Transportation Research Part E 192 (2024) 103782

Contents lists available at ScienceDirect

Transportation Research Part E

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

Express shipments with autonomous robots and public transportation

Umut Ermağan [a,*], Barış Yıldız [b], F. Sibel Salman [b]

[a] Department of Management Science and Engineering, University of Waterloo, Waterloo, ON, Canada
[b] College of Engineering, Koç University, Istanbul, Turkey

ARTICLE INFO

Keywords:
Express shipment
Public transportation
Autonomous robots
Rolling horizon
Machine learning
Sustainable logistics

ABSTRACT

Growing urbanization, exploding e-commerce, heightened customer expectations, and the need to reduce the environmental impact of transportation ask for innovative last-mile delivery solutions. This paper explores a new express shipment model that combines public transportation with Autonomous Robots (ARs) and studies its real-time management. Under dynamic demand arrivals with short delivery time promises, we propose a rolling horizon framework and devise a machine learning-enhanced Column Generation (CG) methodology to solve the real-time AR dispatching problem. The results of our numerical experiments with real-world delivery demand data show the significant potential of the proposed system to reduce travel time, vehicle traffic, emissions, and noise. Our results also reveal the efficacy of the learning-based CG methodology, which provides almost the same quality solutions as the classical CG approach with much less computational effort.

1. Introduction

The growth in urbanization and increasing interest in e-commerce, together with customer expectations for faster deliveries and the need for cheaper and greener transportation bring unprecedented challenges in last-mile delivery. Aiming to address these aspects, researchers and practitioners have been searching for new delivery concepts and technologies that can reduce the economic, social, and environmental costs of last-mile delivery operations while meeting the stringent service quality requirements.

One of the promising new technologies is package delivery with Autonomous Robots (ARs), which can help carriers to cut down on personnel costs and, at the same time, reduce vehicle traffic, emissions, and accidents (Chan, 2017 and Hoffmann and Prause, 2018). As a significant advantage compared to other unmanned delivery modes, such as drones and autonomous vehicles, ARs do not raise as many safety concerns and thus are subject to fewer regulatory/safety limitations in the urban areas over which they can operate and the type of cargo they can carry (Roboticsbiz, 2021). The fact that AR fleets can be fully controlled and routed by a central platform where a single operator can supervise multiple ARs is another distinguishing feature to reduce operational costs (He et al., 2022). In contrast, regulations require each drone to be flown by a dedicated pilot (FAA, 2020). Moreover, ARs can carry multiple packages and simply use the pedestrian ways without needing any special infrastructure. Considering these features, ARs are likely to be more suitable for last-mile deliveries of low-value items in densely populated urban areas. Drones can be a preferable option for delivering urgent and valuable goods such as healthcare and fashion products to remote or rural regions (Simoni et al., 2020). As such, major parcel delivery companies have already started considering the widespread use of AR fleets. For example, FedEx has recently announced its significant investment in ARs to be used in last-mile delivery (FedEx, 2020), while JD Logistics

* Correspondence to: Department of Management Science and Engineering, University of Waterloo, 200 University Ave W, Waterloo, ON, N2L 3G1, Canada.
E-mail address: uermagan@uwaterloo.ca (U. Ermağan).

https://doi.org/10.1016/j.tre.2024.103782
Received 10 April 2024; Received in revised form 22 August 2024; Accepted 14 September 2024
Available online 26 September 2024
1366-5545/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

announced the positive results of its pilot application with more than 13,000 deliveries using ARs during the peak of the Covid-19 pandemic in early 2020 in Wuhan, China (Wang, 2022). In their 2016 report, McKinsey & Company forecasted that autonomous vehicle delivery models are poised to surpass other alternatives in the future, signaling a significant shift in the logistics and delivery landscape (Joerss et al., 2016).

Besides ARs, pickup and delivery stations (PDSs) also enable new delivery models to meet the last-mile challenges. PDSs contain smart lockers where packages are stored. A customer can open a locker with an individual password sent by logistics companies (Ulmer and Streng, 2019). This way, couriers need not go to the customer's address; instead, they go to the nearest PDS. As stated in several recent studies in the literature, delivery companies can take advantage of unique opportunities that PDSs offer to develop flexible and customer-centric solutions. These solutions have the potential to reduce operational costs, pollutant emissions, and the number of failed deliveries often encountered in classical home delivery (Edwards et al., 2009; Behnke, 2019; Rohmer and Gendron, 2020; Kızıl and Yıldız, 2023). A successful example comes from DHL, which established around 14,000 PDSs, ensuring that its customers can walk to pick up or drop off their packages in less than 10 min (DHL, 2022). PDSs can also provide an efficient means to enable the widespread use of ARs in delivery services, especially considering that not every street in a city may be suitable for ARs in the foreseeable future (He et al., 2022).

Although using ARs in combination with PDSs provides several advantages, some challenges, which are discussed in the review studies by Engesser et al. (2023) and Alverhed et al. (2024), also arise. Due to the limited battery charge, ARs cannot travel long distances. Moreover, the delivery by ARs may take longer than by a courier since ARs use pedestrian ways with walking speed. However, as we propose in our study, these challenges can be tackled with novel delivery strategies. More precisely, we investigate a new last-mile delivery concept in which ARs utilize the public transportation network in cities to expand their reach and increase their riding speed. Considering the operations of a delivery company that provides express shipment (i.e., two-hour or three-hour delivery) between PDSs, we introduce the Dynamic Pickup and Delivery Problem with Autonomous Robots (DPDP-R) in an express delivery setting. Similar to Anika et al. (2021), we assume that ARs rented by the delivery company are located and recharged in AR hubs (i.e., AR centers) at different locations in the city center. In the proposed system, as delivery orders arrive, the delivery company assigns them to available ARs and determines their routes. To sustain the promised delivery times, the company leverages additional delivery capacity offered by dedicated couriers, typically from a third-party provider, whenever the capacity of ARs falls short. The primary objective of the company is to minimize operational costs associated with the utilization of both ARs and dedicated couriers.

Similar to the existing studies, we envision the ARs considered in this study are designed such that they can get on/off public transportation vehicles in a reasonable time (e.g., less than 30 s) so that the passengers do not incur extra waiting times at bus stops for ARs to get on/off. To be more specific, in the remainder of the paper, we will call ARs with such capabilities as "rider-bots". The proposed transportation network comprises PDSs, rider-bot hubs, and bus stations. The rider-bots are tasked with picking up the products from PDSs and delivering them to their destinations (i.e., the destination PDSs). Since we focus on the operational level of the last-mile delivery with ARs in this study, we assume the PDS locations are predetermined after solving the respective strategic level location problem that is studied by several recent papers (Lin et al., 2020a,b; Deutsch and Golany, 2018). The rider-bots follow predetermined bot-friendly paths, which need not be the shortest paths. Since multiple pickups and drop-offs are possible in a single rider-bot trip, we optimize the routes of the rider-bots as well as their task assignments.

For the solution of the DPDP-R, we propose a rolling horizon approach, which is a widely adopted decision-making practice in highly dynamic environments (Powell, 2019). Also known as the model-predictive control in the optimal control literature, the main motivation for the rolling horizon approach is to address the dynamism of the underlying problem, particularly dynamic demand arrival, with short re-optimization times (e.g., with five-minute time epochs). To solve the problem within each time epoch of the rolling horizon framework, we develop a CG algorithm. However, due to its high complexity, classical approaches fall short of solving the underlying pricing problem quickly, which limits the applicability of the CG approach in a rolling horizon framework with frequent re-optimizations. To address this challenge, we propose a novel learning-based approach for solving the pricing problem efficiently. In particular, we employ a machine-learning model to filter out some paths during the path enumeration in the labeling algorithm. Since the destination hubs of the rider-bots that have completed their deliveries can determine the cost of serving future customers, we present three different positioning policies to decide the destination hubs of the rider-bots after their deliveries. We explore the potential benefits of the suggested delivery model and test the performance of our solution methodology, including the three positioning policies, on the instances derived from real-world historical data provided by our industry partner Trendyol (trendyol.com), the largest e-commerce company in Turkey.

In the next section, we review the related literature on Pickup and Delivery Problems (PDPs) with ARs and CG approaches employing machine learning algorithms and discuss the contribution of our study. The problem definition and developed CG approach are presented in Section 3 and Section 3.2, respectively. We explain the proposed learning-based labeling algorithm in Section 4 and detail the positioning policies in Section 5. Section 6 presents the results of our computational experiments and managerial insights. We conclude in Section 7.

2. Literature review

As a significant variant of Vehicle Routing Problems (VRPs), PDPs can be mainly categorized into three groups (Berbeglia et al., 2010): many-to-many problems in which any location in the network can be an origin or a destination for any commodity (e.g., Hernández-Pérez and Salazar-González, 2007), one-to-many-to-one problems in which the service is carried out from depot to customers or from customers to depot (e.g., Gribkovskaia and Laporte, 2008), and one-to-one problems in which each commodity has an origin and a destination point (e.g., Cordeau et al., 2008). Our problem falls into the class of one-to-one PDPs. In this section, we focus on studies that involve ARs in delivery due to our context and those that propose novel CG approaches integrating machine learning algorithms as we consider in our study.

2.1. Delivery systems with ARs

Utilizing ARs in delivery is a relatively new concept. Poeting et al. (2019) take ARs into account in a two-tier distribution system. While the parcels are transported by a single truck to the micro-depots in the first leg of the system, ARs are responsible for the last leg of the delivery. They propose an agent-based simulation model supported by optimization algorithms and investigate the required number of ARs to serve a certain number of customers while minimizing travel distances and total delay in delivery. The truck's route is optimized by solving the well-known Traveling Salesman Problem (TSP), whereas the allocation and scheduling of the robots are optimized via a Simulated Annealing heuristic. Similarly, Bakach et al. (2020) investigate a two-tier delivery network. They aim to minimize the number of robot hubs and the operational cost of serving the customers. They present a mixed integer programming model in which facility location and multi-trip robot deliveries are considered simultaneously. Their computational study quantifies the advantages of robot-based deliveries over conventional deliveries. Yu et al. (2020) analyze the use of robots in a two-echelon last-mile delivery system designed for areas such as a campus and a housing estate. In the study, the customers are served by robots carried by vehicles to the rendezvous locations where the robots start their tours. A mixed integer program is formulated, and two hybrid metaheuristics are presented. Alfandari et al. (2022) develop an exact algorithm (i.e., a Benders decomposition approach) for the last-mile delivery with ARs in the context of two-tier urban logistics. In contrast to the existing literature, Yu et al. (2023) propose a two-echelon hybrid van-based robot delivery system that utilizes vans both to transport robots and serve customers. The authors show that the proposed model outperforms two alternative models: a van-based delivery model that relies solely on robots transported by vans and a two-echelon delivery model in which packages are transported to local robot hubs before being delivered by robots at the second echelon.

Boysen et al. (2018) investigate truck-based robot last-mile delivery where ARs with a single package delivery capacity are carried by a single truck to deliver the goods to customers. The truck is not allowed to serve the customers directly. The ARs return to their depots after delivery to avoid the waiting times of the truck and satisfy more customers. The objective is to minimize the number of late deliveries. Chen et al. (2021b,a) study a variant of the VRP in which both vehicles and robots can make the delivery. In particular, they decide when to dispatch the robots from the vehicles to serve nearby customers while the vehicles serve a customer. They minimize the total operational time. Ostermeier et al. (2022) propose a truck-and-robot routing system considering time windows and the costs of each type of vehicle. Following a hierarchical approach, they first solve the truck routing problem and then the robot scheduling problem. In the last step, the truck's route is improved with a local search algorithm. On the other hand, since the orders requested by a customer can exceed the capacity of a single AR when routing the ARs, Olejarz (2020) consider the Synchronized Split Delivery Vehicle Routing Problem (SSDVP) in which the delivery for a customer can be split to be carried by several ARs, but must arrive at the same time. Similarly, we allow the same customer's products to be carried by several rider-bots. However, we guarantee the delivery of all the products in the same time window (i.e., in 2–3 h), albeit not at the same time.

Although the truck-and-robot tandems reduce operational and environmental costs, they do not completely eliminate the use of conventional vehicles (e.g., diesel vans). Also, for an express shipment situation, as we focus on in our study, enough consolidating opportunities do not arise to justify using trucks to carry multiple robots to reduce operational costs. Using the existing networks and excess capacity of public transit vehicles can provide many advantages to logistics providers in this sense. For delivery in crowded cities, innovative concepts can be used to reduce traffic flow, environmental and social harms, and operational costs. In this context, Sonneberg et al. (2019) select the best depot set to open among existing stations in the city while optimizing the assignment of customers to these stations and minimizing the operational cost and the number of robots in delivery simultaneously. Mourad et al. (2021) consider the pick-up and delivery problem where the delivery is carried out by the ARs that utilize the public transportation network. They model the problem as a two-stage stochastic problem where the capacity availability of public vehicles for the ARs at the departure points is unknown. However, the orders are known at the planning time. After planning the routes of the ARs, they are updated over the set of scenarios based on the known probability distribution of passenger demands for public vehicles. They propose a sample average approximation framework in which the stochastic problem is solved with an Adaptive Large Neighborhood Search (ALNS). In our study, taking into account the future research directions highlighted in Mourad et al. (2021) and the comprehensive review study by Srinivas et al. (2022), we explore the dynamic fulfillment of customer requests and the integration of partial recharging strategies for robots in an express delivery framework. We propose a rolling horizon algorithm to optimize real-time operations. We allow rider-bots to use more than one public transit vehicle, carry more than one package with them, and make pickup stops before they deliver all the previously collected orders. These factors significantly alter the solution structure and add complexity to the optimization problem.

2.2. Integrating machine learning algorithms into CG approaches

CG is one of the most effective and widely used methods to solve large-scale problems (Desrosiers et al., 1984, Ribeiro and Soumis, 1994 and Fink et al., 2019), in particular showing significant success in vehicle routing problems. CG starts to solve the linear relaxation of the problem with a subset of columns (i.e., variables), namely *the restricted master problem*, and improves the current solution iteratively with the new columns generated by solving a pricing problem (Barnhart et al., 1998, Desaulniers et al., 2006 and Shen et al., 2022). Two challenges that need to be addressed in the CG approach are selecting the most promising columns among the generated ones and solving the pricing problem, which is often NP-hard, efficiently.

Morabit et al. (2021) utilize a machine learning classification algorithm (i.e., a Graph Neural Network) to select the most promising columns. They show the efficiency of their algorithm with two different problems (i.e., the vehicle and crew scheduling problem and the vehicle routing problem with time windows). Chi et al. (2022) use a deep reinforcement learning framework

(i.e., Q-learning) for column selection. Thus, they can select the columns considering the later iterations of the CG. They show that the CG converges much faster with the proposed approach than the greedy policy (i.e., adding the column(s) with the most favorable reduced cost(s) at each iteration of the CG) when solving the cutting stock problem and the vehicle routing problem with time windows.

Taking a different approach, Shen et al. (2022) use the Support Vector Machine classification algorithm to generate high-quality columns by predicting the optimal solution to the pricing problem. Morabit et al. (2022) also propose a learning-based algorithm to solve the resource-constrained shortest path problem similarly to our study. They use machine learning models to select the arcs more likely to be used in the network. With arc selection, the reduction in network size facilitates faster problem-solving using the labeling algorithm. However, when the algorithm fails to generate the required paths in the reduced network, it necessitates reverting to the complete network, resulting in increased computational time. During the training phase, a reprocessing step is performed for each instance, requiring the algorithm's training to be conducted anew for each case. Unlike their study, we integrate our machine learning model into the labeling algorithm to select the promising partial paths without eliminating any arc in the network. Thus, as an advantage, we can use the same trained machine learning model for different networks since we select the promising partial paths rather than arcs whose chances of being used may change according to the characteristics of the network, as we consider in our numerical experiments in Section 6.4.2.

2.3. Main contributions

With respect to the existing literature, the main contributions of this study are the following:

- We adapt a novel public transport- and AR-based delivery system and show the potential benefits of such a system for express shipments under dynamic demand arrivals in real-life applications. Through extensive numerical experiments with real-world demand data, we show the significant potential of the proposed new approach to reduce vehicle kilometers, pollutant emissions, and noise.
- We propose a rolling horizon framework in which a CG algorithm is solved repeatedly with dynamic data to make rider-bot-order assignments and optimize the rider-bot routes. To achieve short enough computation times for large-scale problem instances, we develop a novel learning-based approach, which can speed up label-setting algorithms for solving the pricing problem in CG. Our results show that, for the problem instances we study, the proposed methodology provides high-quality solutions (i.e., 1.04% optimality gap on average) in approximately one-third of the solution time required for the classical column generation algorithm without learning-based speed up.
- We explore three different re-positioning policies for rider-bots to consider future delivery order arrivals that are not yet known to the system at a decision epoch. We compare the performance of these policies in our numerical experiments to derive valuable managerial insights.

3. Problem definition, model formulation, and column generation

We consider the DPDP-R in a planning horizon $[0, t_{max}]$, where 0 and t_{max} represent the beginning and end of the planning horizon, respectively. Let $T = \{0, t_1, t_2, \dots, t_{max}\}$ be a set of discrete time points, e.g., each separated by five-minute intervals. Let $L = \{l_1, l_2, \dots, l_{|L|}\}$ be the set of PDSs, $B = \{b_1, b_2, \dots, b_{|B|}\}$ be the set of bus stations and $H = \{h_1, h_2, \dots, h_{|H|}\}$ be the set of rider-bot hubs. Thus, the node set can be defined as $N = L \cup B \cup H$.

Let K be the set of orders. The retailer locks the package associated with order $k \in K$ in a nearby PDS. Each order $k \in K$ has an earliest pickup time e_k (i.e., the time that the package is available in a PDS for pickup) and a latest delivery time τ_k (i.e., the latest time that the package can be delivered). Moreover, each $k \in K$ has a pickup location $o_k \in L$ and a delivery destination $d_k \in L$. Order $k \in K$ becomes inactive when assigned to a rider-bot or a *courier* (a third-party service provider). At time $t \in T$, K_t represents the set of *active orders* that consists of the orders that have arrived before $t-1$, but have not been served yet and those arriving between time $t-1$ and t. Similarly, we define L_t as the set of *active* PDSs in the system at time $t \in T$, i.e., $L_t = L_t^P \cup L_t^D$, where $L_t^P = \{l \in L : l = o_k, k \in K_t\}$ and $L_t^D = \{l \in L : l = d_k, k \in K_t\}$.

Let R be the set of rider-bots and V be the set of buses capable of carrying rider-bots. V_t is the set of buses in service at time $t \in T$ (i.e., $V_t \subseteq V$). R_t is a subset of R such that it consists of those rider-bots ready for express delivery at the rider-bot hubs at time $t \in T$. $B_v \subset B$ is the set of bus stations on the planned route of bus $v \in V_t$. We define the set of *active bus stations* B_t as the union of the bus stations of the buses traveling at time $t \in T$ (i.e., $B_t = \cup_{v \in V_t} B_v$).

We consider a homogeneous fleet of rider-bots, each having a capacity for multiple packages denoted by κ and a limited travel range (i.e., the maximum travel time attained with a fully charged battery). The remaining battery charge of rider-bot $r \in R_t$ at time $t \in T$ is denoted by β_r^t. The rider-bots that complete their tours may return to any rider-bot hub in the network, where they can recharge.

We consider a rolling horizon planning approach. In this approach, we determine at each decision epoch $t \in T$ rider-bot-order assignments as well as the routes of the rider-bots by solving a static problem with the information available at time t. As time progresses, the schedule is reoptimized if there is an active order in the system at the next decision epoch. Fig. 1 illustrates the framework of the rolling horizon approach that we adopt to solve the DRDP-R.

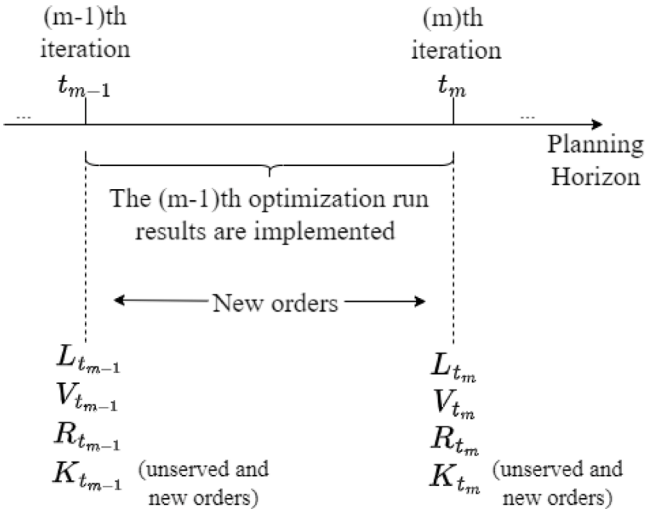


Fig. 1. Rolling horizon framework.

3.1. Static problem

We define the static problem (SP) on a directed, complete graph $G_t = (N_t, A_t)$ where the vertex set N_t consists of the PDS set L_t, bus station set B_t and rider-bot hub set H. Set A_t is defined as $A_t = \{(i,j) : i \in N_t, j \in N_t\}$. Each arc is associated with a non-negative AR travel time d_{ij}. We assume that all AR travel times satisfy the triangle inequality and do not change over time. We define A_t^B as the set of arcs connecting the bus station nodes on the planned routes of the available buses (i.e., $A_t^B = \cup_{v \in V_t} A_v$ where A_v is the set of arcs on the planned route of bus $v \in V_t$). Note that while traveling on a bus, rider-bots do not use their battery charge, and the travel times of the public transit buses, denoted by $d'_{ij}, (i,j) \in A_t^B$, are determined by the timetables as planned by the public transit authorities. Moreover, it is also conceivable that rider-bots could be recharged during their travel on the bus. However, for the sake of simplicity in application and technical requirements, we do not consider recharging on buses in our delivery system, although it does not change our model. The implementation of the proposed algorithm is explained in the following sections.

Since a PDS may be visited by a rider-bot more than once, the graph G_t is augmented with a set of dummy nodes defined for active PDSs to permit multiple visits. For active PDSs with multiple orders, we create a dummy node for each order except one. Hence, the total number of orders at a PDS equals to 1, i.e., each order has its own pickup and delivery station. Similarly, we duplicate the incoming and outgoing arcs to/from the PDSs in the network for the dummy nodes. Since each pickup station has only one order after creating the dummy nodes, we can refer to the orders by their pickup locations as we do in the model presented in the following section (see the definition of the variable y_i). We also note that multiple visits apply to bus stations as well. However, we do not need to create dummy nodes for bus stations due to the structure of the algorithm proposed in Section 4.

The SP aims to find the rider-bot-order assignments and routes of the rider-bots with minimum rental cost, as mentioned in Section 1.

3.2. Path-based formulation and column generation approach

In this section, we present a path-based formulation for the SP. Our formulation is built with the directed simple paths in G_t, where the starting node o_p and ending node e_p of path p are hubs. The set of all the paths in G_t is denoted by P. The set of the nodes in path $p \in P$ is denoted by $N(p)$. The subset of nodes from the starting node o_p to node $i \in N(p)$ on a path $p \in P$ is denoted by $N_i(p)$. The set of arcs in a path $p \in P$ is denoted by $A(p)$. A path is called feasible if the load of a rider-bot does not exceed its total capacity after the pickup stops on the path. In addition, the energy cost c_p of path p (i.e., the energy cost of the arcs constituting path p, except for the arcs in A_t^B along which the rider-bot travels on a bus) must be less than the current remaining time due to the battery charge of rider-bot $r \in R_t$. If the rider-bot uses bus $v \in V_t$ in path p, the cumulative travel time denoted by d_{pb} of the arcs from starting node o_p to bus station $b \in B_t$ where the rider-bot takes bus v must be smaller than the arrival time of the bus to the bus station (i.e., $d_{pb} \leq \alpha_{vb}$). We note that the cumulative travel time of the paths includes the waiting time at the nodes, denoted by w (e.g., w_b refers to the waiting time for a bus at bus station $b \in B_t$). We assume that waiting at a node does not consume the battery charge of the rider-bot. The paths visiting pickup station $i \in L_t^P$ and being feasible for rider-bot $r \in R_t$ are denoted by $P(i,r)$.

With the sets, parameters, and decision variables defined in Table 1, we develop our path-based formulation P_{SP} as follows.

$$\min \sum_{p \in P} \sum_{r \in R_t} \gamma d_p x_{pr} + \sum_{i \in L_t^P} \rho y_i \tag{1}$$

s.t.

Table 1
Notation used in the static problem.

Sets	Description
T	Set of discrete time points
H	Set of rider-bot hubs
L_t^P	Set of active pickup locations at time $t \in T$
L_t^D	Set of active delivery locations at time $t \in T$
L_t	Set of active PDSs at time $t \in T$, i.e., $L_t = L_t^P \cup L_t^D$
B_t	Set of active bus stations at time $t \in T$
N_t	Set of nodes, i.e., $H \cup L_t \cup B_t$
V_t	Set of buses traveling at time $t \in T$
B_v	Set of bus stations on the planned route of bus $v \in V_t$
A_t	Set of arcs at time $t \in T$
A_t^B	Set of arcs connecting the bus station nodes on the planned routes of the buses traveling at time $t \in T$, i.e., $A_t^B = \cup_{v \in V_t} A_v$
A_v	Set of arcs on the planned route of bus $v \in V_t$
K_t	Set of active orders at time $t \in T$
R_t	Set of rider-bots ready for express delivery at the rider-bot hubs at time $t \in T$
P	Set of simple paths
$P(i,r)$	Set of simple paths including node $i \in L_t^P$ and being feasible for rider-bot $r \in R_t$
$N(p)$	Set of nodes in path $p \in P$
Parameters	**Description**
β_r^t	Remaining battery charge of rider-bot $r \in R_t$ at time $t \in T$
α_{vb}	Arrival time of bus $v \in V_t$ to bus station $b \in B_t$
w_i	Waiting time at node $i \in N_t$
e_k	The earliest pickup time of order $k \in K_t$
τ_k	The latest delivery time of order $k \in K_t$
d_{ij}	Travel time of arc $(i,j) \in A_t$
d'_{ij}	Travel time of arc $(i,j) \in A_t^B$ when traveling on a bus
d_p	Total travel time of path $p \in P$
d_{pb}	Cumulative travel time of the arcs in path $p \in P$ from starting node o_p to bus station $b \in B_t$ if $b \in N(p)$
c_{ij}	Energy cost of arc $(i,j) \in A_t$
c_p	Energy cost of path $p \in P$
γ	Unit rental cost per minute to rent a rider-bot
ρ	Cost to rent a courier
Decision variables	**Description**
x_{pr}	Binary variable that indicates whether path $p \in P$ is used by rider-bot $r \in R_t$
y_i	Binary variable that indicates whether pickup station $i \in L_t^P$ is assigned to a courier

$$\sum_{r \in R_t} \sum_{p \in P(i,r)} x_{pr} + y_i \geq 1 \qquad \forall i \in L_t^P \tag{2}$$

$$\sum_{p \in P(r)} x_{pr} \leq 1 \qquad \forall r \in R_t \tag{3}$$

$$x_{pr} \in \{0,1\} \qquad \forall p \in P, r \in R_t \tag{4}$$

$$y_i \in \{0,1\} \qquad \forall i \in L_t^P \tag{5}$$

The objective is to minimize the total rental cost of the rider-bots and the costs associated with the couriers. Constraints (2) ensure that a rider-bot or a courier visits each active pickup location. We note that the paths that include the pickup location of an order also include its delivery location. Constraints (3) ensure that each rider-bot is assigned to at most one path. Constraints (4) and (5) define the variables.

We use the CG procedure to solve the path-based formulation, where the linear relaxation $LP_{P_{SP}}$ is solved iteratively by extending with new feasible paths with favorable reduced costs. We start the CG with a restricted master formulation that does not contain any path because we can obtain a feasible solution by assigning all the orders to the couriers. We solve the pricing problem presented in the next subsection to find the favorable reduced cost paths. When we do not find such a path, we terminate the CG.

Once the CG is terminated, we apply the common column generation heuristic (e.g., see Behnke et al., 2021 and Ermağan et al., 2022) by solving the integer program with only the paths generated throughout the CG iterations to obtain an integer solution. Our computational studies showed that such a procedure could provide high-quality solutions (i.e., with less than 0.3% optimality gap on average). Therefore, we did not see the need to apply a branch and price method.

3.3. Pricing problem

The pricing problem aims to find a feasible path p from $o_p \in H$ to $e_p \in H$ with a negative reduced cost for each rider-bot. The reduced cost π_p of path $p \in P$ can be defined as follows.

$$\pi_p = \gamma d_p - \sum_{i \in L_t^P : i \in N(p)} \delta_i + \lambda_r \tag{6}$$

where δ and λ are the dual variables associated with constraints (2) and (3), respectively.

We update the arc costs (i.e., $\gamma d_{ij}, (i,j) \in A_t$) to find the paths with negative reduced costs as follows.

$$\bar{d}_{ij} = \begin{cases} \gamma d_{ij} - \delta_i & \text{if } i \in L_t^P \\ \gamma d_{ij} & \text{otherwise} \end{cases} \quad \forall (i,j) \in A_t \tag{7}$$

We note that if the rider-bot travels on a bus along arc $(i,j) \in A_t^B$, we consider the cost of traveling on a bus (i.e., $\gamma d'_{ij}$) as the updated travel time $\bar{d}_{ij}$.

The pricing problem is solved on the directed graph G_t for each $(o_p, e_p, r) \in H \times H \times R_t$, where the origin of path p and the current location of rider-bot $r \in R_t$ are the same (i.e., $o_p = o_r$). If a path with negative reduced cost is found (i.e., $\pi_p < 0$), it is added to the restricted master formulation.

The presented pricing problem is an extension of the Resource Constrained Shortest Path Problem (RCSPP), which falls into the class of NP-hard problems. Several efficient algorithms such as Lozano et al. (2016) are proposed for the RCSPP in the literature. Unlike the RCSPP, our pricing problem has precedence and bus constraints requiring a tailored approach. Therefore, we propose a novel Forward Labeling Algorithm (FLA) presented in the next section to solve it quickly.

4. Learning-based forward labeling algorithm

In the FLA, forward partial paths are built from the origin hub $o_p \in H$ to the destination hub $e_p \in H$. Each forward partial path starts with an initial label that holds the information about the resource consumption, and the labels are updated as the forward partial paths are extended toward the destination hub (Gschwind et al., 2018).

For the label of the forward partial path $(o_p, \ldots, i)$, we record the following information: the last visited node i, the cumulative reduced cost $\bar{d}_f$, the cumulative travel time d_f, the cumulative energy cost c_f, the load of rider-bot q_f after visiting node i, the set of completed orders C_f, i.e., the orders whose pickup and delivery destinations have been visited, the set of open orders O_f, i.e., the orders that have been picked up, but not yet delivered. Thus, the label of each forward partial path is denoted by $\mathscr{L}_f = (i, \bar{d}_f, d_f, c_f, q_f, C_f, O_f)$.

A new partial path with label $\mathscr{L}'_f = (j, \bar{d}'_f, d'_f, c'_f, q'_f, C'_f, O'_f)$ is created with the extension of the partial path with label $\mathscr{L}_f$ along arc (i,j) as follows:

$$\bar{d}'_f = \begin{cases} \bar{d}_f + \gamma d'_{ij} + \gamma w_i & \text{if } (i,j) \in A_t^B \text{ and the rider-bot travels on a bus} \\ \bar{d}_f + \bar{d}_{ij} + \gamma w_i & \text{otherwise} \end{cases} \tag{8}$$

$$d'_f = \begin{cases} d_f + d'_{ij} + w_i & \text{if } (i,j) \in A_t^B \text{ and the rider-bot travels on a bus} \\ d_f + d_{ij} + w_i & \text{otherwise} \end{cases} \tag{9}$$

$$c'_f = \begin{cases} c_f & \text{if } (i,j) \in A_t^B \text{ and the rider-bot travels on a bus} \\ c_f + d_{ij} & \text{otherwise} \end{cases} \tag{10}$$

$$q'_f = \begin{cases} q_f + 1 & \text{if } j \in L_t^P \\ q_f - 1 & \text{if } j \in L_t^D \\ q_f & \text{otherwise} \end{cases} \tag{11}$$

$$C'_f = \begin{cases} C_f \cup \{k\} & \text{if } j \in L_t^D : d_k = j, k \in K_t \\ C_f & \text{otherwise} \end{cases} \tag{12}$$

$$O'_f = \begin{cases} O_f \cup \{k\} & \text{if } j \in L_t^P : o_k = j, k \in K_t \\ O_f \backslash \{k\} & \text{if } j \in L_t^D : d_k = j, k \in K_t \\ O_f & \text{otherwise} \end{cases} \tag{13}$$

We note that we also consider the waiting time at node j in the calculations of $\bar{d}_f$ and d'_f to apply the strong domination rule proposed in Proposition 1 below if $j \in L_t^P$ and the arrival time to node j is smaller than its earliest pickup time (e.g., $\bar{d}'_f = \bar{d}_f + \bar{d}_{ij} + \gamma w_i + \gamma w_j$). Extending a partial path along arc (i,j) is allowed only if the feasibility conditions are satisfied. The feasibility conditions ensure that $q'_f \le \kappa$ and $c'_f \le \beta_r^t$. Moreover, the extension is allowed if either $j = o_k$, $k \in K_t$ and $k \notin O_f \cup C_f$ (i.e., $j \in N_t$ is the pickup location of order $k \in K_t$ and the sets O_f and C_f do not include order k), or $j = d_k$, $k \in K_t$, $k \in O_f$ and $d'_f \le \tau_k$ (i.e., $j \in N_t$ is the delivery destination of order $k \in K_t$, the set O_f includes order k and the total travel time of the path is less than the latest delivery time of order k), or $j \in B_t$ (i.e., $j \in N_t$ is a bus station), or $j = e_p$ and $O_f = \emptyset$ (i.e., $j \in N_t$ is the destination hub

and the set O_f is empty). We note that we take into account all available buses at node i in the extension of the partial path with label $\mathscr{L}_f$ if arc $(i,j) \in A_t^B$.

As stated in , the arc cost transformation of the pricing problem is done only for the outgoing arcs of the pickup stations. Thus, the reduced cost matrix still satisfies the triangle inequality for the bus and delivery stations (i.e., $\bar{d}_{im} + \bar{d}_{mj} \geq \bar{d}_{ij}$ for all $(i,j) \in A_t, m \in L_t^D \cup B_t$). In other words, visiting a bus or a delivery station does not make a path cheaper. We call this property *bus and delivery triangle inequality*, similar to what Ropke and Cordeau (2009) call *delivery triangle inequality* where the network consists of only hubs, pickup and delivery stations. The bus and delivery triangle inequality enables us to apply three additional rules, which make the algorithm work faster: (1) We allow the rider-bot to go to a bus station if there is an available bus (i.e., if $j \in B_t$, an extension along arc $(i,j) \in A_t^B$ is allowed if $d_{pj} \leq \alpha_{vj}$ for any $v \in V_t$ where $j \in B_v$), (2) we force the rider-bot to travel on a bus if it goes to a bus station, using the pedestrian ways. More precisely, in the extension above, if node $i \in N_t$, node $j \in B_t$, and the rider-bot does not travel on a bus along arc $(i,j) \in A_t$, we only allow the next extension of the partial path with label $\mathscr{L}'_f$ along arc $(j,l) \in A_t^B$ on a bus, (3) we can use the following stronger domination rule developed based on the ideas from Dumas et al. (1991) and Ropke and Cordeau (2009) to avoid the enumeration of all feasible paths in the network:

Proposition 1 (*Strong Domination Rule*). *The partial path with label* $\mathscr{L}_f^1 = (i, \bar{d}_f^1, d_f^1, c_f^1, q_f^1, C_f^1, O_f^1)$ *dominates the partial path with label* $\mathscr{L}_f^2 = (j, \bar{d}_f^2, d_f^2, c_f^2, q_f^2, C_f^2, O_f^2)$ *if* $i = j$, $\bar{d}_f^1 \leq \bar{d}_f^2$, $d_f^1 \leq d_f^2$, $c_f^1 \leq c_f^2$, $C_f^1 \subseteq C_f^2$, *and* $O_f^1 \subseteq O_f^2$.

Proof. Assume the partial path with label $\mathscr{L}_f^1$, denoted by $P(\mathscr{L}_f^1)$ dominates the partial path $\mathscr{L}_f^2$, denoted by $P(\mathscr{L}_f^2)$. Then $c_f^1 \leq c_f^2$, $O_f^1 \subseteq O_f^2$ and every feasible extension of $P(\mathscr{L}_f^2)$ to destination hub e_p is a feasible extension of $P(\mathscr{L}_f^1)$, assuming that waiting at a node does not incur an energy cost. It is easy to see that the buses available for $P(\mathscr{L}_f^2)$ are also available for $P(\mathscr{L}_f^1)$ because $d_f^1 \leq d_f^2$. If the extension of $P(\mathscr{L}_f^1)$ to destination hub e_p is not feasible, one can eliminate both partial paths. Since $C_f^1 \subseteq C_f^2$, any pickup station that can be visited by $P(\mathscr{L}_f^2)$ can also be visited by $P(\mathscr{L}_f^1)$. Due to the bus and delivery station inequality, any feasible extension of $P(\mathscr{L}_f^1)$ to destination hub e_p has a reduced cost that is either smaller or equal to the reduced cost than the same extension of $P(\mathscr{L}_f^2)$. As a result, the best extension of $P(\mathscr{L}_f^1)$ is at least as good as the best extension of $P(\mathscr{L}_f^2)$. □

With the strong domination rule in Proposition 1, we can remove the dominated partial paths (i.e., redundant partial paths that are not already in the optimal solution) in advance. If the cost matrix does not satisfy the triangle inequality for the delivery stations, we can restrict the last condition (i.e., $O_f^1 \subseteq O_f^2$) in Proposition 1 to $O_f^1 = O_f^2$, which makes the domination rule more restrictive, i.e., it would be weaker than the dominance rule in Proposition 1 since less number of redundant partial paths could be eliminated.

Taking advantage of FLA's enumeration feature, we solve the pricing problem for each $(o_p, e_p) \in H \times H$ by considering the highest charged rider-bot at starting hub o_p, unlike what has been stated in Section 3.3. After obtaining the feasible and non-dominated partial paths that arrive at e_p for the highest charged rider-bot, we check their reduced costs for each rider-bot at the same hub to add to the master problem. Thus, we can significantly speed up the algorithm as we see in our computational results.

4.1. Learning phase

Since the FLA enumerates all feasible and non-dominated paths in the network as stated in Section 4, an additional elimination rule is needed to solve the DPDP-R in large-scale networks. The idea is to set a threshold for the total number of partial paths (i.e., $\mathscr{T}$) and consult a machine learning algorithm to choose the most promising ones for the next extension step when we have more than $\mathscr{T}$ partial paths. More precisely, after all possible extensions of partial paths along the arcs have been considered, we classify new feasible and non-dominated partial paths (i.e., extended partial paths) into one of two classes, i.e., *"good"* and *"no-good"* partial paths if their total number exceeds the threshold $\mathscr{T}$. To classify them, we set another threshold, i.e., the classification threshold $\mathscr{R}$. We remove no-good partial paths (i.e., partial paths whose prediction scores (as known as confidence scores) of being in the class of good partial paths are less than $\mathscr{R}$) from our partial path set. Then we continue with good partial paths for the next extension.

Since it is possible to eliminate some of the partial paths that might be in the optimal solution, in the proposed *"learning based"* pruning, we also use a diversification (also known as exploration) strategy. After solving the pricing problem, we create a tabu list for each starting hub to store the promising no-good partial paths (e.g., no-good partial paths with prediction scores greater than 0.5 where the classification threshold $\mathscr{R}$ is 0.7). In the next iteration of the CG, we do not allow the machine learning algorithm to eliminate a certain number of *no-good* partial paths with higher prediction scores, which are in the tabu list (e.g., the first 10 no-good partial paths that are in the tabu list when sorted by non-increasing prediction score). The overview of the proposed FLA, namely *the Learning-based Forward Labeling Algorithm (L-FLA)* is shown in Fig. 2.

The next subsections explain how we devise the machine learning algorithm.

4.1.1. Feature selection

Feature selection is one of the most important steps in devising a learning algorithm since it can significantly impact the accuracy and complexity of the learning algorithm (Hall, 1999). A feature set is determined with the consideration of learning targets. Also, the problem may require specific features representing the characteristics of partial paths.

We first divide the features into three groups, considering our problem. Then, we determine the features in each group in the light of the target labels (i.e., good and no-good partial paths). The first group (i.e., the first seven features in Table 2), namely *the general features*, consists of features that we think can be used in similar problems. These features aim to capture the characteristics

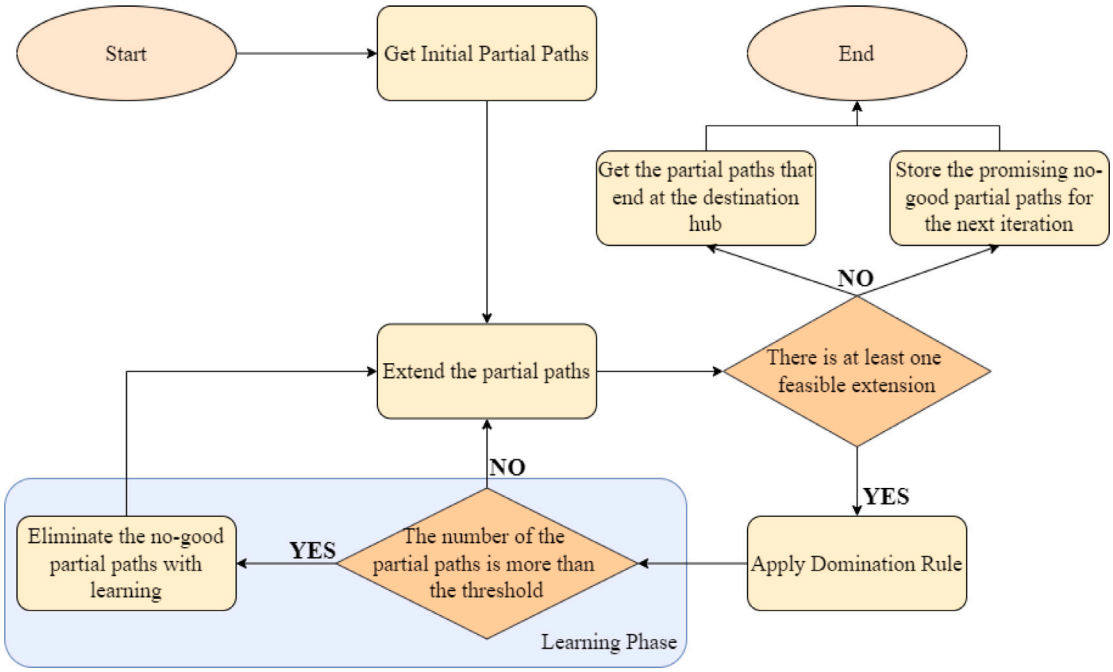


Fig. 2. Overview of the learning-based forward labeling algorithm.

Table 2
Definitions of the features.

ID	Feature	Definition
1	Number of Nodes	The total number of nodes that constitute the partial path
2	Starting Hub	The hub that the partial path starts (i.e., the first node of the partial path)
3	Destination Hub	The hub that the partial path tries to reach
4	Travel Time	The travel time of the partial path
5	Number of Packages	The number of packages on the rider-bot at the last node of the partial path (i.e., the number of open orders)
6	Delivered Packages	The number of delivered packages (i.e., the number of completed orders)
7	Reduced Cost	The cost of the partial path that is calculated with the reduced arc costs
8	Remaining Battery	Remaining battery of the rider-bot in case of using the partial path
9	Energy Cost	The energy cost of the partial path (i.e., the travel time that does not include waiting time at nodes and travel time on buses)
10	Number of Bus Stations	The total number of bus stations used by the partial paths
11	Number of Buses	The total number of buses used by the rider-bot along the partial path

of a general PDP problem. Besides these general features, the problem considered in this study also requires some specific features related to the rider-bots and the underlying network. Thus, the second group, namely the *energy features*, consists of the features F8 and F9 that aim to capture the recharging needs of the rider-bots when searching for new paths. Similarly, in the third group, we consider the *network features*, features F10 and F11, extracted from the characteristics of the underlying network.

Fig. 3 presents the importance scores for the features we use in our model, indicating their impact on the model's accuracy. As expected, feature F7 holds the highest importance score within the feature set, given that the pricing problem aims to find a path with a negative reduced cost. Following feature F7 in importance are the features F8 and F5, which pertain to the battery level and the number of packages on the rider-bot, respectively. This can be explained by the fact that the number of packages affects the potential number of deliveries along the route. At the same time, the battery level is crucial for determining the path's feasibility and impacts the rider-bot's ability to complete subsequent deliveries. Although the other features have relatively small impacts on

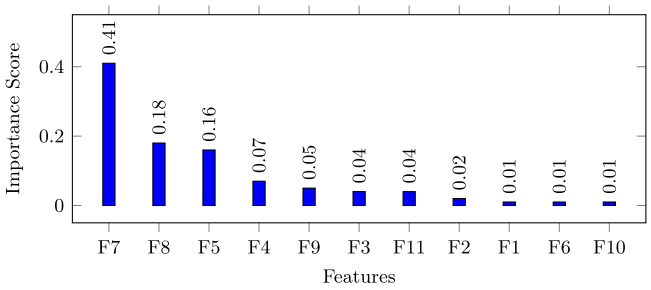


Fig. 3. Impacts of features.

classification accuracy individually, their combined influence contributes to satisfactory classification scores (see Section 4.1.3 for a detailed discussion). Therefore, we keep all of the features to train our classification model.

4.1.2. Generating the training data

After deciding on the feature set, we need a training data set. The training set consists of the partial paths generated by the CG on different instances (i.e., scenarios) derived from the assumed distribution of order arrivals (i.e., the Poisson Distribution) based on the company's historical data. The dominated partial paths during the CG iterations are labeled as no-good partial paths, while those constituting the paths added to the master problem are labeled as good partial paths. More precisely, for the negative reduced cost path p where $N(p) = \{o_p, i, j, e_p\}$, the partial paths (o_p, i) and (o_p, i, j) are labeled as good partial paths in the training set. We note that the CG does not use the learning phase of the L-FLA when creating the training set.

4.1.3. The choice of the machine learning algorithm

Since the learning phase of the L-FLA separates the partial paths into two groups (i.e., good and no-good partial paths), any supervised classification algorithm can be used in this phase. Based on our preliminary studies (see Fig. A.8 in Appendix A for a comparison of accuracy scores among supervised machine learning algorithms) and considering the success in solving similar problems in the literature (e.g., Hottung and Tierney, 2019 and Nair et al., 2020), we prefer a Multi-layer Feed-forward Neural Network (MFNN), which falls into the class of deep learning methods that can learn more complex functions. An MFNN consists of three components: the input layer (i.e., the first layer), the hidden layer(s), and the output layer (i.e., the last layer) (Svozil et al., 1997). Each layer also consists of the *neurons* (i.e., processing units) connected to other neurons in the previous and subsequent layers with weights. Data flows from the input layer to the output layer through the hidden layer(s). In the training of the MFNN, a back-propagation algorithm is used (Sazli, 2006).

Like other machine learning algorithms, the MFNN also has some hyperparameters that must be tuned. Tuning can be performed by experiments or optimization techniques such as the cross-validated grid-search, also provided by the software Python (Scikit-Learn, 2022). Since we achieved a satisfying accuracy score of 94%, with precision, recall, and F1 scores at 93%, 94%, and 93%, respectively, through our experiments using the k-fold cross-validation method, we decided to keep the obtained values. Thus, the number of hidden layers is set to 5; each has 200, 100, 50, 10, and 3 neurons from the input layer to the output layer, respectively. While the hyperbolic tangent activation function is used as the activation function, the optimization algorithm is determined as the stochastic gradient algorithm. The learning rate, regulation rate, and batch size are set to 0.1, 0.001, and 500, respectively. The number of input layer neurons already equals the number of the features explained in the previous subsection. In contrast, the number of output layer neurons equals the number of target labels (i.e., good and no-good partial paths).

Once the hyperparameters in the MFNN are tuned, we train our learning algorithm with the training sets created as described in the previous subsection. Note that since the training procedure is carried out before the delivery operation starts, they do not affect the computational time of the L-FLA.

5. Positioning policies

Once a rider-bot completes its delivery, the decision on which hub it should head to is important since it can affect the cost of serving future customers. We propose and test three positioning policies that specify which hub the rider-bot will return to after delivering its packages. Note that the initial locations of the rider-bots are decided according to the density of the PDSs around the rider-bot hubs. We define the policies and discuss their pros and cons below.

5.1. Nearest hub policy

The first policy, namely the *Nearest Hub Policy (NHP)*, requires the rider-bot to return to the nearest hub to the last visited PDS (i.e., the PDS to which it delivers its last package). The paths obtained from the P_{SP} are followed during the delivery operation since it already provides the shortest paths between the hubs.

The NHP builds shorter paths than other policies presented in the following subsections. Thus, the rider-bots can be recharged more frequently when they return to the hub before being dispatched for the next delivery. However, a drawback is that this policy does not consider the next steps of the delivery operation (i.e., future customers).

5.2. Starting hub policy

The *Starting Hub Policy (SHP)* requires the rider-bot to return to the hub where it started its delivery. Since the rider-bots tend to serve in a particular area to be able to return to their starting hubs due to the limited battery charge, this policy can be considered a divide-and-serve strategy, an example of which is applied by Huang et al., 2018. However, integrating the public transportation network allows the rider-bots to extend the borders of their service areas. The SHP guarantees a certain level of service by the rider-bots since they always return to their starting hubs after their delivery, and therefore, the hubs usually have available rider-bots. However, in this policy, the rider-bots cannot deliver some packages (e.g., those whose delivery stations are too far from the hub) due to the limited battery charge. An advantage of this policy is that future demand arrivals are indirectly considered by deciding the initial distribution of the rider-bots in the hubs. But note that since this allocation stays the same, this policy cannot adapt to the changing demand arrival rates.

5.3. Two-threshold policy

Due to the uncertainty in future demands, the delivery company may need to dispatch rider-bots from any hub at any time during its operation. The scarcity of rider-bots in a particular hub can increase the costs due to the need for dedicated couriers. Although the number of rider-bots is controlled in the previous policy (i.e., the SHP), they are forced to return to their home hubs after each delivery. Moreover, the deployment of rider-bots is decided at the beginning of the operation and does not change during the operation.

In this policy, namely *Two-threshold Policy (TP)*, the number of rider-bots at each hub during the delivery operation is controlled similarly to the (s, S) policy in inventory management (e.g., Freeman, 1957 and Barron, 2019). Under the (s, S) policy, when the stock level is s or less, the related products are ordered to increase it up to S. Unlike the (s, S) policy, in our case, we have to manage the number of rider-bots at each hub by considering the rider-bots on hand. Therefore, we aim for the number of rider-bots at each hub at time $t \in T$ to be between the stock level threshold values s_{ht} and $S_{ht}, h \in H, t \in T$ as much as possible.

To determine the values of s_{ht} and S_{ht} for each hub at time $t \in T$, we solve the P_{SP}, assuming that each hub has a sufficiently large number of rider-bots at time $t \in T$. We first generate a set of scenarios Ω, where each scenario $\omega \in \Omega$ is composed of a set of demands obtained from the historical data or the assumed distribution of demand arrivals based on that data. Then, we solve the P_{SP} for each scenario. Let η^{ω}_{ht} be the number of the rider-bots dispatched from hub $h \in H$ at time $t \in T$ under scenario $\omega \in \Omega$. Once we obtain the number of rider-bots η^{ω}_{ht} at time $t \in T$ for each hub and each scenario, we calculate the s_{ht} and S_{ht} values with the help of Eqs. (14) and (15). Note that in the computation of the values of s_{ht} and S_{ht}, $h \in H, t \in T$, we also consider the values of η^{ω}_{ht}, $h \in H$, $\omega \in \Omega$ in the six consecutive time intervals (i.e., the following 30 min) after $t \in T$, where the values of $\eta^{\omega}_{h(T+1)}$, $\eta^{\omega}_{h(T+2)}$, $\ldots$, $\eta^{\omega}_{h(T+6)}$ are set to 0 for each scenario $\omega \in \Omega$ and hub $h \in H$, for more realistic values to keep the availability of rider-bots at each hub under control. The two threshold parameters are defined as

$$s_{ht} = \frac{\sum_{i=0}^{6} \sum_{\omega} \eta^{\omega}_{h(t+i)}}{|\Omega|} - \sigma_{ht} \qquad \forall h \in H, t \in T \tag{14}$$

$$S_{ht} = \frac{\sum_{i=0}^{6} \sum_{\omega} \eta^{\omega}_{h(t+i)}}{|\Omega|} + \sigma_{ht} \qquad \forall h \in H, t \in T \tag{15}$$

where $\sigma_{ht}, h \in H, t \in T$ is the sample standard deviation. Obtaining the threshold values above does not affect the computational time of the proposed algorithm (i.e., L-FLA) since it is performed before the delivery operation.

In the delivery operation, the destinations of the rider-bots starting a delivery within time interval $t \in T$ are decided according to the stock level threshold values s_{ht} and S_{ht} of the hubs at time $t \in T$. The idea is to modify the objective function values of some paths found during the CG iterations before applying the CG heuristic to manipulate the solution in the desired direction.

Once the CG is terminated, we create three sets: (1) set $\underline{\mathcal{H}} = \{h_{s_1}, h_{s_2}, \ldots, h_{s_{|\underline{\mathcal{H}}|}}\}$ consisting of the hubs with the stock level below s_{ht}, sorted in non-ascending order by the difference between their current stock levels and s_{ht} values, (2) set $\overline{\mathcal{H}} = \{h_{S_1}, h_{S_2}, \ldots, h_{S_{|\overline{\mathcal{H}}|}}\}$ consisting of the hubs with the stock level above S_{ht}, sorted in non-ascending order by the difference between their current stock levels and S_{ht} values, and (3) set $\overline{\underline{\mathcal{H}}} = \{h_{S_{|\overline{\mathcal{H}}|+1}}, h_{S_{|\overline{\mathcal{H}}|+2}}, \ldots, h_{S_{|\overline{\mathcal{H}}|+|\underline{\mathcal{H}}|}}\}$ consisting of the hubs whose stock levels are between s_{ht} and S_{ht}, sorted in non-ascending order by the difference between their current stock levels and s_{ht} values.

After the above sets are created, we define the set $\mathcal{H} = \overline{\mathcal{H}} \cup \overline{\underline{\mathcal{H}}}$ (i.e., $\mathcal{H} = \{h_{S_1}, h_{S_2}, \ldots, h_{S_{|\overline{\mathcal{H}}|}}, h_{S_{|\overline{\mathcal{H}}|+1}}, h_{S_{|\overline{\mathcal{H}}|+2}}, \ldots, h_{S_{|\overline{\mathcal{H}}|+|\underline{\mathcal{H}}|}}\}$). We also define a set for each pair of the hubs in the sets $\mathcal{H}$ and $\underline{\mathcal{H}}$, consisting of the paths found by CG (e.g., the set for the pair of h_{S_1} and h_{s_1}, denoted by $\mathcal{P}^{s_1}_{S_1}$, consists of the paths with origin h_{S_1} and destination h_{s_1}). Then, we set the objective function value of the shortest travel time path in the set $\mathcal{P}^{s_1}_{S_1}$ to "zero" and remove this "modified" path from the set $\mathcal{P}^{s_1}_{S_1}$. If $\mathcal{P}^{s_1}_{S_1} = \emptyset$, we continue with the next hubs in the set $\mathcal{H}$ (i.e., with the sets $\mathcal{P}^{s_1}_{S_2}, \mathcal{P}^{s_1}_{S_3}, \ldots, \mathcal{P}^{s_1}_{S_{|\mathcal{H}|}}$) until we modify a path for hub h_{s_1}. If we modify a path for hub h_{s_1}, we assume that the stock level of hub h_{s_1} increases by one rider-bot and the stock level of hub h_{S_i} (i.e., the origin hub of the modified path) decreases by one rider-bot. With the assumed stock levels of the hubs, we update the sets $\mathcal{H}$ and $\underline{\mathcal{H}}$ based on their definitions above. Unless we can find a path with destination h_{s_1} to modify, we remove hub h_{s_1} from set $\underline{\mathcal{H}}$. We repeat this process until $\underline{\mathcal{H}} = \emptyset$.

After modifying the objective function values of the obtained paths during the CG iterations with the procedure above, we apply the CG heuristic to find an integer solution for time interval $t \in T$ as explained in Section 3.2. Then, we update the stock level threshold values of the hubs according to the origins and destinations of the paths in the solution.

In addition, we reposition the rider-bots between the hubs every hour during the operation according to the stock level values. More precisely, with the same strategy above, we send the required number of highest-charged rider-bots from the hubs with a stock level above s_{ht} to the hubs with a stock level below s_{ht} once an hour at the end of the time interval. We start repositioning at time interval t_{36} since we already have enough rider-bots at each hub at the beginning of the operation. Similarly, since it is not useful, we do not reposition the rider-bots at the end of the operation (i.e., after the time interval t_{84}). That is, we reposition the rider-bots at the end of the time interval $t_i \in \{t_{36}, t_{48}, \ldots, t_{84}\}$. Note that the repositioned rider-bots do not serve any order during their travel. The illustration of the implementation of the TP is shown in Fig. 4.

6. Computational study

In this section, we test our algorithm, compare the proposed positioning policies, and discuss the algorithmic and managerial insights that we derive from our numerical experiments. Before presenting our numerical experiments, we explain the procedures used to generate problem instances. Then, we describe our experimental design and implementation details.

6.1. Instance generation

In consultation with our industry partner Trendyol, we focus on a service area comprising three large districts of Istanbul with a high population, many retailers, and an extensive public transport network (shown in Fig. 5). Considering this region, we generate the delivery network and order arrivals as follows.

PDSs: To cut down on purchase and infrastructure costs, many delivery and e-commerce companies prefer to rent locker stations already located in front of/inside popular points of interest, such as post offices, transportation hubs, and supermarkets. Consistent with this strategy, we determine the locations of the PDSs in our network as the locations of the post offices (PTT) and popular supermarket chain stores such as "Migros", "BİM" and "Sarıyer Market" in the delivery region. As a result, we have 130 PDSs in our network, represented by blue circles in Fig. 5.

Bus Stations and Buses: The locations of the bus stations in the delivery region are extracted from Google Maps. The departure hours and lines of the buses traveling between these bus stations are obtained from the official website of the Istanbul Municipality Public Transit Agency (IETT, 2021). We consider 507 bus stations and 75 bus lines in the problem. We note that if the time between two consecutive bus departure times is less than 30 min, we assume that one of these scheduled trips can carry rider-bots. We think that this assumption makes sense in reality because not every bus of the municipality can be equipped to carry rider-bots, at least in the near future.

Hubs: Our preliminary studies showed that the rider-bots with the features stated in Section 6.4 cannot deliver sufficiently enough packages in the delivery region in Fig. 5 if we have less than five rider-bot hubs. Therefore, we set the number of rider-bot hubs to the lowest possible number (i.e., 5). Their locations are also decided by a clustering algorithm (Gaussian Mixture Model), considering the locations of the PDSs.

Order Arrivals: From the historical data, we see that the order arrivals can be closely approximated by a non-homogeneous Poisson process with different arrival rates in each time interval. We apply the following procedure for order origins and destinations since the exact coordinates of the customers and retailers are not shown in the historical data for confidentiality. For the origin, we first determine the neighborhood by randomly choosing a retailer from the pool. Then, the exact location is determined by a randomly drawn coordinate within the neighborhood boundaries of the retailer. We apply the same procedure for the destination (customer location). Finally, we choose the closest PDSs to the origin and destination to determine an order's pick-up and drop-off points. We note that in each time interval in the generated instances, there is a minimum of one order and a maximum of 16 orders.

6.2. Experimental design

To assess the computational performance of the L-FLA and understand the potential benefits of the suggested delivery system under different operational conditions, we conduct four groups of experiments, each focusing on different system dynamics.

Taking the main features of the *Starship* delivery robot produced by Starship Technologies (Starship, 2022) as a reference, we define a *Base Case (BC) setting* as follows.

- The fully charged battery of the rider-bot allows a 120 min travel.
- The charging time of a rider-bot is assumed as 30 min.
- The rider-bots can carry at most two packages.
- The total number of rider-bots is set to 200.
- The delivery time limit, which starts after the package is dropped at the pickup PDS, is two and a half hours.

Building on the BC setting, we define our experiment groups as follows:

- The first group of experiments, E1, aims to investigate how the system performance changes for different numbers of rider-bots to use. Setting the rest of the problem parameters to their BC values, we study the problem instances with $\mathcal{I} = \{100, 150, 250, 300\}$.

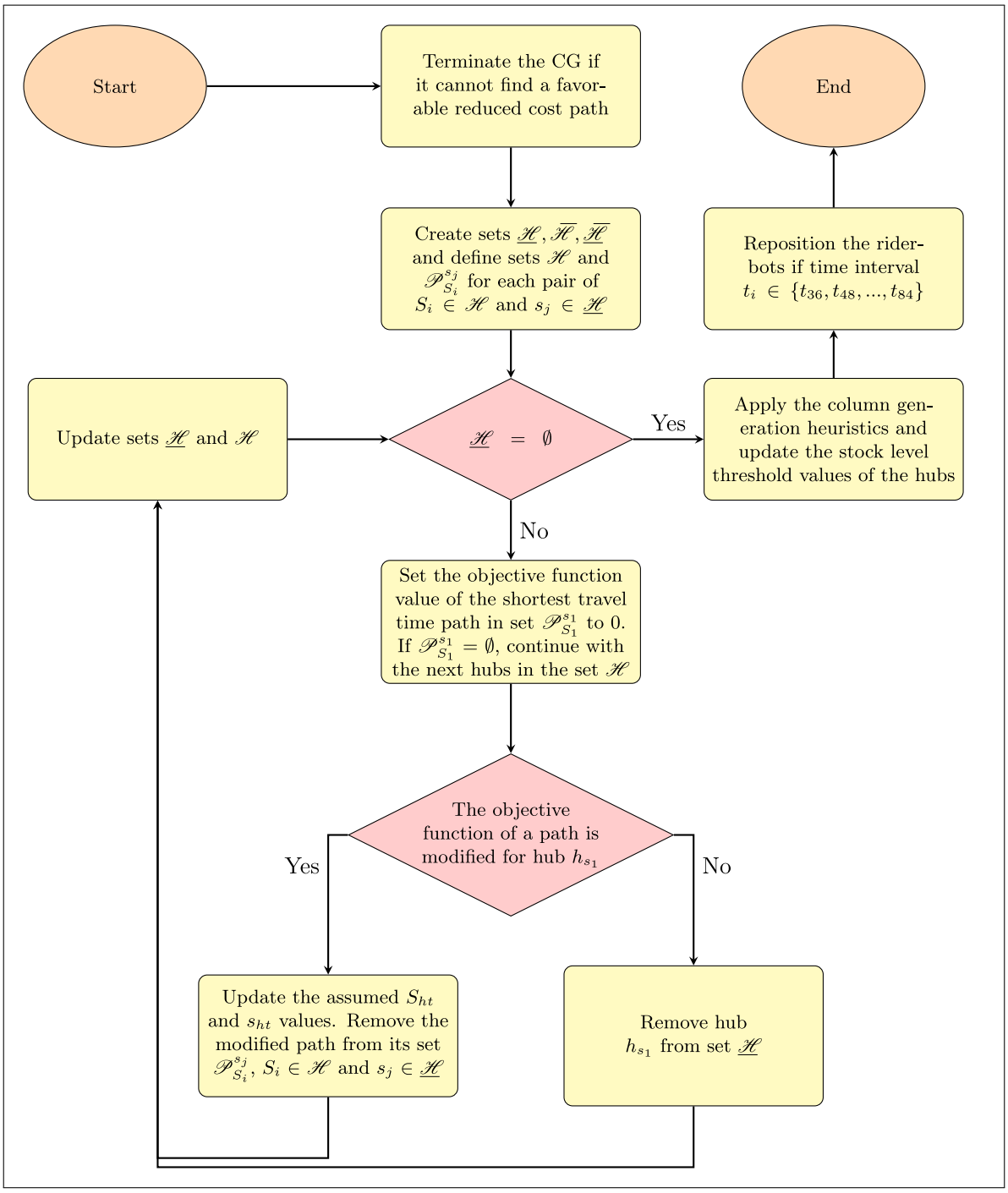


Fig. 4. Overview of the implementation of the two-threshold policy.

- The second group of experiments, E2, aims to investigate how the system performance would change if the rider-bot battery capacities increase with the advances in energy storage technologies. E2 consists of problem instances with rider-bot battery capacity $\mathscr{B} = \{150, 180, 210\}$ minutes.
- The third group of experiments, E3, aims to investigate the impact of the guaranteed delivery time on the delivery rate of the rider-bots. In particular, the maximum delivery times in set $\mathscr{D} = \{30, 60, 90, 120\}$ (minutes) are studied.

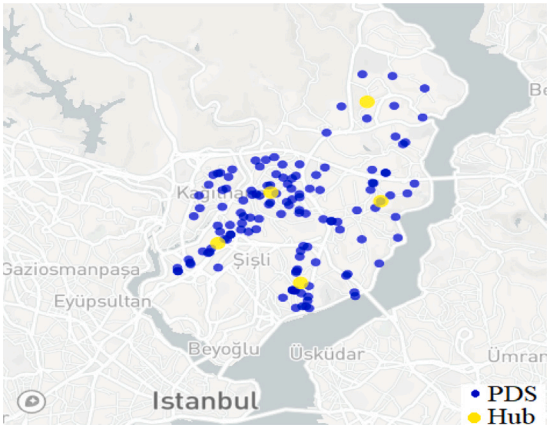


Fig. 5. The delivery region including the hubs and PDSs.

6.3. Implementation details and other parameters

Based on our preliminary experiments, we set the parameters for the L-FLA as in the following:

- We do not apply the learning-based pruning for the partial paths with less than or equal to three nodes since they do not have reliable information to classify them as good or no-good partial paths with a machine learning algorithm. That is, we always keep them in our partial path set for the next iteration of the CG. Let $\mathcal{N}$ be the set of partial paths with more than three nodes.
- The threshold $\mathcal{T}$ is set to 50 and increased by 50 at every iteration of the CG.
- The classification threshold $\mathcal{R}$ is set to 1. Thus, all partial paths in $\mathcal{N}$ are evaluated as no-good partial paths. We choose the first $\mathcal{T}$ no-good partial paths with higher prediction scores in the first iteration of the CG. In later iterations of the CG, we decide on the partial paths to be selected from the set $\mathcal{N}$ by considering the partial paths in the tabu list created with the diversification strategy.
- In the diversification strategy, we add the eliminated partial paths with less than seven nodes and greater than 0.5 prediction scores to the tabu list. Since the partial paths with at least seven nodes have more reliable information for the learning-based pruning (i.e., their prediction scores can be estimated with higher accuracy), we do not consider them in the diversification strategy.
- In the next iteration of the CG, we make sure that we do not remove up to $\mathcal{T}/2$ no-good partial paths with higher prediction scores, which are in the tabu list from the partial path set. Let ζ be the number of these partial paths. After separating ζ partial paths, we choose $\min\{\mathcal{T}-\zeta, |\mathcal{N}|-\zeta\}$ partial paths with higher prediction scores from the set $\mathcal{N}$ for the next extension.

For P_{SP}, the unit travel cost is set to 0.5 units per minute, as done in the study by Mourad et al. (2021). The cost to rent a courier is set to 120 units, based on express courier fees for subscribers of a major on-demand delivery company operating in Istanbul (liderkurye.com.tr). At each iteration of the CG, we add the 30 most negative reduced cost paths to the restricted master formulation. We terminate the CG algorithm when the time limit of five minutes (the length of the decision intervals) is reached, even if it continues to find favorable reduced cost paths.

The packages are assumed to be available in their pickup PDSs 20–25 min after the customers request them. If we cannot assign an order to a rider-bot in four consecutive time intervals (i.e., 20 min), that order is assigned to a courier to complete the transfer in the guaranteed delivery time. We have chosen this threshold as we observed in our preliminary experiments that increasing it can significantly increase the computational effort needed to solve the problem (due to the increased number of orders to consider in the problem) without providing a considerable increase in the rate of orders served by the rider-bots. To make our system applicable in real-world scenarios, we set a maximum waiting time of 5 min for rider-bots at bus stations, considering the priority of passengers and the level of congestion at the bus stations.

For the TP, we used 30 scenarios to compute the S_{ht} and s_{ht}, $h \in H, t \in T$ values, as we observed that larger number of scenarios provides almost the same values.

Our computational experiments were performed on a computer with Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60 GHz 2 processors with a memory of 128 GB. Our algorithm was implemented using Java.

6.4. Analysis

Before analyzing the positioning policies under different conditions, we first test the efficiency of the proposed algorithm (i.e., L-FLA). Then, we discuss the positioning policies with the problem settings presented in Section 6.2.

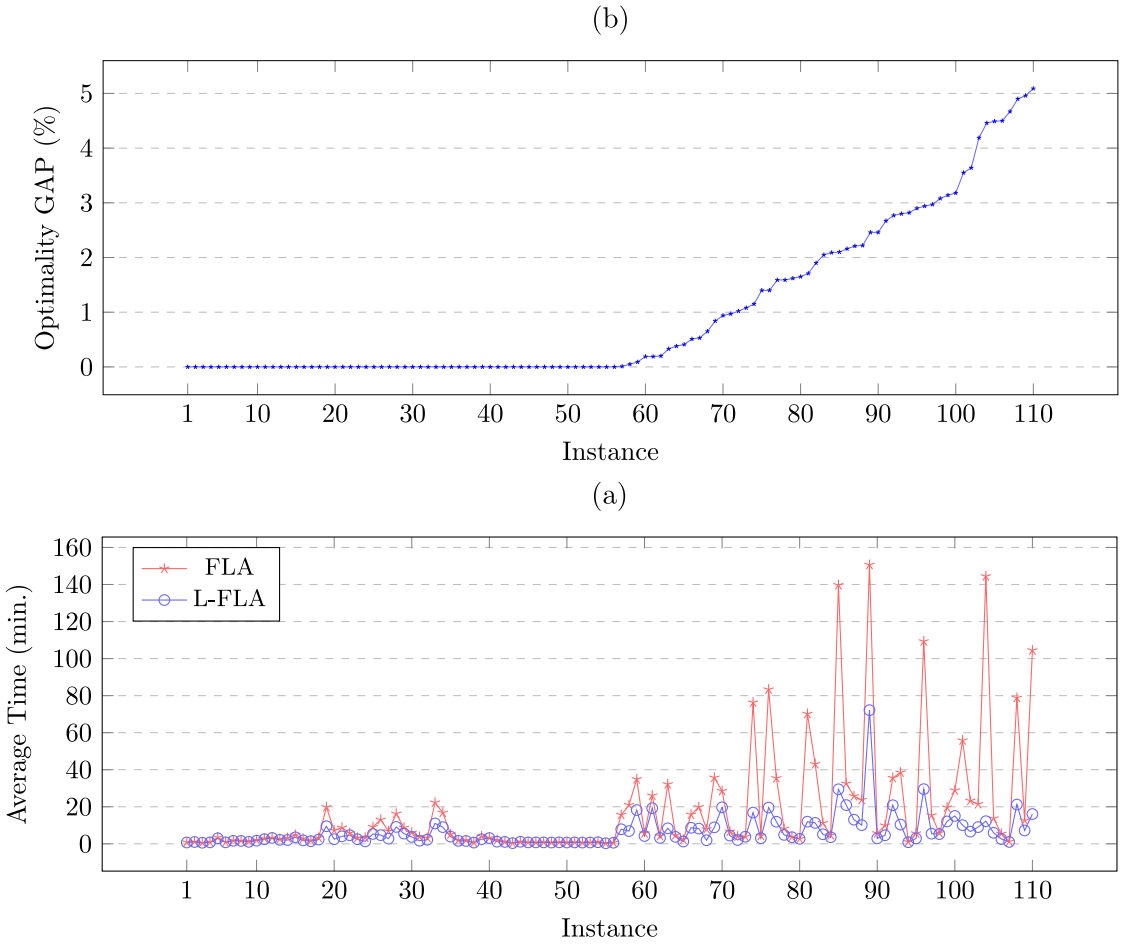


Fig. 6. Performance of the learning-based labeling algorithm without time restriction.

6.4.1. Value of the learning-based labeling algorithm

To assess the computational performance of the suggested learning-based pruning approach, we first run the CG algorithm with and without learning to solve the problem instances (one for each decision epoch) in the BC setting with the nearest hub positioning policy (NHP). To see how much the learning-based pruning would impact the solution quality and run time of the CG algorithm, we did not impose any time limit in these tests, i.e., we did not terminate the CG at the end of the five-minute run time limit. For a fair comparison, to have exactly the same parameters in each problem instance for both CG implementations, we let the system transition to the next decision epoch, following the decisions of the CG without learning. The results are presented in Fig. 6 that shows the optimality gaps incurred due to the learning-based pruning (Fig. 6a) and the run times with and without learning (Fig. 6b), ordering the problem instances by their optimality gaps. As we see in the figure, although the learning-based pruning reduces the average run time by more than 80%, the maximum optimality gap incurred in these tests is less than 5.09% for all problem instances (1.04% on the average).

In practice, to use the CG algorithm in the rolling horizon approach we consider, one must impose a strict time limit to solve the problem instances in each time epoch. In Table 3, we compare the quality of the solutions we get from the CG algorithm with and without learning-based pruning. These are presented in the columns L-FLA and FLA, respectively. The row "ODR (%)" indicates the percentage of orders delivered by rider-bots, and the row "Bus Usage (%)" shows what percentage of rider-bot routes use the public transportation network. The total cost of the delivery operation is shown in the "Cost" row. Finally, the "Time (min.)" row shows the average run time of the CG algorithm to converge if we do not impose the time limit.

As we can clearly see in Table 3, the faster convergence of the CG with the learning-based pruning has a drastic impact on the solution quality for the solutions that can be found within the five-minute interval in which the assignment decisions have to be made. Remarkably, the solutions with learning are 22% less costly, make 10.8% more deliveries with rider-bots, and increase the utilization of public transit lines by 9.95%.

Table 3
Value of the learning-based labeling algorithm.

	FLA	L-FLA
ODR (%)	64.47	85.27
Bus Usage (%)	46.04	55.99
Cost	70,801.34	55,276.75
Time (min.)	17.69	6.45

Table 4
Performance of learning-based labeling algorithm on different networks.

	FLA	L-FLA
ODR (%)	84.59	84.36
Bus Usage (%)	46.31	44.49
Cost	56,055.75	56,272.52
Time (min.)	7.28	3.18

6.4.2. Addressing stochasticity in public transportation trips

In this section, we analyze the generalizability of our learning-based solution approach to accommodate unexpected changes in the public transit network without retraining the learning model with the updated network structure. More precisely, we randomly remove some public transit trips for the robots' which may happen in practice due to several reasons, such as (1) the limited bus capacities, which are prioritized for passengers, (2) occasional cancellations of trips due to vehicle breakdowns/accidents, and (3) change of time tables due to the unusual traffic conditions. Then, we solve the problem on the dynamically updated networks with the available information at the time of decision, using the same trained learning model.

In Table 4, we show the performance of the L-FLA on the dynamically updated networks. Note that, as in the experiments in the previous subsection, we relax the time limit in these tests to make sure that the classical CG algorithm converges and provides a strong benchmark for comparing the quality of solutions with L-FLA. As seen in the table, the obtained results support the findings in the previous subsection. On average, L-FLA can solve each problem instance in three minutes, reducing the solution time by more than 50% compared to FLA, with only less than a 0.4% increase in the total cost.

6.4.3. Impact of the number of rider-bots

The number of rider-bots in the system and their positioning during the delivery operations are critical factors determining the rider-bot availability and, hence, the system performance. In this section, we analyze the results of E1 experiments to see how system performance changes with the number of rider-bots and positioning policies.

We present the results of the E1 experiments in Table 5 that shows the percentage of orders that are delivered by rider-bots (ODR), the percentage of rider-bot routes that use the public transportation (Bus Usage), and the resulting total cost (Cost), including the payments to the dedicated couriers in case the rider-bots cannot be used for the delivery. As expected, we see that the rider-bots deliver more packages, and as a result, the total delivery cost decreases as their number increases. However, we also see that increasing the number of rider-bots beyond 250 does not significantly improve the system performance, whereas reducing the number of rider-bots below 150 would drastically impact the percentage of the packages the rider-bots can deliver. As such, one can see the importance of having the right number of rider-bots to maximize the benefits.

6.4.4. Impact of higher battery charge capacities

The battery capacity is another important problem parameter affecting rider-bots' availability. Table 6 presents the results of the E2 experiments investigating whether increasing the battery capacities would significantly improve the system performance. As an interesting result, we see in the table that increasing battery capacities does not improve the system performance as much. For example, with a 50% percent increase in the full charge operation capacity (i.e., an increase from 120 to 180-minute capacity), the total cost is only reduced by 5%. Comparing these results with those in Table 5, one can infer that acquiring more delivery robots can provide more benefits than spending more on higher capacity batteries to have full charge operation capacities beyond 120 min. One important reason for such a result is the use of public transit vehicles for parts of the rider-bot journeys, which reduces the need for large battery sizes to use them effectively.

6.4.5. Impact of the guaranteed delivery time

With the E3 experiment group, we try to reveal the trade-off between the service quality (guaranteed delivery lead times) and the delivery cost. Summarizing our results for these experiments, Fig. 7 shows how the available time impacts the delivery rate by rider-bots, which has a very strong negative correlation with the overall costs, as discussed in the previous subsections. As an interesting result, the figure shows that the rider-bots can be efficiently used even for very tight delivery time promises, such as 90 min, with the problem parameters we consider in our study. Since we cover a large area in our study, this remarkable result shows that by integrating public transit vehicles into rider-bot deliveries, the proposed system can provide high levels of customer satisfaction for much smaller economic and environmental costs, as discussed in the following.

Table 5
Impact of the number of rider-bots.

# of Rider-bots	Policy	ODR (%)	Bus Usage (%)	Cost
100	**NHP**	56.57	57.81	80,281.61
	SHP	51.42	48.38	86,206.60
	TP	56.62	58.81	80,253.58
150	**NHP**	73.55	58.10	65,728.04
	SHP	66.04	48.56	74,245.55
	TP	73.52	58.42	65,749.06
200	**NHP**	85.27	55.99	55,276.75
	SHP	74.30	48.02	67,055.42
	TP	85.19	57.01	55,320.29
250	**NHP**	89.13	55.64	51,656.87
	SHP	76.71	48.16	64,848.58
	TP	89.04	55.17	51,797.37
300	**NHP**	89.89	55.55	50,668.16
	SHP	77.73	47.98	63,780.87
	TP	89.79	55.49	50,697.66

Table 6
Impact of higher battery charge capacities.

Full charge capacity (min)	Policy	ODR (%)	Bus Usage (%)	Cost
120	**NHP**	73.55	58.10	65,728.04
	SHP	66.04	48.56	74,245.55
	TP	73.52	58.42	65,749.06
150	**NHP**	74.74	54.58	64,311.65
	SHP	68.74	44.40	72,335.13
	TP	74.49	55.10	64 479.20
180	**NHP**	76.25	51.45	62,536.73
	SHP	69.84	42.16	71,633.65
	TP	76.20	50.95	62,673.07
210	**NHP**	77.62	50.54	61,105.85
	SHP	70.88	41.56	70,789.30
	TP	77.55	50.22	61,062.50

6.4.6. Environmental and social benefits of the proposed delivery system

To analyze the potential environmental and social benefits of the suggested system, we compare the vehicle kilometers and noise and emissions levels it is responsible for with the current system using dedicated couriers to serve express delivery requests.

Due to cost advantages and the small size of a typical e-commerce parcel, on-demand express delivery providers prefer to use motorcycles for their courier services. Considering all the packages included in our demand data are smaller than 12 deciliters (shoe box size), we considered express courier services with motorcycles as a benchmark. In Table 7, we compare the vehicle kilometers (a proxy measure for traffic congestion and accidents), noise, and emissions levels for the proposed system with two alternative cases with express deliveries with motorcycle couriers for the BC setting. Representing the current practice more closely, the first alternative (i.e., Case 1) assumes each courier transfers a single online order in a trip. In the second alternative (i.e., Case 2), we consider the case where the couriers transfer more than one order in their trips. To calculate the reported metrics for Case 2, we use the results from Hoang et al. (2019), Lechner et al. (2020), and Comi and Savchenko (2021). As we see in Table 7, the suggested system can reduce the negative externalities of package transfers by more than 80% in our BC setting. This reduction in the negative externalities is tied to the rate of deliveries completed by the rider-bots, which are almost emission-free and do not create extra vehicle traffic. It is seen that the suggested system can provide significant advantages from environmental and social perspectives as long as a significant percentage of the deliveries are completed by the rider-bots, which turned out to be the case for almost all the problem instances we studied.

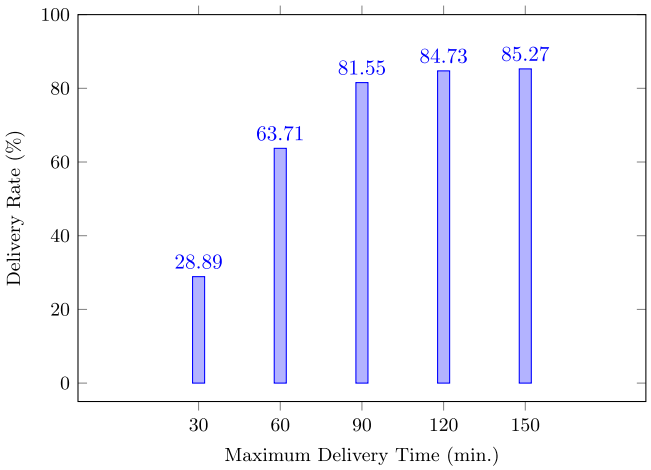


Fig. 7. Delivery rates of rider-bots at different delivery times.

Table 7
Comparison of environmental impacts.

	Units	Case 1			Case 2		
		Current system	Proposed system	Reduction	Current system	Proposed system	Reduction
Distance.estimate	km/day	7.7K	1.1K	85.71%	4K	605	84.88%
Noise.estimate	dB/day	52K	7.8K	85.00%	7.8K	1.1K	85.90%
CO_2.*estimate*	g/day	274K	41K	85.00%	142K	21K	85.21%
CO.*estimate*	g/day	32K	4.8K	85.00%	16.5K	2.4K	85.45%
NO_x.*estimate*	g/day	2.2K	334	84.81%	1.1K	173	84.27%
HC.*estimate*	g/day	2.9K	440	84.82%	1.5K	227	84.87%

7. Conclusion

In this study, we investigated a novel last-mile delivery system employing rider-bots using public transport networks to extend their delivery reach. We adopted a rolling horizon optimization method for real-time control of a dynamic system with random order arrivals throughout the day. We determined rider-bot assignments to the orders and optimized their routes to reduce overall delivery costs. Given the complex nature of the problem, traditional solution methodologies fall short. Hence, we introduced a novel learning-based labeling algorithm (L-FLA) integrated into a CG approach to find cost-efficient paths for rider-bots. This innovative algorithm allows us to solve large-scale problems, providing high-quality solutions quickly. This way, re-optimization can be achieved frequently. Once a delivery robot finishes its task, choosing the right hub for its next destination is crucial as it can impact the cost of delivering upcoming orders. We introduced and evaluated three placement strategies determining the hub where the robot will go after completing its assigned deliveries, highlighting their advantages and disadvantages.

To test the computational performance of our solution methodology and obtain managerial insights into the potential benefits of the suggested delivery system, we conducted numerical tests with data from an e-commerce company. The findings highlight the importance of the learning phase in the proposed CG approach, showing that the proposed algorithm yields significantly higher-quality solutions compared to the CG approach without the learning phase. Specifically, the delivery rate of the rider-bots increases from approximately 64% up to 85% with the learning phase. As important managerial insights, our results show that with the help of public transit vehicles, rider-bots can complete a significant number of express shipments in a large geographical area without needing large battery capacities and can provide significant environmental and social benefits.

In this study, our focus was on the operational problem, assuming that strategic decisions like hub locations, rider-bot fleet sizes, and their allocation to hubs at the start of the day have already been made. Future studies can explore methods to optimize system configuration. Another promising research avenue is examining multi-echelon delivery networks with transshipments, integrating rider-bots with other vehicles to expand coverage and delivery capabilities.

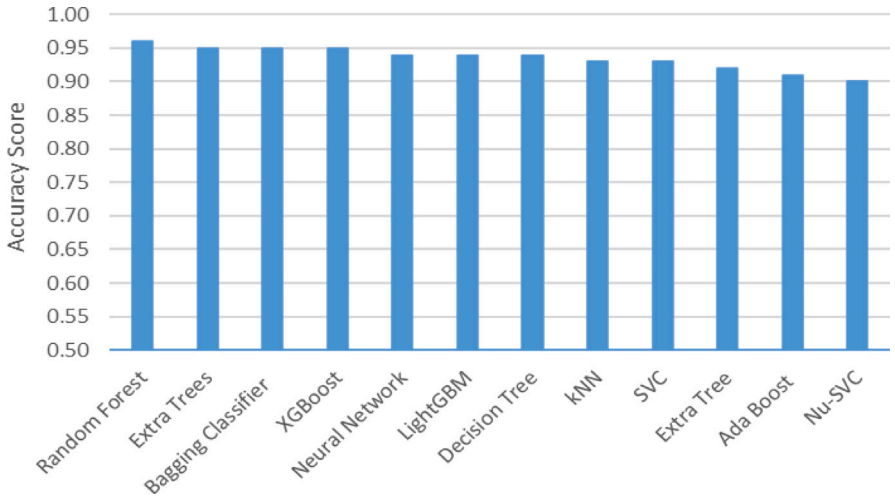


Fig. A.8. Comparison of the machine learning algorithms.

CRediT authorship contribution statement

Umut Ermağan: Writing – original draft, Visualization, Validation, Software, Methodology, Data curation, Conceptualization. **Barış Yıldız:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization. **F. Sibel Salman:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study has been supported by the European Research Council (ERC) under grant number 101076231 as a part of the GoodMobility project (https://goodmobility.ku.edu.tr). This work would not have been possible without the assistance of the Trendyol Operations and Innovation Department. We thank them for providing us with real-life data and for informative discussions.

Appendix A. Comparison of the machine learning algorithms

In Fig. A.8, we present a performance comparison of the trained machine learning algorithms based on accuracy scores obtained via the Lazypredict package (Pandala, 2023). Note that the figure only includes machine algorithms that achieve 90% or a higher accuracy score among the 30 classification algorithms implemented in the package.

Appendix B. Base case setting solutions

In Tables B.8, B.9, B.10 and B.11, the detailed results of our experiments with the BC setting are reported. In the tables, "Total Order" represents the total daily orders delivered. "ODR (%)" shows what percentage of the orders in a day are delivered by rider-bots. In the "Bus Usage (%)" column, we show what percentage of rider-bot routes use the public transportation network. The total delivery cost is shown in the "Cost" column. Fig. B.9 demonstrates the percentage of cost reduction achieved through the implementation of learning-based pruning by comparing the cost values in Tables B.8 and B.9.

Table B.8
Solutions of the NHP-w/o-learning (Base Case).

Instance	Total Order	ODR (%)	Bus Usage (%)	Cost
Day1	1047	67.53	43.98	65,111.93
Day2	1113	63.61	46.98	71,514.12
Day3	1145	59.74	45.71	77,760.38
Day4	1056	71.69	46.54	61,402.12
Day5	1098	67.30	46.75	68,518.72
Day6	1149	62.05	47.22	74,988.47
Day7	1124	62.19	44.50	72,976.30
Day8	1080	63.70	46.12	69,822.38
Day9	1110	61.44	46.91	72,780.83
Day10	1140	62.54	40.10	72,995.22
Day11	1124	68.33	53.43	68,694.16
Day12	1173	62.40	43.98	75,752.27
Day13	1097	63.72	43.63	70,807.53
Day14	1119	61.93	48.09	73,343.91
Day15	1097	68.82	46.73	65,551.74
Avg.	**1111**	**64.47**	**46.04**	**70,801.34**

Table B.9
Solutions of the NHP (Base Case).

Instance	Total Order	ODR (%)	Bus Usage (%)	Cost
Day1	1047	84.34	54.77	54,170.11
Day2	1113	84.19	54.86	56,206.91
Day3	1145	83.06	55.32	58,853.53
Day4	1056	87.31	57.1	51,756.54
Day5	1098	84.97	57.12	55,204.73
Day6	1149	85.90	56.82	56,467.11
Day7	1124	85.14	57.23	55,028.11
Day8	1080	84.44	55.36	55,411.00
Day9	1110	86.13	54.57	53,620.18
Day10	1140	85.53	53.9	54,902.07
Day11	1124	86.48	56.33	54,630.47
Day12	1173	84.74	55.86	58,890.29
Day13	1097	88.06	58.00	52,875.97
Day14	1119	83.20	57.00	56,706.88
Day15	1097	85.51	55.67	54,427.33
Avg.	**1111**	**85.27**	**55.99**	**55,276.75**

Table B.10
Solutions of the SHP (Base Case).

Instance	Total Order	ODR (%)	Bus Usage (%)	Cost
Day1	1047	72.97	45.17	64,488.38
Day2	1113	74.66	47.15	66,502.90
Day3	1145	73.80	50.40	69,494.93
Day4	1056	73.67	45.25	64,690.43
Day5	1098	74.77	47.25	65,847.66
Day6	1149	75.54	45.57	68,137.48
Day7	1124	75.44	49.77	66,939.08
Day8	1080	73.8	49.92	66,092.98
Day9	1110	75.14	48.79	66,197.27
Day10	1140	75.70	46.76	65,865.30
Day11	1124	72.69	49.01	68,227.27
Day12	1173	74.42	47.8	70,871.77
Day13	1097	72.74	47.55	68,410.61
Day14	1119	73.91	49.51	67,926.70
Day15	1097	75.21	50.47	66,138.56
Avg.	**1111**	**74.30**	**48.02**	**67,055.42**

Table B.11
Solutions of the TP (Base Case).

Instance	Total order	ODR (%)	Bus Usage (%)	Cost
Day1	1047	84.24	54.25	53,904.02
Day2	1113	83.83	56.73	56,891.89
Day3	1145	83.32	57.85	58,566.95
Day4	1056	86.74	56.56	52,384.26
Day5	1098	85.25	58.37	54,604.10
Day6	1149	85.55	56.68	56,617.90
Day7	1124	83.54	58.08	57,032.26
Day8	1080	85.83	55.36	53,627.57
Day9	1110	86.22	55.38	53,476.37
Day10	1140	85.96	57.29	54,623.59
Day11	1124	85.94	57.45	54,874.98
Day12	1173	84.40	58.63	59,568.95
Day13	1097	87.88	58.33	52,979.74
Day14	1119	83.38	58.67	56,563.63
Day15	1097	85.78	55.54	54,088.17
Avg.	**1111**	**85.19**	**57.01**	**55,320.29**

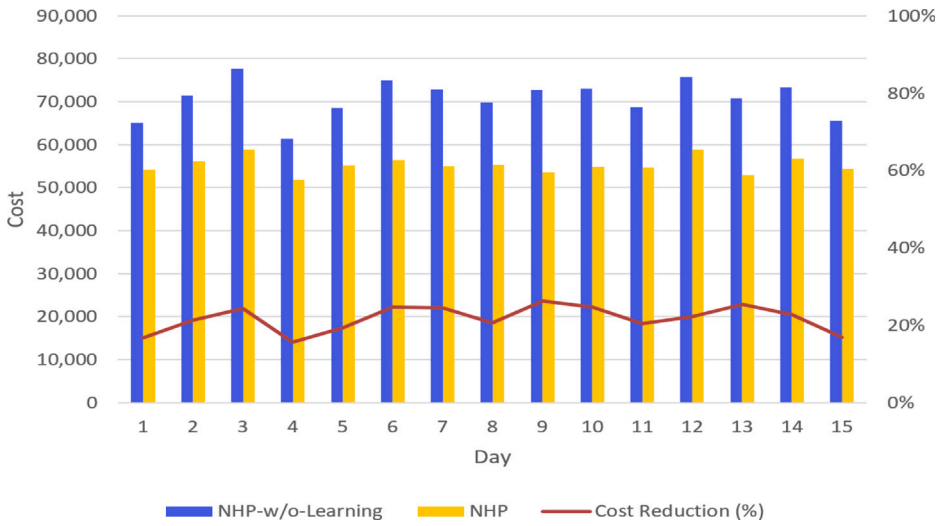


Fig. B.9. Cost comparison.

References

Alfandari, L., Ljubić, I., da Silva, M.D.M., 2022. A tailored Benders decomposition approach for last-mile delivery with autonomous robots. European J. Oper. Res. 299 (2), 510–525.

Alverhed, E., Hellgren, S., Isaksson, H., Olsson, L., Palmqvist, H., Flodén, J., 2024. Autonomous last-mile delivery robots: a literature review. Eur. Transp. Res. Rev. 16 (1), 4.

Anika, T., Dan, L., Kaisar, E.I., 2021. An optimization model for determining optimal fleet size for a robot-sharing system. In: International Conference on Transportation and Development 2021. pp. 450–467.

Bakach, I., Campbell, A.M., Ehmke, J.F., 2020. A two-tier urban delivery network with robot-based deliveries. Working Paper Series.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H., 1998. Branch-and-price: Column generation for solving huge integer programs. Oper. Res. 46 (3), 316–329.

Barron, Y., 2019. A state-dependent perishability (s, S) inventory model with random batch demands. Ann. Oper. Res. 280 (1), 65–98.

Behnke, M., 2019. Recent trends in last mile delivery: Impacts of fast fulfillment, parcel lockers, electric or autonomous vehicles, and more. In: Logistics Management. Springer, pp. 141–156.

Behnke, M., Kirschstein, T., Bierwirth, C., 2021. A column generation approach for an emission-oriented vehicle routing problem on a multigraph. European J. Oper. Res. 288 (3), 794–809.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. European J. Oper. Res. 202 (1), 8–15.

Boysen, N., Schwerdfeger, S., Weidinger, F., 2018. Scheduling last-mile deliveries with truck-based autonomous robots. European J. Oper. Res. 271 (3), 1085–1099.

Chan, C.-Y., 2017. Advancements, prospects, and impacts of automated driving systems. Int. J. Transp. Sci. Technol. 6 (3), 208–216.

Chen, C., Demir, E., Huang, Y., 2021a. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. European J. Oper. Res. 294 (3), 1164–1180.

Chen, C., Demir, E., Huang, Y., Qiu, R., 2021b. The adoption of self-driving delivery robots in last mile logistics. Transp. Res. E 146, 102214.

Chi, C., Aboussalah, A.M., Khalil, E.B., Wang, J., Sherkat-Masoumi, Z., 2022. A deep reinforcement learning framework for column generation. arXiv preprint arXiv:2206.02568.

Comi, A., Savchenko, L., 2021. Last-mile delivering: Analysis of environment-friendly transport. Sustainable Cities Soc. 74, 103213.
Cordeau, J.-F., Laporte, G., Ropke, S., 2008. Recent models and algorithms for one-to-one pickup and delivery problems. In: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 327–357.
Desaulniers, G., Desrosiers, J., Solomon, M.M., 2006. Column generation, vol. 5, Springer Science & Business Media.
Desrosiers, J., Soumis, F., Desrochers, M., 1984. Routing with time windows by column generation. Networks 14 (4), 545–565.
Deutsch, Y., Golany, B., 2018. A parcel locker network as a solution to the logistics last mile problem. Int. J. Prod. Res. 56 (1–2), 251–261.
DHL, 2022. What are DHL POP. https://bit.ly/3ZYaPpw. (Last Accessed 1 February 2023).
Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European J. Oper. Res. 54 (1), 7–22.
Edwards, J., McKinnon, A., Cherrett, T., McLeod, F., Song, L., 2009. The impact of failed home deliveries on carbon emissions: Are collection/delivery points environmentally-friendly alternatives. In: 14th Annual Logistics Research Network Conference. p. M117.
Engesser, V., Rombaut, E., Vanhaverbeke, L., Lebeau, P., 2023. Autonomous delivery solutions for last-mile logistics operations: a literature review and research agenda. Sustainability 15 (3), 2774.
Ermağan, U., Yıldız, B., Salman, F.S., 2022. A learning based algorithm for drone routing. Comput. Oper. Res. 137, 105524.
FAA, 2020. Fact sheet – Small unmanned aircraft systems (UAS) regulations (Part 107). https://bit.ly/3ZYaPpw. (Last Accessed 1 June 2022).
FedEx, 2020. Fedex corporation 2020 annual report. https://bit.ly/3Jhi4mv. (Last Accessed 2 June 2021).
Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., Soumis, F., 2019. Column generation for vehicle routing problems with multiple synchronization constraints. European J. Oper. Res. 272 (2), 699–711.
Freeman, R.J., 1957. Ss inventory policy with variable delivery time. Manage. Sci. 3 (4), 431–434.
Gribkovskaia, I., Laporte, G., 2008. One-to-many-to-one single vehicle pickup and delivery problems. In: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 359–377.
Gschwind, T., Irnich, S., Rothenbächer, A.-K., Tilk, C., 2018. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European J. Oper. Res. 266 (2), 521–530.
Hall, M.A., 1999. Correlation-based feature selection for machine learning (Ph.D. thesis). The University of Waikato.
He, L., Liu, S., Shen, Z.-J.M., 2022. Smart urban transport and logistics: A business analytics perspective. Prod. Oper. Manage..
Hernández-Pérez, H., Salazar-González, J.-J., 2007. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. Netw. Int. J. 50 (4), 258–272.
Hoang, A.T., Tran, Q.V., Al-Tawaha, A.R.M.S., Nguyen, X.P., et al., 2019. Comparative analysis on performance and emission characteristics of an in-Vietnam popular 4-stroke motorcycle engine running on biogasoline and mineral gasoline. Renew. Energy Focus 28, 47–55.
Hoffmann, T., Prause, G., 2018. On the regulatory framework for last-mile delivery robots. Machines 6 (3), 33.
Hottung, A., Tierney, K., 2019. Neural large neighborhood search for the capacitated vehicle routing problem. arXiv preprint arXiv:1911.09539.
Huang, Y., Savelsbergh, M., Zhao, L., 2018. Designing logistics systems for home delivery in densely populated urban areas. Transp. Res. B 115, 95–125.
IETT, 2021. The website including the information of the bus lines in Istanbul. https://iett.istanbul/.
Joerss, M., Schröder, J., Neuhaus, F., Klink, C., Mann, F., 2016. Parcel delivery: The future of last mile delivery. https://bit.ly/3TDX9ih. (Last Accessed 19 March 2024).
Kızıl, K.U., Yıldız, B., 2023. Public transport-based crowd-shipping with backup transfers. Transp. Sci. 57 (1), 174–196.
Lechner, C., Schnaiter, D., Siebert, U., Böse-O'Reilly, S., 2020. Effects of motorcycle noise on annoyance—A cross-sectional study in the Alps. Int. J. Env. Res. Public Health 17 (5), 1580.
Lin, Y.H., Wang, Y., He, D., Lee, L.H., 2020a. Last-mile delivery: Optimal locker location under multinomial logit choice model. Transp. Res. E 142, 102059.
Lin, Y.H., Wang, Y., Lee, L.H., 2020b. Parcel locker location problem under threshold luce model. arXiv preprint arXiv:2002.10810.
Lozano, L., Duque, D., Medaglia, A.L., 2016. An exact algorithm for the elementary shortest path problem with resource constraints. Transp. Sci. 50 (1), 348–357.
Morabit, M., Desaulniers, G., Lodi, A., 2021. Machine-learning–based column selection for column generation. Transp. Sci. 55 (4), 815–831.
Morabit, M., Desaulniers, G., Lodi, A., 2022. Machine-learning-based arc selection for constrained shortest path problems in column generation. INFORMS J. Optim. 5 (2), 191–210.
Mourad, A., Puchinger, J., Van Woensel, T., 2021. Integrating autonomous delivery service into a passenger transportation system. Int. J. Prod. Res. 59 (7), 2116–2139.
Nair, V., Bartunov, S., Gimeno, F., Von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., et al., 2020. Solving mixed integer programs using neural networks. arXiv preprint arXiv:2012.13349.
Olejarz, D.A., 2020. An Assessment of the Use of Autonomous Ground Vehicles for Last-Mile Parcel Delivery (Ph.D. thesis). University of Toronto.
Ostermeier, M., Heimfarth, A., Hübner, A., 2022. Cost-optimal truck-and-robot routing for last-mile delivery. Networks 79 (3), 364–389.
Pandala, S.R., 2023. Lazy predict. MIT License http://bit.ly/47ekCtP. (Last Accessed 20 December 2023).
Poeting, M., Schaudt, S., Clausen, U., 2019. A comprehensive case study in last-mile delivery concepts for parcel robots. In: 2019 Winter Simulation Conference. WSC, IEEE, pp. 1779–1788.
Powell, W.B., 2019. A unified framework for stochastic optimization. European J. Oper. Res. 275 (3), 795–821.
Ribeiro, C.C., Soumis, F., 1994. A column generation approach to the multiple-depot vehicle scheduling problem. Oper. Res. 42 (1), 41–52.
Roboticsbiz, 2021. Autonomous delivery robots – Benefits and drawbacks. https://bit.ly/3YIF56x. (Last Accessed 9 February 2023).
Rohmer, S., Gendron, B., 2020. A Guide to Parcel Lockers in Last Mile Distribution: Highlighting Challenges and Opportunities from an OR Perspective, (No.11), Cirrelt, Montreal.
Ropke, S., Cordeau, J.-F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. 43 (3), 267–286.
Sazli, M.H., 2006. A Brief Review of Feed-Forward Neural Networks, vol. 50, (01), Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering.
Scikit-Learn, 2022. Cross-validated grid search. Python https://bit.ly/2QBg3Hy. (Last Accessed 16 July 2022).
Shen, Y., Sun, Y., Li, X., Eberhard, A., Ernst, A., 2022. Enhancing column generation by a machine-learning-based pricing heuristic for graph coloring. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 9. pp. 9926–9934.
Simoni, M.D., Kutanoglu, E., Claudel, C.G., 2020. Optimization and analysis of a robot-assisted last mile delivery system. Transp. Res. E 142, 102049.
Sonneberg, M.-O., Leyerer, M., Kleinschmidt, A., Knigge, F., Breitner, M.H., 2019. Autonomous unmanned ground vehicles for urban logistics: Optimization of last mile delivery operations. In: Proceedings of the 52nd Hawaii International Conference on System Sciences.
Srinivas, S., Ramachandiran, S., Rajendran, S., 2022. Autonomous robot-driven deliveries: A review of recent developments and future directions. Transp. Res. E 165, 102834.
Starship, 2022. Starship delivery robot. https://bit.ly/2xV94ND. (Last Accessed 13 September 2022).
Svozil, D., Kvasnicka, V., Pospichal, J., 1997. Introduction to multi-layer feed-forward neural networks. Chemometr. Intell. Lab. Syst. 39 (1), 43–62.
Ulmer, M.W., Streng, S., 2019. Same-day delivery with pickup stations and autonomous vehicles. Comput. Oper. Res. 108, 1–19.
Wang, Y., 2022. JD.com utilizes robots in Shanghai for contactless delivery. https://bit.ly/3gE602Y. (Last Accessed 2 October 2022).
Yu, S., Campbell, A.M., Ehmke, J.F., Puchinger, J., 2023. Last-mile robot deliveries with access restrictions. In: ROADEF 2023: 24ème congrès de la Société Française de Recherche opérationnelle et d'Aide à la Décision.
Yu, S., Puchinger, J., Sun, S., 2020. Two-echelon urban deliveries using autonomous vehicles. Transp. Res. E 141, 102018.