisfy the standards of air quality and carbon-emissions. Interested readers can find more information on EVs and their environmental impact in McKinnon, Browne, Whiteing, and Piecyk (2015). Converting to alternative fuel vehicles not only benefits the society in the long run, but also reduces the operational cost by 39%–60% in the short term (Sahin, Yilmaz, Ust, Guneri, & Gulsun, 2009). Amazon has invested in 100,000 EVs, of which 10,000 vehicles are planned to be on the road by 2022 and all 100,000 by 2030 (Meisenzahl, 2021).

Lin, Zhou, and Wolfson (2016) highlight several challenges of EVs, including the fact that they have a shorter service range compared to conventional vehicles. In order to stay operational, EVs must be recharged during their daily tasks. The charging services are provided at facilities, known as *intermediate stops* or *intra-route facilities* in the literature. Schneider, Stenger, and Goeke (2014) explain and provide examples of a few different types of intermediate stops. Additionally, some authors have mainly been interested in the optimal placement of charging stations that eventually provides insightful inputs for our problem network (He, Venkatesh, & Guan, 2012; Li, Huang, & Mason, 2016).

In our research context, we study a novel *Pickup and Delivery Problem*. There are requests to be picked up and delivered, involving a set of EVs and depots. Depots are the origin and final nodes for vehicles, and each vehicle may use a different depot. In the original PDP settings, a single vehicle serves a request, e.g., food and package, by picking it up and delivering it. However, intra-route facilities allow vehicles to exchange requests so that a request can be carried by more than one

* Corresponding author.
E-mail address: cagrali@purdue.edu (C. Agrali).

https://doi.org/10.1016/j.cie.2023.109207
Received 20 February 2022; Received in revised form 30 January 2023; Accepted 23 March 2023
Available online 29 March 2023

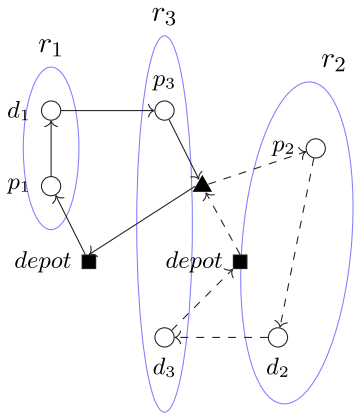

Fig. 1. Example for transferring requests with an intra-route facility.

vehicle (PDP-T). A vehicle picks up a request from its origin node and may drop it at an intra-route facility. Later, another vehicle picks up the request from the facility and delivers the request to its final destination. Note that, pickup and delivery nodes are associated with each other (ellipses in Fig. 1). Therefore, a delivery node cannot be fulfilled by any random commodity that is already picked. For further reading about paired/unpaired nodes, we refer readers to Parragh, Doerner, and Hartl (2008).

The PDP-T aims to meet the demand at each node by devising routing schedules for vehicles and simultaneously satisfying the predetermined constraints, e.g., time-windows and vehicles' capacities and batteries. An example can be seen in Fig. 1. There are three requests r_1, r_2, r_3 (blue ellipses), one intra-route facility (triangle), and two depots (rectangular), with a total of nine nodes. There are two vehicles v_1 and v_2, represented by the solid and the dashed routes, respectively. p_1 and d_1 together define one of the requests (r_1). This paper will refer to them as pairs or associated nodes. Moreover, r_1 and r_2 are handled by a single vehicle - v_1 and v_2, respectively; however, r_3 is transported to the intra-route facility by v_1, and v_2 delivers r_3 to its final destination after picking it up from the facility. Note that, "intra-route facilities" and "transfer nodes" are interchangeable terms in this paper.

1.1. Related literature

The third-party delivery systems can be studied under the PDP, a variant of the Vehicle Routing Problem (VRP). The main difference is that in the PDP, the pickup node must be visited before its associated delivery node. Several VRP surveys including Braekers, Ramaekers, and Van Nieuwenhuyse (2016), Golden, Raghavan, and Wasil (2008), Lin, Choy, Ho, Chung, and Lam (2014), Montoya-Torres, López Franco, Nieto Isaza, Felizzola Jiménez, and Herazo-Padilla (2015) acknowledge the PDP in their state-of-arts.

After the PDP was introduced by Mosheiov (1994) and Savelsbergh and Sol (1995), it has been extensively studied under many names and with many variations: e.g., the VRP with simultaneous pickup and deliveries (Chen & Wu, 2006), the PDP with time-windows (PDP-TW) by Li and Lim (2003) and Ropke and Pisinger (2006), and the Dial-a-Ride (Borndorfer, Grotschel, Klostermeiner, & Kuttner, 1997). For further information about the PDP variants, we refer readers to these studies: Irnich (2000), Dondo, Méndez, and Cerdá (2008), Ben Alaia, Dridi, Bouchriha, and Borne (2013), and Furtado, Munari, and Morabito (2017). To the extent of our knowledge, Berbeglia, Cordeau, Gribkovskaia, and Laporte (2007), Berbeglia, Cordeau, and Laporte (2010), Koç, Laporte, and Tükenmez (2020), Parragh et al. (2008) are the most recent surveys conducted on the PDP. In the rest of this section, we will discuss the recent publications on the PDP with transfers.

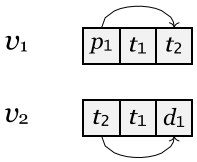

Fig. 2. Transfer case example.

Cortés, Matamala, and Contardo (2010) propose a node-based mixed-integer linear programming (MILP) model for the PDP-T with time windows (PDP-TWT). They treat each transfer node as two separate nodes: "start" and "final". While visiting a transfer node, vehicles must enter the associated start node (passengers get off the vehicle if they need) and proceed to the associated final node to collect passengers before leaving the final node. In addition to their MILP model, they develop a branch-and-cut method with Benders Decomposition. The approach is tested against a generic MILP solver regarding the computational runtime. They suggest that their extra variables (separating a transfer into two nodes) can be promising for novel customer-based objective functions.

Masson, Lehuede, and Peton (2013) provide an Adaptive Large Neighborhood Search (ALNS) algorithm for the PDP-T. They test the algorithm with a real-life case (i.e., people with disabilities) and highlight that transfer nodes can improve the objective value up to 9%. They discuss the computational runtime of the problem and provide insights on why decision makers should tolerate it.

Masson, Lehuédé, and Péton (2014) examine the Dial-a-Ride problem with transfers, which is a variant of the PDP-T. They develop an ALNS metaheuristic to reduce the computational runtime and point out that transfers can help to improve the objective value by 8% on real-life instances.

Rais, Alvelos, and Carvalho (2014) formulate a new generic MILP model for the PDP-T and, then, specify the model by taking into account real-life problems, such as time-windows and separable requests. Using existing benchmark instances, they demonstrate the importance of the transfer operation by comparing their study with the PDP (without transfer). However, we have found that this model has room for improvement: specifically, some cases of transfer scenarios are identified but not represented by the model. The constraint given below is from Rais et al. (2014):

$$\sum_{j:ji\in\mathcal{A}} y_{ji}^{kr} + \sum_{j:ij\in\mathcal{A}} y_{ij}^{lr} \le s_{jr}^{kl} + 1 \qquad \forall i \in \mathcal{T}, \forall r \in \mathcal{R}, \forall k,l \in \mathcal{V} : k \neq l$$

Binary variable y_{ij}^{kr} gets 1 if request r is transported from node i to node j by vehicle k, and another binary variable s_{jr}^{kl} gets 1 if request r is transferred from vehicle k to l at transfer node j. The constraint implies that if request r visits transfer node i with vehicle k, and the same request r leaves the same transfer node with vehicle l, then it means that the request is transferred. The inequality follows a correct logic; however, it overlooks the case of when more than one vehicle visits the same two transfer nodes. An example is given in Fig. 2. Assume that, p_1 is transferred through t_2. In this case, the request enters and leaves t_1 with vehicle v_1 ($y_{p_1t_1}^{v_1r_1} = 1$), and the same request also visits t_1 with vehicle v_2 ($y_{t_1d_1}^{v_2r_1} = 1$). While this given scenario is a possibility, the above constraint would indicate that the example is infeasible due to $s_{t_1r_1}^{v_1v_2} = 0$. Thus, our correction to the constraint will be presented in Section 2.1.

Danloup, Allaoui, and Goncalves (2018) introduce the first genetic algorithm (GA) for the PDP-T. They compare the GA with a large neighborhood search (LNS) metaheuristic. Both algorithms are evaluated on existing benchmark instances. Their results suggest that GA outperforms LNS for the PDP-T.

Peng, Al Chami, Manier, and Manier (2019) consider the selective PDP-T. In the selective PDP, carriers have the flexibility of not serving

specific customers. They propose a multi-objective mathematical model and test it on new benchmark instances. Due to long computational runtimes for large instances, they develop a multi-objective Particle Swarm Optimization (PSO) algorithm. They highlight that their algorithm generates good solutions regarding computational runtimes and the gap between the PSO and the mathematical model.

Sampaio, Savelsbergh, Veelenturf, and Woensel (2020) also develop an ALNS metaheuristic for the PDP-T. They show that transferring requests would significantly impact the total distance when drivers are occupied with too many requests. Note that, they apply the same MILP model from Rais et al. (2014).

Thus far, all the studies reviewed in this section do not consider EVs, that come with battery constraints. Hou et al. (2016) formulate a MILP model and proposes a dispatch algorithm for a ride-sharing problem with electric taxis and transfers. The objective function is to maximize the number of passengers, and they test the algorithm on a case study for Shanghai. In their study, passengers are only allowed to transfer to another vehicle once. Although they mention charging vehicles, the mathematical model does not include any constraints on charging EVs. The transfer constraint in this study is the same as Rais et al. (2014)'s.

1.2. Our contributions

The PDP-TWT with multi-depot and capacitated EVs is considered in this paper. To the best of our knowledge, we are the first to combine PDP-TWT with capacitated EVs and partial charging. Although conventional vehicles are often used by delivery companies nowadays, EVs are expected to be the primary vehicle in the near future (Meisenzahl, 2021). Moreover, many papers do not include partial charging, but Keskin and Çatay (2016) highlight the practical impact of it. Observe that, the problem can be applied in drone deployments due to similar battery constraints and can be investigated in both long-haul and in-city logistics contexts.

We formulate a MILP model for this novel problem and develop a hybrid metaheuristic approach. They are tested with 371 instances, and several computational experiments are conducted to give insights to readers.

In the next section, the problem definition and the mathematical model will be presented. Section 3 includes our heuristic approach with several sub-algorithms. The computational experiments are shown in Section 4. We conclude our paper with a discussion in Section 5.

2. Problem definition

In our problem context, $\mathcal{N}$ and $\mathcal{R}$ indicate the set of all nodes and all requests, respectively. Each request $i \in \mathcal{R}$ is associated with two nodes (pickup and delivery), and each node has a load L_i. Pickup nodes have positive loads, while delivery nodes have negative. $p(r)$ and $d(r)$ respectively represent the pickup and the delivery node of the corresponding request r. A request r must be picked up from its origin node $p(r)$ and delivered to its destination node $d(r)$ during the time period $[E_{p(r)}, T_{d(r)}]$ (Table 1).

We are given a fleet of EVs, $\mathcal{V}$. Each vehicle $k \in \mathcal{V}$ starts its route from its depot $o(k) \in \mathcal{D}$ and ends at the same node $o'(k) \in \mathcal{D}$. The origin and the destination depots are the exact same location but represented by two separate nodes. A heterogeneous fleet of EVs is considered in this research. Therefore, vehicles may have different capacities, batteries, and ranges per an-hour-charging. It is assumed that the vehicles leave their depot fully charged.

The transfer nodes (intra-route facilities) are indicated by $\mathcal{T}$, and the placement of these facilities is not a part of this study. Moreover, we assume that the facilities have unlimited chargers for EVs and capacity for storing requests between drop-off and pick-up times. Note that, EVs can be recharged partially instead of full-charge.

The mathematical model is presented in the next subsection. Table 1 includes all sets and parameters while Table 2 shows all decision variables used in the model.

Table 1
Sets and parameters.

$\mathcal{N}$:	Set of all nodes
$\mathcal{D}$:	Set of depots (includes final depots), $\mathcal{D} \subset \mathcal{N}$
$\mathcal{T}$:	Set of transfer nodes, $\mathcal{T} \subset \mathcal{N}$
$\mathcal{R}$:	Set of requests
$\mathcal{V}$:	Set of vehicles
c_{ij} :	The cost of traveling from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$
t_{ij} :	The distance between nodes $i \in \mathcal{N}$ and $j \in \mathcal{N}$
Q_k :	The capacity of vehicle $k \in \mathcal{V}$
B_k :	The mileage that vehicle $k \in \mathcal{V}$ can travel with a full battery
H_k :	The mileage that vehicle $k \in \mathcal{V}$ can travel with 1-minute charge
L_i :	The load of node $i \in \mathcal{N}$
E_i :	The earliest time that a vehicle can arrive to node $i \in \mathcal{N}$
T_i :	The latest time that a vehicle can arrive to node $i \in \mathcal{N}$
M :	A large number

Table 2
Decision variables.

x_{ij}^k :	binary variable, gets 1 if vehicle $k \in \mathcal{V}$ directly travels from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$
y_{ij}^{kr} :	binary variable, gets 1 if request $r \in \mathcal{R}$ is carried by vehicle $k \in \mathcal{V}$ between nodes $i \in \mathcal{N}$ and $j \in \mathcal{N}$
z_{ir}^{kl} :	binary variable, gets 1 if request $r \in \mathcal{R}$ is transferred between vehicles $k \in \mathcal{V}$ and $l \in \mathcal{V}$ at transfer node $i \in \mathcal{T}$
p_i^k :	vehicle $k \in \mathcal{V}$'s charging duration at transfer node $i \in \mathcal{T}$
a_i^k :	arrival time of vehicle $k \in \mathcal{V}$ in node $i \in \mathcal{N}$
d_i^k :	departure time of vehicle $k \in \mathcal{V}$ from node $i \in \mathcal{N}$
b_i^k :	remaining battery (mileage) of vehicle $k \in \mathcal{V}$ when it arrives to node $i \in \mathcal{N}$

2.1. Mathematical model

$$\text{minimize} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}:i\neq j} \sum_{k\in\mathcal{V}} c_{ij} x_{ij}^k \tag{1}$$

subject to

$$\sum_{j\in\mathcal{N}} x_{ij}^k = 1 \qquad \forall k \in \mathcal{V}, i = o(k) \tag{2}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^k = \sum_{j\in\mathcal{N}} x_{jl}^k \qquad \forall k \in \mathcal{V}, i = o(k), l = o'(k) \tag{3}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^k = \sum_{j\in\mathcal{N}} x_{ji}^k \qquad \forall i \in \mathcal{N} \setminus \{o(k), o'(k)\}, k \in \mathcal{V} \tag{4}$$

$$\sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} y_{ij}^{kr} = 1 \qquad \forall r \in \mathcal{R}, i = p(r) \tag{5}$$

$$\sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} y_{ji}^{kr} = 1 \qquad \forall r \in \mathcal{R}, i = d(r) \tag{6}$$

$$\sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} y_{ij}^{kr} = \sum_{j\in\mathcal{N}} \sum_{k\in\mathcal{V}} y_{ji}^{kr} \qquad \forall i \in \mathcal{T}, r \in \mathcal{R} \tag{7}$$

$$\sum_{j\in\mathcal{N}} y_{ij}^{kr} = \sum_{j\in\mathcal{N}} y_{ji}^{kr} \qquad \forall i \in \mathcal{N} \setminus \{\mathcal{T}, p(r), d(r)\}, k \in \mathcal{V}, r \in \mathcal{R} \tag{8}$$

$$y_{ij}^{kr} \leq x_{ij}^k \qquad \forall i, j \in \mathcal{N}, k \in \mathcal{V}, r \in \mathcal{R} : i \neq j \tag{9}$$

$$d_i^k + t_{ij} - a_j^k \leq M(1 - x_{ij}^k) \qquad \forall i, j \in \mathcal{N}, k \in \mathcal{V} : i \neq j \tag{10}$$

$$a_i^k \leq d_i^k \qquad \forall i \in \mathcal{N} \setminus \{\mathcal{T}\}, k \in \mathcal{V} \tag{11}$$

$$a_i^k + p_i^k \leq d_i^k \qquad i \in \mathcal{T}, k \in \mathcal{V} \tag{12}$$

$$d_i^k \geq E_i \qquad \forall k \in \mathcal{V}, r \in \mathcal{R}, i = p(r) \tag{13}$$

$$a_i^k \leq T_i \qquad \forall k \in \mathcal{V}, r \in \mathcal{R}, i = d(r) \tag{14}$$

$$\sum_{j\in\mathcal{N}} (y_{ji}^{kr} - y_{ij}^{kr}) + \sum_{j\in\mathcal{N}} (y_{ij}^{lr} - y_{ji}^{lr}) \leq 1 + z_{ir}^{kl} \qquad \forall i \in \mathcal{T}, k, l \in \mathcal{V}, r \in \mathcal{R} : k \neq l \tag{15}$$

$$a_i^k - d_i^l \leq M(1 - z_{ir}^{kl}) \qquad \forall i \in \mathcal{T}, k, l \in \mathcal{V}, r \in \mathcal{R} : k \neq l \tag{16}$$

$$\sum_{r\in\mathcal{R}} L_i y_{ij}^{kr} \le Q_k x_{ij}^k \qquad \forall i,j\in\mathcal{N}, k\in\mathcal{V}: i\neq j \quad (17)$$

$$b_j^k \le b_i^k - t_{ij} + M(1-x_{ij}^k) \qquad \forall i\in\mathcal{N}\setminus\{\mathcal{T}\}, j\in\mathcal{N}\setminus\{o(k)\}, k\in\mathcal{V} \quad (18)$$

$$b_j^k \le b_i^k - t_{ij} + M(1-x_{ij}^k) + H_k\, p_i^k \qquad \forall i\in\mathcal{T}, j\in\mathcal{N}\setminus\{o(k)\}, k\in\mathcal{V} \quad (19)$$

$$b_i^k = B_k \qquad \forall k\in\mathcal{V}, i=o(k) \quad (20)$$

$$b_i^k + H_k\, p_i^k \le B_k \qquad i\in\mathcal{T}, k\in\mathcal{V} \quad (21)$$

$$p_i^k \le (B_k/H_k)\sum_{j\in\mathcal{N}} x_{ji}^k \qquad i\in\mathcal{T}, k\in\mathcal{V} \quad (22)$$

$$x_{ij}^k, y_{ij}^{kr} \in \{0,1\} \qquad \forall i,j\in\mathcal{N}, k\in\mathcal{V}, r\in\mathcal{R}: i\neq j \quad (23)$$

$$z_{ir}^{kl} \in \{0,1\} \qquad \forall i\in\mathcal{T}, k,l\in\mathcal{V}, r\in\mathcal{R}: k\neq l \quad (24)$$

$$a_i^k, d_i^k, b_i^k, p_i^k \ge 0 \qquad \forall i\in\mathcal{N}, t\in\mathcal{T}, k\in\mathcal{V} \quad (25)$$

The objective function minimizes the total cost (1) that occurs as a result of the traveled distance. Constraints (2) ensure that all vehicles leave their depot, and each vehicle has one and only one route. All vehicles must return to the depot in (3). It is possible that a vehicle can directly travel from the origin depot to the destination depot, which would incur zero cost to the system. If a vehicle enters a node (except for the vehicle's origin and destination depots), then the vehicle has to leave the node in constraints (4). In constraint sets (5) and (6), all requests are picked up and delivered, respectively. If a request visits a transfer node, it must leave the transfer node either with the same vehicle or with another vehicle in (7). If a request enters a node that is not its pickup, its delivery, or a transfer node, then the request must leave that node with the same vehicle in (8). Constraints (9) connect two variables, and if a vehicle does not directly travel from node i to j, it cannot carry a request through arc (i, j). Constraints (10) explain the relationship between departure and arrival times: the arrival time to the successor node (j) must be greater than or equal to the summation of the departure time from the predecessor node (i) and the travel time between nodes i and j. Constraints (11) ensure that for all vehicles and at all nodes, the departure time of a vehicle cannot be earlier than its arrival time. The departure time from a transfer node is calculated considering recharging duration in (12). The time-window constraints for pickup and delivery nodes are captured in constraints (13) and (14). Constraints (15) are the modified constraints (see Section 1.1), and they ensure that the corresponding z variable will be one if and only if the request enters the transfer node with vehicle k and leaves the node with vehicle l — at the same time it does not leave the node with vehicle k and does not enter with vehicle l. Referring to Fig. 2, when the constraint is written for t_1, v_1, v_2, and r_1, the decision variables get values as follows: $y_{p_1t_1}^{v_1r_1} = 1$, $y_{t_1t_2}^{v_1r_1} = 1$, $y_{t_2t_1}^{v_2r_1} = 1$, and $y_{t_1d_1}^{v_2r_1} = 1$. Thus, $z_{t_1r_1}^{v_1v_2}$ may get zero, unlike Rais et al. (2014)'s constraint. If a request is transferred between vehicles at a transfer node, then the arrival time of the pickup vehicle to the node must be earlier than the departure time of the delivery vehicle in (16). Constraints (17) assure that vehicle capacities are never exceeded. Constraints (18) and (19) calculate battery levels (remaining mileage) after leaving a regular and a transfer node, respectively. All vehicles leave their depots fully charged in (20). Vehicles cannot be charged more than their battery capacities in (21). Vehicles must visit a transfer node to be recharged in (22). We assume that recharging EVs is a linear function of time (Jun, Lee, & Yih, 2021). Constraints (23) and (24) require the given variables to be binary. Constraints (25) declare positive continuous variables.

2.2. Valid inequalities

In order to improve the runtime of the MILP model, we introduce a set of problem-specific valid inequalities. Constraints (26) eliminate the transits from a delivery node to its associated pickup node for all request and all vehicles — precedence constraints. Remember that, a request must be picked up first and then delivered.

$$x_{ij}^k \le 0 \qquad \forall r\in\mathcal{R}, k\in\mathcal{V}, i=d(r), j=p(r) \quad (26)$$

If there are arcs whose distance is longer than a vehicle can travel with a full battery ($t_{ij} > B_k : i,j\in\mathcal{N}, k\in\mathcal{V}: i\neq j$), the vehicle cannot directly transit between those nodes without recharging. The parameter γ_{ij}^k gets zero if the above situation exists. Thereby, some variables would be initially set to zero by Constraints (27).

$$x_{ij}^k \le \gamma_{ij}^k \qquad \forall i,j\in\mathcal{N}, k\in\mathcal{V}: i\neq j \quad (27)$$

Constraints (28) and (29) ensure that no vehicle enters an origin depot and leaves a destination depot. Here, D^+ indicates all origin depots while D^- shows all destination depots. Recall that, a vehicle starts its route from an origin depot and ends at a destination depot.

$$x_{ij}^k \le 0 \qquad \forall i\in\mathcal{N}, j\in D^+, k\in\mathcal{V}: i\neq j \quad (28)$$

$$x_{ij}^k \le 0 \qquad \forall i\in D^-, j\in\mathcal{N}, k\in\mathcal{V}: i\neq j \quad (29)$$

Intuitively, due to the objective function, a vehicle will not pass through arc (j, i) if it travels from node i to node j — Constraints (30).

$$x_{ij}^k + x_{ji}^k \le 1 \qquad \forall i,j\in\mathcal{N}, k\in\mathcal{V}: i\neq j \quad (30)$$

Order matching constraint (a valid inequality) for the PDP is proposed by Ruland and Rodin (1997). The constraint is stated in Eq. (31).

$$x_{ij}^k + x_{il}^k + x_{lm}^k \le 2 \qquad \forall k\in\mathcal{V}, r^1, r^2\in\mathcal{R}: i=p(r^1), j=d(r^1), l=p(r^2), m=d(r^2) \quad (31)$$

Although the runtimes of the mathematical model are improved with the valid inequalities, for large instances (more than seven requests), it is not possible to obtain a feasible solution in 24 h — the details are discussed in Section 4. In order to tackle this, we devise a metaheuristic approach, that is explained in the next section.

3. Heuristic approach

The PDP-TW is classified as *NP*-Hard (Lau & Liang, 2002) and is a peculiar case of our problem, where no requests are transferred. Thereby, we can conclude that the PDP-TWT is also an *NP*-Hard problem. The time to solve these problems exactly increases exponentially with the number of variables, and so an efficient heuristic method is needed to provide a good solution.

Metaheuristic algorithms are used for their efficacy in finding good solutions within a reasonable time (Gandomi, Yang, Talatahari, & Alavi, 2013). They start from an initial solution or a solution pool and aim to improve the given solution(s) throughout the search process. In this study, we use a hybrid metaheuristic, which benefits from particular advantages of sub-heuristics. In order to find an initial solution, a construction heuristic is devised. The idea behind the construction heuristic is assigning requests to either a single vehicle or two vehicles by using transfer nodes. These are commonly used methods in the literature to generate initial solutions for the PDP-T, e.g., Danloup et al. (2018), Sampaio et al. (2020). It is important to note that obtaining a feasible solution in our problem context is time-consuming due to the strict constraints.

The main flow of the proposed heuristic approach can be seen in Fig. 3. It starts with the construction heuristic that contains two sub-algorithms. Then, the hybrid heuristic improves the given solution by utilizing eight neighborhoods, two of which are explained in separate algorithms (Algorithms 5 and 6).

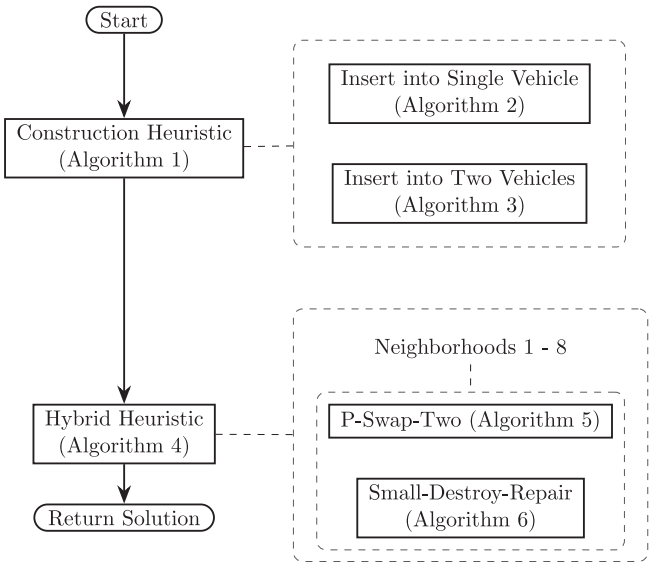

Fig. 3. Flow diagram of algorithms.

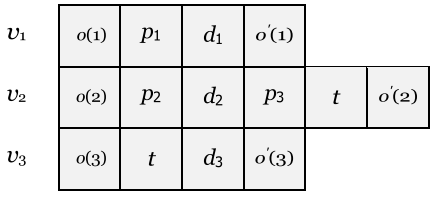

Fig. 4. Solution representation.

3.1. Solution representation

In this heuristic approach, a solution is represented by an array that contains $|\mathcal{V}|$ sub arrays. Each sub array defines the corresponding vehicle's route. A 3-vehicle-3-request example can be seen in Fig. 4. Vehicle 1 (v_1) leaves its origin depot $o(1)$, handles request 1 by picking it up and delivering it, and returns to its destination depot $o'(1)$. v_2 and v_3 work together to deliver request 3: v_2 picks it up and drops off at transfer facility t, and then v_3 picks the request from the facility and delivers it to the final destination. Note that, throughout this paper, the words "vehicle" and "route" may be used interchangeably as vehicles are defined by their routes.

3.2. Constructing an initial solution

In order to construct an initial solution, a heuristic approach that does not guarantee a feasible solution is proposed. Preliminary tests suggest that searching for a feasible solution in the construction step aggravates computational runtimes and does not improve the average final solution quality.

In the construction phase, the request set ($\mathcal{R}$) is divided into two subsets based on the distance between associated pickup and delivery nodes of each request: singles and doubles. These sets describe how requests will be assigned to vehicles: either by Algorithm 2 or Algorithm 3. If the distance between associated pickup and delivery nodes is less than a predetermined value, then the request is assigned to singles. Otherwise, it is allocated to doubles. Fig. 5 demonstrates an instance of how single and double vehicle assignments work. Assume that, request 3 (p_3 and d_3) is the request to be assigned. p_3 and d_3 are on the same route in the single insertion, while they are separated into two vehicles by using transfer t in the double insertion.

Algorithm 1 Construction Heuristic

```
singles ← {}
doubles ← {}
S ← initial solution where routes include only depots
maxDist ← distance between the farthest nodes
p(r) ← pickup node of request r
d(r) ← delivery node of request r
d^r_{p(r)d(r)} ← distance between p(r) and d(r)
for all r ∈ R do
    if d^r_{p(r)d(r)} < maxDist/2 then
        singles ← singles ∪ {r}
    else
        doubles ← doubles ∪ {r}
for all r ∈ singles do
    S ← Algorithm 2(S, r)
for all r ∈ doubles do
    S ← Algorithm 3(S, r)
return S
```

Algorithm 2 Insert into Single Vehicle

```
Require: Request r, Solution S
obj* : best objective found so far
obj̃ : objective of current solution
S* : best solution found so far
S̃ : current solution
obj* ← +∞
R_v ← route of vehicle v
for all v ∈ V do
    for 0 ≤ i ≤ |R_v| do
        for i + 1 ≤ j ≤ |R_v| do
            Add p(r) in R_v at index i
            Add d(r) in R_v at index j
            S̃ ← S
            obj̃ ← calculate_objective(S̃)
            if obj̃ < obj* then
                obj* ← obj̃
                S* ← S̃
            Remove p(r) and d(r) from R_v
            j++
        i++
return S*
```

The requests in the singles set are assigned to vehicles by Algorithm 2 first. Then, Algorithm 3 places the requests in the doubles. Algorithm 3 is executed later; thereby, the routes are not updated by singles' assignments after transfer insertions. Algorithm 2 starts with a partial solution and a request, and then inserts the associated pickup and delivery nodes of the request into the same vehicle with the lowest cost. The requests inserted by Algorithm 2 are not transferred (Fig. 5(a)). Similar to the single insertion algorithm, Algorithm 3 inserts the requests with the lowest cost by considering transfers and precedence constraints.

3.3. Feasibility check

In the literature, a solution's feasibility is controlled by either a MILP model or an algorithm. However, MILP models are computationally expensive to run at each iteration. Thus, in the improvement phase, we use an algorithm with $O(|\mathcal{N}|^2|\mathcal{V}|^2)$ complexity. The algorithm checks time-windows, precedence constraints, and vehicles' battery and capacity. Precedence constraints ensure that for all requests, the pickup node is visited before its associated delivery node, or if a request is transferred, the transfer node is visited after the pickup node and before the delivery node. While checking time-windows constraints, transfer times and precedence relationships are considered in the calculation.

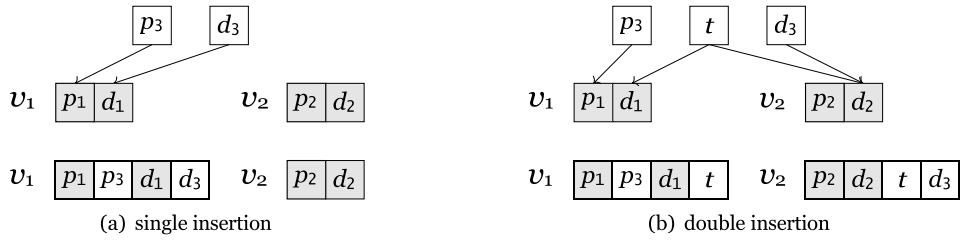

Fig. 5. Examples of single and double insertions.

Algorithm 3 Insert into Two Vehicles

Require: Request r, Solution S
obj^* : best objective found so far
$o\tilde{b}j$: objective of current solution
S^* : best solution found so far
$obj^* \leftarrow +\infty$
$R_v \leftarrow$ route of vehicle v
for all $t \in \mathcal{T}$ **do**
  for all $v_1 \in \mathcal{V}$ **do**
    for all $v_2 \in \mathcal{V}$ **do**
      if $v_1 \neq v_2$ **then**
        Remove t from R_{v_1} and R_{v_2} if exists
        for $0 \leq i \leq |R_{v_1}|$ **do**
          for $i+1 \leq j \leq |R_{v_1}|$ **do**
            for $0 \leq i_2 \leq |R_{v_2}|$ **do**
              for $i_2+1 \leq j_2 \leq |R_{v_2}|$ **do**
                Add $p(r)$ in R_{v_1} at index i
                Add t in R_{v_1} at index j
                Add t in R_{v_2} at index i_2
                Add $d(r)$ in R_{v_2} at index j_2
                $o\tilde{b}j \leftarrow$ calculate_objective(S)
                if $o\tilde{b}j < obj^*$ **then**
                  $obj^* \leftarrow o\tilde{b}j$
                  $S^* \leftarrow S$
                Remove $p(r), t, d(r)$ from R_{v_1} and R_{v_2} accordingly
return S^*

Algorithm 4 Hybrid Heuristic

Require: Solution S, temperature t, cooling schedule $c \in (0,1)$, maximum iteration number MI
$i \leftarrow 0$
$\hat{t} \leftarrow t$
$\rho \leftarrow 1/8$ (8 refers to number of neighborhood definitions, it is 7 when there is 1 transfer node in instance)
S^n: neighbor solution
S^c: current solution
S^*: best solution
$S^c \leftarrow S$
$obj^*, obj^c, obj^n \leftarrow +\infty$
while `true` **do**
  $\sigma \leftarrow$ random number ($\sigma \in [0,1]$)
  if $\sigma \leq \rho$ **then**
    $S^n \leftarrow$ P-Swap-Single(S^c)
  else if $\sigma \leq 2\rho$ **then**
    $S^n \leftarrow$ D-Swap-Single(S^c)
  else if $\sigma \leq 3\rho$ **then**
    $S^n \leftarrow$ MergeDP(S^c)
  else if $\sigma \leq 4\rho$ **then**
    $S^n \leftarrow$ P-Swap-Two(S^c)
  else if $\sigma \leq 5\rho$ **then**
    $S^n \leftarrow$ D-Swap-Two(S^c)
  else if $\sigma \leq 6\rho$ **then**
    $S^n \leftarrow$ Small-Destroy-Repair(S^c)
  else if $\sigma \leq 7\rho$ **then**
    $S^n \leftarrow$ T-Insert(S^c)
  else
    $S^n \leftarrow$ T-Update(S^c)
  if S^n is feasible **then**
    $obj^n \leftarrow$ calculate_objective(S^n)
    if $obj^n < obj^c$ **then**
      $S^c \leftarrow S^n$
      $obj^c \leftarrow obj^n$
      if $obj^n < obj^*$ **then**
        $S^* \leftarrow S^n$
        $obj^* \leftarrow obj^n$
    else
      $\alpha \leftarrow$ a random number: $\alpha \in [0,1]$
      if $\alpha \leq e^{-\left(\frac{obj^n - obj^c}{t}\right)}$ **then**
        $S^c \leftarrow S^n$
        $obj^c \leftarrow obj^n$
      i++
  else
    if $i = 0$ **then**
      $S^c \leftarrow S^n$
    else
      i++
  $t \leftarrow t \times c$
  if $t < \hat{t}$ **then**
    t $\leftarrow \hat{t}$
  if $i = MI$ **then**
    return S^*

3.4. Improvement phase

A hybrid metaheuristic is used to improve the initial solution. The word *hybrid* is used, as the metaheuristic combines some elements of *Simulated Annealing* (SA) and *Large Neighborhood Search* (LNS) algorithms. The SA main structure is applied in the metaheuristic.

The hybrid metaheuristic takes four arguments: a solution from the construction phase (S), an initial temperature (t), a cooling schedule (c), and a maximum non-improved iteration number (MI). Until MI is reached, a new solution is generated by one of the neighborhoods in each iteration. As a given initial solution's feasibility is uncertain, until the algorithm finds the first feasible solution, it accepts infeasible solutions, does not count non-improved iterations and does not update the current objective value. In case a better feasible solution is found, the new solution is accepted, and the best and the current solutions are updated. If the new solution is not better than the previous solution, but feasible, it is accepted with a probability, and the algorithm increases the non-improved iteration counter by one. The temperature is adjusted with the cooling schedule. If the temperature drops to a certain degree, it is raised back to the initial temperature. The steps can be seen in Algorithm 4. Note that, at every iteration, the generated solution is cleaned by a primitive sub-algorithm that removes unused transfer nodes from the routes.

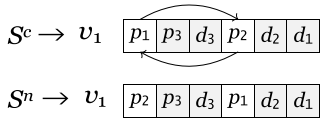

Fig. 6. Swapping two pickup nodes in a vehicle.

3.4.1. Neighborhoods

Within the hybrid heuristic, eight different neighborhood definitions are implemented. The neighbor selection is performed by the roulette wheel method, where all neighborhoods have an equal probability. Turkeš et al. (2020) share a metadata analysis on the adaptive layer of ALNS and argues its effectiveness with regard to the solution quality. Thus, we do not implement adaptive probabilities. Neighborhood definitions are as follows:

P-Swap-Single. Two pickup nodes on the same route are swapped. An example can be seen in Fig. 6, where p_1 and p_2 are swapped. The neighborhood first identifies routes with more than one pickup node. Then, it randomly chooses one of the vehicles and two pickup nodes within the vehicle's route.

D-Swap-Single. Two delivery nodes on the same route are swapped, and it has the same rules as P-Swap-Single, but for delivery nodes.

MergeDP. It chooses a delivery node which is not on the same route with its associated pickup node. Then, the delivery node is inserted into the same route as its associated pickup node. The neighborhood

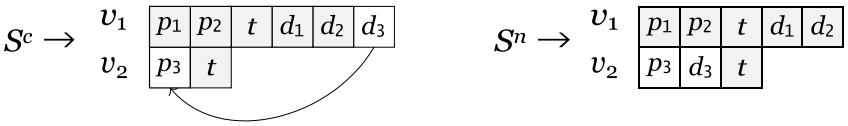

Fig. 7. Inserting the delivery node into the same route.

Fig. 8. Exchanging two pickup nodes' vehicles.

first finds all of the transferred requests and randomly selects one of them. The pickup node's vehicle stays the same, and the delivery node is inserted at the lowest cost considering that the delivery node must be visited later than its associated pickup node. An example can be seen in Fig. 7, where d_3 is removed from v_1's route and placed after p_3 on v_2's route.

P-Swap-Two. Two pickup nodes exchange their routes. However, transfer decisions and their associated delivery nodes are elaborated within the neighborhood while exchanging routes. An example is illustrated in Fig. 8. Assume that, currently p_2 and d_2 are on the same route (v_1), while request 3 (p_3 and d_3) is transferred. After the exchange, p_3 will be on the same route with its associated delivery node so that p_3 is inserted with the lowest cost before d_3. However, p_2 will be separated from d_2. Therefore, p_2 is inserted before the transfer node. Note that, the neighborhood does not force them to be inserted with each other's locations, i.e., p_3 to p_2's location in v_1.

Initially, the neighborhood creates a list of pickup nodes for all vehicles and randomly chooses one pickup node from two different routes (p_1 and p_2). Assume that, their associated delivery nodes are labeled as d_1 and d_2, respectively. The neighborhood checks if both pickup nodes are in the same vehicle as their associated node before exchanging. The pickup nodes are removed from their current routes while storing transfer nodes' indexes on both routes. Transfer nodes might be necessary later in the algorithm.

If p_1 and d_1 are on the same route before the exchange, then the new route of p_1 must contain a transfer node. If the new route already has one, p_1 is inserted before the transfer node with the lowest cost. Otherwise, the neighborhood checks if d_2 is on the same (new) route with p_1. If this is the case, a transfer node, from p_1's original route or another from $\mathcal{T}$, is used. p_1 is placed before d_2, and the selected transfer node is inserted just after p_1. If d_2 is not on the same route, then p_1 is placed with the lowest cost, and a transfer node is inserted just after it.

If p_1 and d_1 currently are not on the same route, a transfer node may be needed. Thereby, the neighborhood checks if they will be on the same route after the exchange. p_1 is inserted with the lowest cost before d_1 if they will be. Otherwise, the same procedure from the previous paragraph applies.

The neighborhood repeats the same process for p_2. The details are given in Algorithm 5.

D-Swap-Two. Two delivery nodes exchange their routes. A similar procedure from P-Swap-Two applies for delivery nodes. However, Algorithm 5 will be slightly different for this neighborhood as delivery nodes must be visited later than their associated pickup nodes or the transfer node used for exchange.

Small-Destroy-Repair. It is from Large Neighborhood Search (LNS). It randomly selects one request and removes the associated pickup and delivery node from their current routes. Note that, the pickup and delivery nodes can be on the same route as well as on different routes. Thereafter, inserting criteria are decided that could be either single

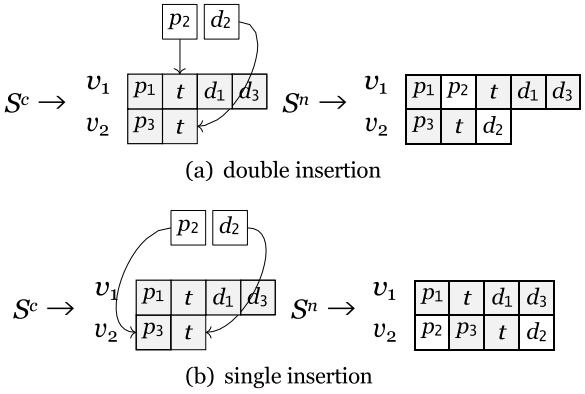

Fig. 9. Remove and insert a request.

insertion or double insertion by transferring. If the request is currently transferred (not transferred), it will be transferred (not transferred) again with a probability after the insertion. The neighborhood's details can be seen in Algorithm 6. Fig. 9 demonstrates an example, where request 2 (p_2 and d_2) is to be inserted — it could be either transferred (above) or handled by a single vehicle (below).

T-Insert. A transfer node is inserted into two routes. The reason for applying T-insert is to be able to generate two routes that can be utilized later by exchanging requests among these vehicles. Vehicles are randomly selected among the ones whose route does not contain a transfer node. However, the transfer node selection and the insertion are performed by aiming for the lowest cost.

T-Update. This neighborhood could only be selected if the problem instance contains more than one transfer node. First, the neighborhood finds the routes with transfer nodes. Then, one transfer node is randomly chosen from the set $\mathcal{T}$, and the transfer nodes within the selected routes are replaced by the new transfer node. This procedure is applied to ensure that each route has the same transfer node; thereby, synchronization between vehicles would be easier, and it lets the algorithm search a different region of the solution space.

4. Computational experiments

In this section, the proposed heuristic approach is tested against the MILP formulation in terms of objective values and runtimes. The MILP model is utilized to evaluate the value of including transfer decisions and the valid inequalities' impact on the runtimes. The hybrid heuristic is compared with a single neighborhood definition and also analyzed with a relaxed version of tight constraints.

Algorithm 5 P-Swap-Two

Require: Solution S
$P_v \leftarrow$ pickup nodes in vehicle v
same1, same2, transfer1, transfer2 $\leftarrow$ `false`
$R_v \leftarrow$ route of vehicle v in solution S
$d_n \leftarrow$ associated delivery node of pickup node n
$i(x) \leftarrow$ index of node x
if $\sum_{v \in \mathcal{V}:|P_v|>0} v > 1$ **then**
  Randomly select two vehicles (v_1 and v_2)
  Randomly select one pickup node from each route (p_1 and p_2)
  if $\{p_1, d_1\} \in R_{v_1}$ **then**
    same1 $\leftarrow$ `true`
  if $\{p_2, d_2\} \in R_{v_2}$ **then**
    same2 $\leftarrow$ `true`
  $R_{v_1} \leftarrow R_{v_1} \setminus p_1$
  $R_{v_2} \leftarrow R_{v_2} \setminus p_2$
  if R_{v_1} contains $t \in \mathcal{T}$ **then**
    transfer1 $\leftarrow$ `true`
    $t_1 \leftarrow t$
  if R_{v_2} contains $t \in \mathcal{T}$ **then**
    transfer2 $\leftarrow$ `true`
    $t_2 \leftarrow t$
  if same1 **then**
    if transfer2 **then**
      $R_{v_2} \leftarrow R_{v_2} \cup p_1 : i(p_1) < i(t_2)$ with the lowest cost
    else
      if transfer1 **then**
        $t_2 \leftarrow t_1$
      else
        $t_2 \leftarrow \mathcal{T}[1]$
      if $d_2 \in R_{v_2}$ **then**
        $R_{v_2} \leftarrow R_{v_2} \cup p_1 : i(p_1) < i(d_2)$ with the lowest cost
        $R_{v_2} \leftarrow R_{v_2} \cup t_2 : i(t_2) = i(p_1) + 1$
      else
        $R_{v_2} \leftarrow R_{v_2} \cup p_1$ with the lowest cost
        $R_{v_2} \leftarrow R_{v_2} \cup t_2 : i(t_2) = i(p_1) + 1$
  else
    if $\{d_1\} \in R_{v_2}$ **then**
      $R_{v_2} \leftarrow R_{v_2} \cup p_1 : i(p_1) < i(d_1)$ with the lowest cost
    else
      if transfer2 **then**
        $R_{v_2} \leftarrow R_{v_2} \cup p_1 : i(p_1) < i(t_2)$ with the lowest cost
      else
        if transfer1 **then**
          $t_2 \leftarrow t_1$
        else
          $t_2 \leftarrow \mathcal{T}[1]$
        if $d_2 \in R_{v_2}$ **then**
          $R_{v_2} \leftarrow R_{v_2} \cup p_1 : i(p_1) < i(d_2)$ with the lowest cost
          $R_{v_2} \leftarrow R_{v_2} \cup t_2 : i(t_2) = i(p_1) + 1$
        else
          $R_{v_2} \leftarrow R_{v_2} \cup p_1$ with the lowest cost
          $R_{v_2} \leftarrow R_{v_2} \cup t_2 : i(t_2) = i(p_1) + 1$
  if same2 **then**
    if transfer1 **then**
      $R_{v_1} \leftarrow R_{v_1} \cup p_2 : i(p_1) < i(t_1)$ with the lowest cost
    else
      if transfer2 **then**
        $t_1 \leftarrow t_2$
      else
        $t_1 \leftarrow \mathcal{T}[1]$
      if $d_1 \in R_{v_1}$ **then**
        $R_{v_1} \leftarrow R_{v_1} \cup p_1 : i(p_2) < i(d_1)$ with the lowest cost
        $R_{v_1} \leftarrow R_{v_1} \cup t_1 : i(t_1) = i(p_2) + 1$
      else
        $R_{v_1} \leftarrow R_{v_1} \cup p_2$ with the lowest cost
        $R_{v_1} \leftarrow R_{v_1} \cup t_1 : i(t_1) = i(p_2) + 1$
  else
    if $\{d_2\} \in R_{v_1}$ **then**
      $R_{v_1} \leftarrow R_{v_1} \cup p_2 : i(p_2) < i(d_2)$ with the lowest cost
    else
      if transfer1 **then**
        $R_{v_1} \leftarrow R_{v_1} \cup p_2 : i(p_2) < i(t_1)$ with the lowest cost
      else
        if transfer2 **then**
          $t_1 \leftarrow t_2$
        else
          $t_1 \leftarrow \mathcal{T}[1]$
        if $d_1 \in R_{v_1}$ **then**
          $R_{v_1} \leftarrow R_{v_1} \cup p_2 : i(p_2) < i(d_1)$ with the lowest cost
          $R_{v_1} \leftarrow R_{v_1} \cup t_1 : i(t_1) = i(p_2) + 1$
        else
          $R_{v_1} \leftarrow R_{v_1} \cup p_2$ with the lowest cost
          $R_{v_1} \leftarrow R_{v_1} \cup t_1 : i(t_1) = i(p_2) + 1$
return S

4.1. Test instances and settings

The data set used is generated from Sampaio (2020)'s *0.ins* instance. The benchmark includes only one depot, one vehicle, and 50 requests. However, in our problem setting, we consider multiple depots and capacitated EVs. Thus, we modified the instance by adding vehicles, depots, and transfer nodes. The node attributes, e.g., coordinates and loads, were kept the same as the original instance; however, the late time-windows were extended considering EVs' batteries. We created 10 different instances by selecting the first $|\mathcal{R}|$ number of requests. The instances are labeled regarding the number of requests, depots, transfers, and vehicles, respectively. For example, instance 5_1_1_3 contains five requests, one depot, one transfer, and three vehicles. The total number of nodes in an instance is calculated as $2\times|\mathcal{R}|+2\times|D|+|\mathcal{T}|$. The generated data set can be listed as: 5_1_1_3, 7_1_1_3, 10_1_1_3, 12_1_1_5, 15_2_1_5, 17_2_1_7, 20_2_2_7, 25_2_2_7, 30_2_2_9, 30_2_3_9. In order to stress-test the hybrid heuristic, 70_2_3_9 is introduced by using minimum/maximum coordinates and time windows in the benchmark instance.

Another benchmark from Lyu and Yu (2021) is also used in the computational experiments. The benchmark includes four sets, and each has 90 instances as a total of 360. The sets are labeled as 3_8_4_4, 3_8_5_4, 4_8_4_4, and 5_8_4_4. While these instances are classified as "new", the previous ones are as "old" so as not to confuse readers with two different five-request instances. Only late time windows and vehicles are modified to be consistent with EVs.

Chevrolet Bolt EV is chosen as a default vehicle type as in He et al. (2012). According to Chevrolet (2019), Chevrolet Bolt EV can travel up to 238 miles on a full charge and can be driven 25 miles with one hour of charging. As we solve the problem for a day, we multiplied all late time windows, where the maximum is 180, by eight to extend them to 24 h (1440 min). For computational experiments, the EVs' capacities are set to 10 for all vehicles. Although parameter selection homogenizes all vehicles, the MILP and the hybrid heuristic can work with a heterogeneous fleet without any further modification.

The parameters t_{ij} are calculated by the Euclidean distance rule. The cost per mile is determined to be 10¢ to calculate the costs c_{ij}; however, it is a coefficient to multiply the distance, and it can vary between nodes. We assume that one unit of distance can be traveled in one time unit. All service times are assumed to be zero, but it can be included in the model as a constant parameter.

We performed the tests on a server equipped with Two Sky Lake CPUs @ 2.60 GHz, with 96 GB RAM per node, where each node is responsible of running 24 jobs simultaneously. The heuristic approach was coded in Java, and the MILP model was solved by IBM CPLEX 12.8.

Algorithm 6 Small-Destroy-Repair

Require: Solution S
same ← `false`, $\rho \leftarrow$ constant probability
Randomly select a request r for removal operator
$R_p \leftarrow$ current route of $p(r)$ in solution S
$R_d \leftarrow$ current route of $d(r)$ in solution S
$i(x) \leftarrow$ index of node x
if $R_p = R_d$ **then**
  same ← `true`
$R_p \leftarrow R_p \setminus p(r)$
$R_d \leftarrow R_d \setminus d(r)$
if same **then**
  $\alpha \leftarrow$ a random number: $\alpha \in [0, 1]$
  if $\alpha < \rho$ **then**
    $R_v \leftarrow R_v \cup \{p(r), d(r)\}$ with the lowest cost
  else
    for all $k \in \mathcal{V}$ **do**
      for all $l \in \mathcal{V}$ **do**
        if $k \neq l$ **then**
          $T_x \leftarrow$ transfers in vehicle x
          if $|T_k| = |T_l| = 0$ **then**
            $t \leftarrow \mathcal{T}[1]$
          if $|T_k| = 0$ **then**
            $R_{v_k} \leftarrow R_{v_k} \cup p(r)$ with the lowest cost
            $R_{v_k} \leftarrow R_{v_k} \cup t : i(t) = i(p(r)) + 1$
          else
            $R_{v_k} \leftarrow R_{v_k} \cup p(r) : i(p(r)) < i(t)$ with the lowest cost
          if $|T_l| = 0$ **then**
            $R_{v_l} \leftarrow R_{v_l} \cup d(r)$ with the lowest cost
            $R_{v_l} \leftarrow R_{v_l} \cup t : i(t) = i(d(r)) + 1$
          else
            $R_{v_l} \leftarrow R_{v_l} \cup d(r) : i(d(r)) < i(t)$ with the lowest cost
else
  if $\alpha < \rho$ **then**
    for all $k \in \mathcal{V}$ **do**
      for all $l \in \mathcal{V}$ **do**
        if $k \neq l$ **then**
          $T_x \leftarrow$ transfers in vehicle x
          if $|T_k| = |T_l| = 0$ **then**
            $t \leftarrow \mathcal{T}[1]$
          if $|T_k| = 0$ **then**
            $R_{v_k} \leftarrow R_{v_k} \cup p(r)$ with the lowest cost
            $R_{v_k} \leftarrow R_{v_k} \cup t : i(t) = i(p(r)) + 1$
          else
            $R_{v_k} \leftarrow R_{v_k} \cup p(r) : i(p(r)) < i(t)$ with the lowest cost
          if $|T_l| = 0$ **then**
            $R_{v_l} \leftarrow R_{v_l} \cup d(r)$ with the lowest cost
            $R_{v_l} \leftarrow R_{v_l} \cup t : i(t) = i(d(r)) + 1$
          else
            $R_{v_l} \leftarrow R_{v_l} \cup d(r) : i(d(r)) < i(t)$ with the lowest cost
  else
    $R_v \leftarrow R_v \cup \{p(r), d(r)\}$ with the lowest cost
return S

Table 3
The summary of the hybrid heuristic results.

Instance	Total request	Transferred request	Transferred requests (%)	Used vehicles
5_1_1_3_old	5	0	0.00	1
7_1_1_3_old	7	2	28.57	2
10_1_1_3_old	10	5	50.00	2
12_1_1_5_old	12	10	83.33	4
15_2_1_5_old	15	6	40.00	5
17_2_1_7_old	17	2	11.76	4
20_2_2_7_old	20	10	50.00	5
25_2_2_7_old	25	7	28.00	7
30_2_2_9_old	30	12	40.00	9
30_2_3_9_old	30	17	56.67	8
70_2_3_9_old	70	39	55.71	9

4.2. Heuristic results

The hybrid heuristic parameters are chosen after preliminary experiments by using sets of different values for each parameter. The initial temperature, cooling schedule, and the maximum non-improved iteration number are set to 1000, 0.95, and 1,000,000 $\times|\mathcal{R}|$, e.g., 5,000,000 iterations for 5_1_1_3_old, respectively. The temperature is raised back to its initial degree when it reaches a minimum value. The lowest possible temperature is set to the initial temperature multiplied by 0.2. Thus, the search can continue from a different region after one part of the solution space is examined.

Each instance is run 30 times to obtain normally distributed results. The summary of the best solutions of 30 replications is given in Table 3 that contains the total number of requests and the transferred request, the percentage of transferred requests, and the number of vehicles used. The results reveal that intra-route facilities are likely visited to transfer requests when the number of requests increases; however, there is no direct correlation between them. Transferring requests mostly happen as a result of how nodes are distributed over the network rather than the number of nodes in the network. The table only includes the old instances because instance 5_1_1_3_old (the smallest) gives a general idea for the new ones, which are smaller than 5_1_1_3_old instance.

Table 4
Example row from the raw table.

	f_m	t_m	$\%gap_m$	f_h	t_h	$\%gap_f$	$\%gap_t$
3_8_4_4_new_1	30.47	534.28	0.00	30.47	25.00	0.00	−95.32

4.3. Heuristic performance compared to MILP

The average of 30 objective values obtained from the hybrid heuristic are compared with the MILP's results. A one-hour time limit is used for the MILP model (24 h for the instances larger than five requests). The results are presented in Table 5. Letters f and t represent objective values and computational runtimes (in seconds), respectively. Subscript m stands for the mathematical model, and h is for the hybrid heuristic. The column $\%gap_m$ is the percentage gap between the upper and the lower bound when the mathematical model stops after one hour (24 h for the instances larger than five requests). The columns $\%gap_f$ and $\%gap_t$ describe the percentage gaps between the objective values (f_m and f_h) and the runtimes values (t_m and t_h), respectively. The gaps are calculated as in Equations (32) and (33).

$$\%gap_f = \frac{(f_h - f_m)}{f_m} \times 100 \tag{32}$$

$$\%gap_t = \frac{(t_h - t_m)}{t_m} \times 100 \tag{33}$$

The table rows indicate the statistics obtained from 90 instances' average results (one set): minimum, the first quartile, mean, median, the third quartile, and max. The link to the raw tables can be found in supplemantary materials. Observe that, the $\%gap_f$ and $\%gap_t$ values cannot directly be calculated from the columns f_m, f_h, t_m, and t_h. The reason is that the gaps are individually calculated for each instance, but only the statistics are presented in the table. For example, consider one instance of set 3_8_4_4_new (3_8_4_4_new_1), the row would be as follows in the raw table (Table 4), and the gap values can directly be calculated. Remember that, each set has 90 of this row, and each row shows the average of 30 runs under f_h and t_h.

With our current computer hardware, it is not possible to obtain even a feasible solution in 24 h for the instances with more than seven

Table 5
Heuristic results compared to MILP model.

		f_m	t_m	$\%gap_m$	f_h	t_h	$\%gap_f$	$\%gap_t$
3_8_4_4_new (1–90)	min	18.00	12.64	0.00	18.00	22.61	0.00	−98.17
	Q1	25.26	329.38	0.00	25.26	23.63	0.00	−96.43
	mean	27.27	548.99	0.00	27.30	24.62	0.13	−89.40
	median	27.25	500.52	0.00	27.27	24.57	0.00	−95.11
	Q3	29.02	713.08	0.00	29.12	25.50	0.00	−92.63
	max	38.72	1267.97	0.00	38.89	27.60	1.39	86.79
		f_m	t_m	$\%gap_m$	f_h	t_h	$\%gap_f$	$\%gap_t$
3_8_5_4_new (1–90)	min	18.00	17.06	0.00	18.00	21.79	−0.27	−98.87
	Q1	24.98	414.34	0.00	25.27	22.89	0.00	−97.46
	mean	27.24	734.26	0.00	27.28	23.80	0.13	−91.82
	median	27.25	661.79	0.00	27.25	23.72	0.00	−96.24
	Q3	29.02	929.15	0.00	29.03	24.53	0.06	−94.43
	max	38.49	1946.26	0.00	38.49	26.85	1.45	30.05
		f_m	t_m	$\%gap_m$	f_h	t_h	$\%gap_f$	$\%gap_t$
4_8_4_4_new (1–90)	min	20.86	256.97	0.00	21.41	31.76	−2.86	−99.11
	Q1	28.93	856.99	0.00	28.93	33.49	0.00	−98.74
	mean	31.61	1790.46	3.08	32.06	34.18	1.42	−96.68
	median	31.14	1379.22	0.00	31.86	33.98	0.86	−97.55
	Q3	33.33	2718.85	0.00	33.43	34.94	2.31	−96.01
	max	42.62	3607.91	31.67	44.36	36.99	5.58	−87.32
		f_m	t_m	$\%gap_m$	f_h	t_h	$\%gap_f$	$\%gap_t$
5_8_4_4_new (1–90)	min	28.48	1406.84	0.00	28.61	38.41	−42.98	−98.93
	Q1	36.58	3600.96	5.75	34.74	45.05	−13.93	−98.75
	mean	41.53	3334.45	19.27	38.52	45.59	−5.67	−98.56
	median	40.43	3604.80	18.24	38.59	45.88	−3.00	−98.72
	Q3	44.94	3605.83	28.04	41.47	46.59	5.27	−98.68
	max	58.69	3644.65	59.51	55.52	47.85	13.65	−96.67
		f_m	t_m	$\%gap_m$	f_h	t_h	$\%gap_f$	$\%gap_t$
5_1_1_3_old		43.98	1820.99	0.00	45.17	35.04	2.71	−98.08
7_1_1_3_old		46.53	7589.17	0.00	52.89	58.60	13.66	−99.22
10_1_1_3_old		–	–	–	64.22	107.00	–	–
12_1_1_5_old		–	–	–	88.21	199.06	–	–
15_2_1_5_old		–	–	–	102.30	308.80	–	–
17_2_1_7_old		–	–	–	125.98	517.61	–	–
20_2_2_7_old		–	–	–	142.05	778.80	–	–
25_2_2_7_old		–	–	–	169.00	2017.53	–	–
30_2_2_9_old		–	–	–	195.08	23066.91	–	–
30_2_3_9_old		–	–	–	196.68	15300.18	–	–
70_2_3_9_old		–	–	–	195.05	33097.65	–	–

requests. However, the results from small instances indicate that the hybrid heuristic may find the optimal solution on average 90% faster than the mathematical model. The negative $\%gap_f$ values indicate that the hybrid heuristic yields better solutions than the mathematical model with a one-hour (or 24 h) limit. The results highlight the problem complexity and justify the need for an efficient solution method.

4.4. Importance of transfers

In order to give readers a better understanding of how transfer nodes improve the delivery system's capability, we solve the one-hour-limited MILP model for all the instances by removing the transfer nodes. Again, f and t represent the objective values and the runtimes (in second), respectively, while w and wo stand for "with" and "without" transfers. In Table 6, the gap values are calculated as follows:

$$\%gap_f = \frac{(f_{wo} - f_w)}{f_w} \times 100 \tag{34}$$

$$\%gap_t = \frac{(t_{wo} - t_w)}{t_w} \times 100 \tag{35}$$

Table 3 indicates that vehicles do not travel to an intra-route facility for exchanging requests in small instances. However, the resulting routes reveal that EVs still visit intra-route facilities to get recharged, and Table 6 highlights that using transfer nodes may still improve the objective value by up to 129%. This emphasizes the importance of transfer nodes for EVs although they do not exchange requests. On the other hand, without transfer nodes, the computational runtimes decrease 40% on average as the total number of nodes and constraints is reduced. Δi represents the number of instances in the set become infeasible after removing the transfer nodes. All Δi values are zero for the instances with less than or equal to five requests. However, one might expect to observe positive Δi values for large instances by looking at Table 3 because intra-route facilities are visited to exchange requests in large instances. Although most of the 5_8_4_4_new instances are stopped by the time limit, the objective value is 24% better on average than the "without" transfer scenario. As we could not obtain results for the instances with more than five requests, we removed the last ten instances' results from the table (7_1_1_3 - 70_2_3_9).

4.5. Effect of valid inequalities on the runtimes

In addition to the first two comparisons, the impact of the valid inequalities (VI) is also investigated. Table 7 contains the objective values and the runtimes (in second) of the MILP model with and without the VI. The percent gap values ($\%gap_f$ and $\%gap_t$) are calculated as the previous subsection by using Eqs. (34) and (35). Columns gap_w and gap_{wo} show the percent gap between the upper and the lower bounds for "with" and "without" the VI when the MILP stops, respectively.

Although the MILP model is run with a one-hour time limit, the increase in average runtimes is 75%. Additionally, Δi values represent the number of instances, where even a feasible solution cannot be found

Table 6
Transfer nodes impact.

		f_w	t_w	f_{wo}	t_{wo}	$\%gap_f$	$\%gap_t$	Δi
3_8_4_4_new (1–90)	min	18.00	12.64	18.02	133.36	0.00	−69.93	0
	Q1	25.26	329.38	26.83	244.34	0.11	−50.44	
	mean	27.27	548.99	32.76	396.10	19.03	33.36	
	median	27.25	500.52	32.41	342.32	16.84	−34.71	
	Q3	29.02	713.08	38.08	511.19	32.45	−4.26	
	max	38.72	1267.97	48.66	893.15	49.22	1576.19	
		f_w	t_w	f_{wo}	t_{wo}	$\%gap_f$	$\%gap_t$	Δi
3_8_5_4_new (1–90)	min	18.00	17.06	18.02	128.52	0.00	−81.26	0
	Q1	24.98	414.34	26.83	238.18	0.11	−68.72	
	mean	27.24	734.26	32.76	395.04	19.13	4.79	
	median	27.25	661.79	32.41	339.50	17.37	−40.01	
	Q3	29.02	929.15	38.08	507.15	32.45	−13.27	
	max	38.49	1946.26	48.66	902.36	49.22	1205.02	
		f_w	t_w	f_{wo}	t_{wo}	$\%gap_f$	$\%gap_t$	Δi
4_8_4_4_new (1–90)	min	20.86	256.97	28.71	185.25	0.00	−90.29	0
	Q1	28.93	856.99	33.82	431.48	9.58	−67.38	
	mean	31.61	1790.46	41.46	946.55	30.51	−18.88	
	median	31.14	1379.22	38.57	931.75	30.58	−50.06	
	Q3	33.33	2718.85	46.97	1477.95	45.76	22.49	
	max	42.62	3607.91	68.49	1850.26	86.16	240.30	
		f_w	t_w	f_{wo}	t_{wo}	$\%gap_f$	$\%gap_t$	Δi
5_8_4_4_new (1–90)	min	28.48	1406.84	33.03	259.77	−25.62	−84.19	0
	Q1	36.58	3600.96	39.61	1005.37	6.08	−70.79	
	mean	41.53	3334.45	48.83	2160.93	24.58	−33.46	
	median	40.43	3604.80	44.64	2190.47	15.11	−31.50	
	Q3	44.94	3605.83	55.64	3604.09	36.34	−0.03	
	max	58.69	3644.65	72.38	3605.72	129.49	81.71	
		f_w	t_w	f_{wo}	t_{wo}	$\%gap_f$	$\%gap_t$	Δi
5_1_1_3_old		43.98	1820.99	49.45	151.37	12.44	−91.69	–

Table 7
Effect of valid inequalities.

		f_w	t_w	gap_w	f_{wo}	t_{wo}	gap_{wo}	$\%gap_f$	$\%gap_t$	Δi
3_8_4_4_new (1–90)	min	18.00	12.64	0.00	18.00	13.84	0.00	−1.58	−6.09	2
	Q1	25.26	329.38	0.00	25.18	358.33	0.00	0.00	−1.42	
	mean	27.27	548.99	0.00	27.47	1046.53	3.44	0.84	203.49	
	median	27.25	500.52	0.00	27.31	622.30	0.00	0.00	0.19	
	Q3	29.02	713.08	0.00	29.10	954.23	0.00	0.00	3.89	
	max	38.72	1267.97	0.00	41.67	3607.24	51.44	49.30	7853.74	
		f_w	t_w	gap_w	f_{wo}	t_{wo}	gap_{wo}	$\%gap_f$	$\%gap_t$	Δi
3_8_5_4_new (1–90)	min	18.00	17.06	0.00	18.00	18.73	0.00	−0.24	−77.81	4
	Q1	24.98	414.34	0.00	25.05	481.17	0.00	0.00	−1.39	
	mean	27.24	734.26	0.00	27.46	1088.60	3.10	0.59	94.56	
	median	27.25	661.79	0.00	27.30	766.77	0.00	0.00	0.49	
	Q3	29.02	929.15	0.00	29.03	1071.44	0.00	0.00	2.18	
	max	38.49	1946.26	0.00	38.49	3606.49	37.88	22.04	3200.02	
		f_w	t_w	gap_w	f_{wo}	t_{wo}	gap_{wo}	$\%gap_f$	$\%gap_t$	Δi
4_8_4_4_new (1–90)	min	20.86	256.97	0.00	20.86	252.20	0.00	−4.91	−4.53	16
	Q1	28.93	856.99	0.00	29.74	869.69	0.00	0.00	−1.19	
	mean	31.61	1790.46	3.08	31.94	1874.95	4.22	0.20	3.30	
	median	31.14	1379.22	0.00	31.14	1405.53	0.00	0.00	−0.02	
	Q3	33.33	2718.85	0.00	33.92	2770.37	0.01	0.00	0.41	
	max	42.62	3607.91	31.67	42.62	3606.67	29.41	10.55	261.24	
		f_w	t_w	gap_w	f_{wo}	t_{wo}	gap_{wo}	$\%gap_f$	$\%gap_t$	Δi
5_8_4_4_new (1–90)	min	28.48	1406.84	0.00	29.99	1616.99	0.00	−28.31	−2.50	27
	Q1	36.58	3600.96	5.75	37.15	3602.16	8.27	0.00	−0.03	
	mean	41.53	3334.45	19.27	41.51	3365.74	20.55	−0.56	0.02	
	median	40.43	3604.80	18.24	41.32	3605.39	18.21	0.00	0.00	
	Q3	44.94	3605.83	28.04	42.84	3606.19	32.27	0.00	0.03	
	max	58.69	3644.65	59.51	56.97	3646.80	59.51	19.11	3.57	
		f_w	t_w	gap_w	f_{wo}	t_{wo}	gap_{wo}	$\%gap_f$	$\%gap_t$	Δi
5_1_1_3_old		43.98	1820.99	0.00	47.65	3601.85	17.39	8.34	97.80	–

within the time limit without the VI over 90 instances. Observe that, even for small instances, the rate can reach up to 30%. In this table, again the instances with more than five requests are removed due to not able to find a solution within the time limit.

4.6. Effect of strict constraints

If the time-windows and the vehicle battery and capacity constraints are removed from our problem, then the problem becomes the PDP-T.

Table 8
Comparison of results from metaheuristic with and without strict constraints.

		f_h	t_h	f_{rh}	t_{rh}	$\%gap_f$	$\%gap_t$
3_8_4_4_new (1–90)	min	18.00	22.61	18.00	21.49	0.00	−4.39
	Q1	25.26	23.63	24.92	23.05	0.00	2.38
	mean	27.30	24.62	27.13	23.49	0.58	4.81
	median	27.27	24.57	27.04	23.43	0.08	4.83
	Q3	29.12	25.50	28.81	23.78	0.74	7.27
	max	38.89	27.60	36.99	26.52	5.14	14.91
		f_h	t_h	f_{rh}	t_{rh}	$\%gap_f$	$\%gap_t$
3_8_5_4_new (1–90)	min	18.00	21.79	18.00	21.77	−0.02	−3.10
	Q1	25.27	22.89	24.92	22.38	0.00	2.14
	mean	27.28	23.80	27.13	22.72	0.49	4.72
	median	27.25	23.72	27.05	22.60	0.08	4.85
	Q3	29.03	24.53	28.81	22.91	0.71	7.45
	max	38.49	26.85	36.99	25.99	4.06	13.42
		f_h	t_h	f_{rh}	t_{rh}	$\%gap_f$	$\%gap_t$
4_8_4_4_new (1–90)	min	21.41	31.76	21.73	31.21	−4.38	−4.02
	Q1	28.93	33.49	28.94	32.29	0.00	1.94
	mean	32.06	34.18	31.74	32.89	0.90	3.93
	median	31.86	33.98	31.43	32.85	0.54	4.06
	Q3	33.43	34.94	33.33	33.45	1.52	6.07
	max	44.36	36.99	41.98	35.06	5.68	10.45
		f_h	t_h	f_{rh}	t_{rh}	$\%gap_f$	$\%gap_t$
5_8_4_4_new (1–90)	min	28.61	38.41	28.61	43.49	−3.74	−13.57
	Q1	34.74	45.05	33.56	44.72	−0.80	−1.13
	mean	38.52	45.59	37.98	45.23	1.28	0.81
	median	38.59	45.88	37.91	45.27	0.42	1.11
	Q3	41.47	46.59	40.76	45.75	2.82	3.36
	max	55.52	47.85	50.70	47.32	9.88	6.45
		f_h	t_h	f_{rh}	t_{rh}	$\%gap_f$	$\%gap_t$
5_1_1_3_old		45.17	35.04	43.02	34.21	4.99	2.43
7_1_1_3_old		52.89	58.60	50.98	58.17	3.74	0.74
10_1_1_3_old		64.22	107.00	63.49	109.72	1.15	−2.48
12_1_1_5_old		88.21	199.06	84.90	202.31	3.90	−1.61
15_2_1_5_old		102.30	308.80	107.23	321.28	−4.60	−3.88
17_2_1_7_old		125.98	517.61	123.63	542.22	1.90	−4.54
20_2_2_7_old		142.05	778.80	147.61	791.86	−3.77	−1.65
25_2_2_7_old		169.00	2017.53	182.18	1521.61	−7.24	32.59
30_2_2_9_old		195.08	23066.91	223.47	2615.17	−12.71	782.04
30_2_3_9_old		196.68	15300.18	218.63	2622.52	−10.04	483.42
70_2_3_9_old		195.05	33097.65	204.02	32678.77	−4.40	1.28

In order to see the impact of these strict constraints, we relax them by assigning large-enough numbers to T_i, B_k, and Q_k parameters. We test this version of the problem with the hybrid heuristic approach.

The subscripts h and rh indicate the hybrid heuristic and the relaxed hybrid heuristic, respectively. The percentage gap columns are calculated as follows:

$$\%gap_f = \frac{(f_h - f_{rh})}{f_{rh}} \times 100 \tag{36}$$

$$\%gap_t = \frac{(t_h - t_{rh})}{t_{rh}} \times 100 \tag{37}$$

When the strict constraints are relaxed, it is expected that the objective value would improve. However, in Table 8, the results contradict our expectations. Remember that, the hybrid heuristic stops when the maximum number of non-improved iterations is reached and does not start the iterator until it finds the first feasible solution. Without the strict constraints, a feasible solution is found at the early stage of the algorithm. On the other hand, when the search space is stricter, the iterator starts when the algorithm already converges to a better solution than the initial one.

Resulting vehicle schedules indicate that transfer nodes are still visited even if EVs are not the part of the system in this experiment. We can conclude that intra-route facilities do not only contribute to the delivery system by providing chargers, but also help to reduce the total traveled distance.

4.7. Comparison with large neighborhood search

In this test, the impact of one particular neighborhood on the performance of the hybrid approach is evaluated. Remember that, Small-Destroy-Repair neighbor removes a request from the current solution and inserts it back with the lowest cost, which is the LNS part of the approach. We apply this test to compare our method with others as Adaptive LNS is a widely used technique to solve logistics problems in the literature, and we already mentioned the adaptive layer of it. Thereby, the remaining neighborhood definitions, besides Small-Destroy-Repair, are removed from the hybrid heuristic approach.

The new subscript lns in Table 9 refers to the new approach. The percentage gaps are calculated as follows:

$$\%gap_f = \frac{(f_{lns} - f_h)}{f_h} \times 100 \tag{38}$$

$$\%gap_t = \frac{(t_{lns} - t_h)}{t_h} \times 100 \tag{39}$$

The results reveal that the hybrid heuristic finds better results on average than the new approach in a shorter time period for the instances with less than 10 requests. However, the new approach may yield better solutions than the hybrid heuristic for larger sets if one can bear with a 604% (on average) longer computational runtime. The findings are validated by the paired t-test. The result of the test highlights that there is not a significant difference between the objective values of two methods (2.84E−09). On the other hand, the test returns 0.51 (5.13E−01) for the runtime comparison.

Table 9
Comparison of LNS and SA.

		f_h	t_h	f_{lns}	t_{lns}	$\%gap_f$	$\%gap_t$
3_8_4_4_new (1–90)	min	18.00	22.61	22.22	31.27	0.02	19.02
	Q1	25.26	23.63	30.05	32.95	15.34	29.71
	mean	27.30	24.62	33.73	33.09	23.36	34.71
	median	27.27	24.57	33.11	33.08	22.27	34.63
	Q3	29.12	25.50	37.50	33.27	32.24	39.82
	max	38.89	27.60	46.91	34.38	54.70	48.16
		f_h	t_h	f_{lns}	t_{lns}	$\%gap_f$	$\%gap_t$
3_8_5_4_new (1–90)	min	18.00	21.79	22.22	30.66	0.02	21.82
	Q1	25.27	22.89	30.38	32.84	14.40	34.84
	mean	27.28	23.80	33.77	33.00	23.72	38.98
	median	27.25	23.72	32.52	33.01	21.53	39.23
	Q3	29.03	24.53	37.99	33.20	33.85	42.90
	max	38.49	26.85	46.70	34.12	47.74	55.84
		f_h	t_h	f_{lns}	t_{lns}	$\%gap_f$	$\%gap_t$
4_8_4_4_new (1–90)	min	21.41	31.76	28.57	58.79	1.37	68.28
	Q1	28.93	33.49	35.88	62.54	18.59	79.45
	mean	32.06	34.18	41.17	62.74	28.79	83.79
	median	31.86	33.98	40.42	62.95	27.28	83.86
	Q3	33.43	34.94	45.10	63.27	37.27	88.99
	max	44.36	36.99	55.83	64.14	63.25	101.07
		f_h	t_h	f_{lns}	t_{lns}	$\%gap_f$	$\%gap_t$
5_8_4_4_new (1–90)	min	28.61	38.41	38.43	101.62	2.96	117.04
	Q1	34.74	45.05	43.55	109.60	16.46	136.16
	mean	38.52	45.59	48.09	110.18	25.55	142.03
	median	38.59	45.88	47.47	110.76	23.65	141.62
	Q3	41.47	46.59	50.59	111.23	32.10	145.80
	max	55.52	47.85	65.13	112.77	62.20	184.59
		f_h	t_h	f_{lns}	t_{lns}	$\%gap_f$	$\%gap_t$
5_1_1_3_old		45.17	35.04	49.40	70.90	9.37	102.34
7_1_1_3_old		52.89	58.60	56.50	184.32	6.83	214.54
10_1_1_3_old		64.22	107.00	66.02	274.91	2.81	156.93
12_1_1_5_old		88.21	199.06	87.86	1581.16	−0.40	694.31
15_2_1_5_old		102.30	308.80	90.85	1975.40	−11.19	539.70
17_2_1_7_old		125.98	517.61	106.14	5542.16	−15.75	970.72
20_2_2_7_old		142.05	778.80	113.29	4442.30	−20.24	470.40
25_2_2_7_old		169.00	2017.53	133.89	9068.51	−20.77	349.49
30_2_2_9_old		195.08	23066.91	155.71	17413.99	−20.18	−24.51
30_2_3_9_old		196.68	15300.18	155.21	17135.76	−21.08	12.00
70_2_3_9_old		195.05	33097.65	139.85	219120.72	−28.30	562.04

5. Conclusion

In this study, we propose a novel *NP*-Hard logistic problem that combines the pickup and delivery problem with transfers and electric vehicles. To the best of our knowledge, we are the first to study the PDP-T with multiple depots and capacitated heterogeneous EVs with partial charging. We propose a MILP model for the problem and update the existing model in the literature. Although the model's runtime is improved by several valid inequalities, for large instances of more than five requests, obtaining optimal and even feasible solutions is not possible. Thus, we provide a metaheuristic approach (the hybrid heuristic), that contains eight neighborhood definitions. Numerous tests are conducted on the MILP model and the hybrid heuristic approach. The results reveal that the hybrid heuristic finds the optimal solutions 90% faster than CPLEX for small instances where the MILP model can be solved exactly. The advantages of transfer nodes for the delivery systems with EVs are provided by two experiments. The results in Section 4.6 highlight that transfer nodes help to reduce the total traveled distance and also provide chargers for EVs to stay operational. If EVs are not recharged during their daily tasks, then it is possible that not all requests are delivered (Section 4.4).

The current paper has some limitations to be improved. Battery recharging and consumption depend on a linear function of time and traveled distance, respectively. Besides, transfer nodes provide infinite capacity to charge vehicles and store requests. These issues will be addressed in future research.

Our follow-up research is solving the PDP-TWT as a *Location Routing Problem* by combining routing and intra-route facility location decisions. Further research could cover drones as an alternative vehicle type, and the impact of transfer nodes can be assessed on such networks. Note that, drones have more strict battery and capacity constraints, but they travel faster. Moreover, the selective PDP-T could help researchers to explore more on how transfer nodes improve the overall customer satisfaction while minimizing travel times and waiting times.

CRediT authorship contribution statement

Cansu Agrali: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Seokcheon Lee:** Supervision.

Data availability

The benchmark generated for this study and the raw tables can be accessed through this link.

Acknowledgments

We thank Dr. Mario Ventresca, Dr. Nihat Oner, Dr. Sungbum Jun, and anonymous reviewers for their comments and advice to improve our study.

References

Arslan, A. M., Agatz, N., Kroon, L., & Zuidwijk, R. (2019). Crowdsourced delivery—A dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, *53*(1), 222–235.

Ben Alaia, E., Dridi, I. H., Bouchriha, H., & Borne, P. (2013). Optimization of the multi-depot Multi-vehicle pickup and delivery problem with time windows using genetic algorithm. In *2013 international conference on control, decision and information technologies* (pp. 343–348).

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, *15*, 1–31.

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*(1), 8–15.

Borndorfer, R., Grotschel, M., Klostermeiner, F., & Kuttner, C. (1997). Telebus berlin: vehicle routing scheduling in a dial a ride system. *Konrad Zuse Zentrum fur Information Technik Berlin*.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300–313.

Chen, T. D., Kockelman, K. M., & Hanna, J. P. (2016). Operations of a shared, autonomous, electric vehicle fleet: Implications of vehicle & charging infrastructure decisions. *Transportation Research Part A: Policy and Practice*, *94*, 243–254.

Chen, J. F., & Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, *57*(5), 579–587.

Chevrolet (2019). *2019 bolt EV electric car*. Chevrolet, Retrieved 2019-11-11, from www.chevrolet.com/index/vehicles/2019/cars/bolt-ev/overview.html.

Cortés, C. E., Matamala, M., & Contardo, C. (2010). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, *200*(3), 711–724.

Danloup, N., Allaoui, H., & Goncalves, G. (2018). A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. *Computers & Operations Research*, *100*, 155–171.

Dondo, R., Méndez, C. A., & Cerdá, J. (2008). Optimal management of logistic activities in multi-site environments. *Computers & Chemical Engineering*, *32*(11), 2547–2569.

Furtado, M. G. S., Munari, P., & Morabito, R. (2017). Pickup and delivery problem with time windows: A new compact two-index formulation. *Operations Research Letters*, *45*(4), 334–341.

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). Metaheuristic algorithms in modeling and optimization. *Metaheuristic applications in structures and infrastructures*, 1–24.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: Latest advances and new challenges*. New York, NY, United States: Springer.

He, Y., Venkatesh, B., & Guan, L. (2012). Optimal scheduling for charging and discharging of electric vehicles. *IEEE Transactions on Smart Grid*, *3*(3), 1095–1105.

Hou, Y., Zhong, W., Su, L., Hulme, K., Sadek, A. W., & Qiao, C. (2016). TASeT: Improving the efficiency of electric taxis with transfer-allowed rideshare. *IEEE Transactions on Vehicular Technology*, *65*(12), 9518–9528.

Irnich, S. (2000). A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. *European Journal of Operational Research*, *122*(2), 310–328.

Jun, S., Lee, S., & Yih, Y. (2021). Pickup and delivery problem with recharging for material handling systems utilising autonomous mobile robots. *European Journal of Operational Research*, *289*(3), 1153–1168.

Keskin, M., & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C (Emerging Technologies)*, *65*, 111–127.

Koç, C., Laporte, G., & Tükenmez, İ. (2020). A review on vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, Article 104987.

Lau, H. C., & Liang, Z. (2002). Pickup and delivery with time windows: algorithms and test case generation. *International Journal on Artificial Intelligence Tools*, *11*(03), 455–472, Publisher: World Scientific Publishing Co.

Li, S., Huang, Y., & Mason, S. J. (2016). A multi-period optimization model for the deployment of public electric vehicle charging stations on network. *Transportation Research Part C (Emerging Technologies)*, *65*, 128–143.

Li, H., & Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, *12*(02), 173–186.

Lin, C., Choy, K. L., Ho, G. T. S., Chung, S. H., & Lam, H. Y. (2014). Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, *41*(4, Part 1), 1118–1138.

Lin, J., Zhou, W., & Wolfson, O. (2016). Electric vehicle routing problem. *Transportation Research Procedia*, *12*, 508–521.

Lyu, Z., & Yu, A. J. (2021). *Data for: The pickup and delivery problem with transshipments*. Mendeley Data, V2. https://data.mendeley.com/datasets/w925jygjct/2.

Masson, R., Lehuede, F., & Peton, O. (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, *47*(3), 344–355.

Masson, R., Lehuédé, F., & Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, *41*, 12–23.

McKinnon, A., Browne, M., Whiteing, A., & Piecyk, M. (2015). *Green logistics: Improving the environmental sustainability of logistics*. Kogan Page Publishers.

Meisenzahl, M. (2021). *Amazon's first electric delivery vans are now making deliveries — see how they were designed*. Business Insider, Retrieved 2021-04-01, from https://www.businessinsider.com/amazon-creating-fleet-of-electric-delivery-vehicles-rivian-2020-2.

Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, *79*, 115–129.

Mosheiov, G. (1994). The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research*, *79*(2), 299–310.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal Für Betriebswirtschaft*, *58*(1), 21–51.

Peng, Z., Al Chami, Z., Manier, H., & Manier, M.-A. (2019). A hybrid particle swarm optimization for the selective pickup and delivery problem with transfers. *Engineering Applications of Artificial Intelligence*, *85*, 99–111.

Rais, A., Alvelos, F., & Carvalho, M. (2014). New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, *235*(3), 530–539.

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

Ruland, K., & Rodin, E. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & Mathematics with Applications*, *33*(12), 1–13.

Sahin, B., Yilmaz, H., Ust, Y., Guneri, A. F., & Gulsun, B. (2009). An approach for analysing transportation costs and a case study. *European Journal of Operational Research*, *193*(1), 1–11.

Sampaio, A. (2020). The benefits of transfers in crowdsourced pickup-and-delivery systems (dataset), Mendeley. Retrieved from https://data.mendeley.com/datasets/pywzcgyzrv/2.

Sampaio, A., Savelsbergh, M., Veelenturf, L. P., & Woensel, T. V. (2020). Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. *Networks*, *76*(2), 232–255.

Savelsbergh, M. W. P., & Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, *29*(1), 17–29.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500–520, Publisher: INFORMS.

Statista, Research Department (2021). *U.S. e-commerce market size 2016–2023*. Statista, Retrieved 2021-04-01, from https://www.statista.com/statistics/272391/us-retail-e-commerce-sales-forecast/.

Sundstrom, O., & Binding, C. (2012). Flexible charging optimization for electric vehicles considering distribution grid constraints. *IEEE Transactions on Smart Grid*, *3*(1), 26–37.

Turkeš, R., Sörensen, K., Hvattum, L. M., Barrena, E., Chentli, H., Coelho, L. C., . . ., et al. (2020). Data for a meta-analysis of the adaptive layer in adaptive large neighborhood search. *Data in Brief*, *33*, Article 106568.