Transportation Research Part E 216 (2026) 105170

Contents lists available at ScienceDirect

Transportation Research Part E

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

Deep reinforcement learning for optimizing mobile depots allocation and hybrid order dispatch in last-mile delivery under multi-participants behavioral uncertainty

Shixuan Hou[a,*], Chengming Hu[b], Annabathuni Sandeep Chowdary[c], Bissan Ghaddar[a], Joe Naoum-Sawaya[a], Jie Gao[d]

[a] Ivey Business school, Western University, and IE University, 1255 Western Rd, London, N6G 0N1, Ontario, Canada
[b] School of Computer Science, McGill University, 3480 University Street, Montréal, H3A 0E9, Quebec, Canada
[c] Bell Canada, 160 Elgin Street, Ottawa, K2P 2C4, Ontario, Canada
[d] Department of Transport & Planning, Delft University of Technology, Delft, 2628 CD, South Holland, Netherlands

ARTICLE INFO

Keywords:
Transportation and logistics
Mixed integer programming
Behavior uncertainty
Reinforcement learning

ABSTRACT

As last-mile delivery demand continues to grow and customer preferences become increasingly diverse, traditional single-mode logistics systems are inadequate in addressing the complexity and variability of modern urban delivery environments. In this paper, we propose an innovative hybrid last-mile delivery system that leverages mobile depots as intermediate hubs, integrating crowd-shipping, and customer self-pickup, to enhance cost-effectiveness. However multiple stakeholders in the system have conflicting interests, ignoring their behavioral uncertainties can impair operational efficiency and increase delivery costs. For example, customers and crowd-shippers may compete for the limited storage capacity of mobile depots, and crowd-shippers may be unwilling to deliver orders that customers decline to pick up. Specifically, the study captures crowd-shippers' order-acceptance behavior and customers' self-pickup choice through two binomial logit models, and embeds these behavioral components into a stochastic mixed-integer programming model. This model jointly optimizes mobile depots allocation and order dispatch decisions, aiming to minimize the expected total operational cost. Considering computational complexity of this model, we propose PRiME-DQN, a reinforcement learning framework that incorporates behavior-prioritized resource filtering, action biasing, and model-efficient temperature tuning to accelerate convergence and improve solution quality. A case study on Amazon's last-mile delivery in Seattle shows that the proposed hybrid model achieves up to 49.5% cost reduction in low-shipper settings and 11.3% in higher-shipper settings compared to a crowd-shipping system, while also consistently surpassing the self-pickup baseline by 3–6%. Extensive scaling experiments show that PRiME-DQN produces high-quality feasible solutions efficiently. For the largest instances, it achieves delivery costs 1.97% lower than Gurobi's best incumbent solution obtained within the 3,600-second time limit, while reducing the solution time to approximately 1400 seconds.

* Corresponding author.
E-mail addresses: shou@ivey.ca (S. Hou), chengming.hu@mail.mcgill.ca (C. Hu), avgchowdary@gmail.com (A.S. Chowdary), bghaddar@ivey.ca (B. Ghaddar), joe.naoum@ie.edu (J. Naoum-Sawaya), j.gao-1@tudelft.nl (J. Gao).

https://doi.org/10.1016/j.tre.2026.105170
Received 28 April 2026; Received in revised form 5 July 2026; Accepted 12 August 2026
Available online 24 August 2026

1. Introduction

Global retail e-commerce sales are projected to reach approximately $6.88 trillion in 2026, accounting for about 21.1% of total worldwide retail sales, and are expected to approach $7.89 trillion by 2028,[1] a surge that is putting unprecedented strain on urban logistics networks. Within this system, last-mile delivery, defined as the final leg from distribution hubs to end consumers, accounts for 41% to 53%[2] of total fulfillment costs and is widely recognized as a leading source of urban congestion and carbon emissions (Joerss et al., 2016; Macrina et al., 2020). Improving efficiency in this segment is crucial for developing a sustainable and effective urban logistics infrastructure. However, conventional logistics frameworks, characterized by fixed distribution points and predetermined routing, lack the flexibility necessary to accommodate evolving demands, such as heterogeneous customer preferences and uncertainties in delivery personnel behaviors (Mohri et al., 2023). These rigid systems often result in redundant deliveries, low customer satisfaction, and inefficient resource utilization, clearly highlighting the need for innovation (Moroz and Polkowski, 2016; Deutsch and Golany, 2018).

In practice, leading e-commerce and logistics firms have already adopted many innovative order delivery modes, such as crowd-shipping (e.g., Amazon Flex[3]), staffed or autonomous pickup points such as DHL Packstations[4] and 7-Eleven Parcel Lockers,[5] and pilot deployments of unmanned delivery devices ranging from Starship robots [6] to JD.com's sidewalk bots.[7] The academic community has also explored methods for the efficient operation of such models. These include crowd-shipping integrated with parcel lockers to extend delivery reach and reduce courier workload (Ghaderi et al., 2022; Zhang et al., 2023a; Stokkink and Geroliminis, 2023), self-service pickup via stationary or mobile lockers to enhance customer convenience and reduce failed delivery attempts (Wen and Li, 2016; Yuen et al., 2018; Peppel et al., 2024), and unmanned delivery technologies such as autonomous robots and drones to reduce labor dependency (Sajid et al., 2022; Sonneberg et al., 2019). While each of these approaches demonstrates clear operational potential, the majority of existing applications and models treat them as stand-alone, single-mode solutions. However, in light of growing market volatility and increasingly diverse customer needs, there is a pressing demand for a flexible, demand-driven, and multifunctional delivery platform that can dynamically integrate and orchestrate multiple delivery modes within a unified, behavior-aware system (Beck et al., 2025; Dos Santos et al., 2022). Specifically, compared with the hybrid framework of Dos Santos et al. (2022), which assumes exogenous availability of occasional couriers based on predefined detour thresholds and compensation rules, our work explicitly models both crowd-shippers' and customers' decisions through discrete-choice models, thereby capturing heterogeneous behavioral responses and their competition for limited depot capacity.

However, managing such an integrated system introduces several unique challenges that are not present in single-mode settings. These include 1) the need to model and coordinate heterogeneous participant behaviors across different order delivery channels; 2) the behavioral uncertainty associated with participant participation decisions, such as crowd-shipper acceptance or customer pickup willingness (Mohri et al., 2023); and 3) the computational complexity arising from the large-scale, combinatorial decision space with shared, capacity-constrained resources (e.g., mobile depots). These challenges motivate the design of our optimization framework, which explicitly accounts for participant behaviors, their interdependencies, and the operational trade-offs needed to efficiently allocate limited fulfillment resources in a dynamic urban logistics environment.

Moreover, irrespective of the physical mode employed, the success of any last-mile operation hinges on participant behavior. Ineffective order assignment or depot placement can trigger crowd-shippers to reject delivery tasks or customers to forgo pickup, wasting capacity and eroding network-wide performance. Although prior studies have optimized logistics decisions under crowd-shipper acceptance uncertainty (Hou et al., 2022) or customers' delivery-mode choice (Wang et al., 2020), they treat each participant group in isolation. How heterogeneous agents, with divergent spatial-temporal preferences and cost–benefit perceptions, compete for shared, capacity-constrained resources such as mobile-depot slots (Mohri et al., 2024), and how that competition should be managed within a unified hybrid architecture, remains an open research question. Addressing this gap, the present study develops a behavior-aware, hybrid order-delivery framework that jointly optimizes mobile-depot allocation and multi-mode order dispatch, thereby advancing both the feasibility and the efficiency of integrated last-mile logistics.

The main contributions of this paper are summarized as follows:

- We propose a hybrid order delivery system that exploits the flexibility of mobile depots to seamlessly integrate multiple delivery modes, including crowd-shipping, and customer self-pickup, thereby enabling highly efficient last-mile logistics.
- We model crowd-shippers' order-acceptance behavior and customers' self-pickup decisions using discrete choice models. These behaviorally-derived probabilities are embedded into a stochastic mixed-integer programming formulation, enabling robust optimization of order assignment and mobile depot positioning under demand-side uncertainty.
- We formulate a stochastic mixed-integer programming model that jointly determines the optimal allocation of mobile depots, order dispatch decisions, including delivery postponement, assignment to crowd-shippers, assignment to customer self-pickup, and a drone-based backup delivery, so as to minimize the expected total operating cost under behaviorally quantified uncertainties.

[1] https://wiserreview.com/blog/ecommerce-statistics/
[2] https://www.routific.com/blog/last-mile-delivery-costs
[3] https://flex.amazon.ca/
[4] https://www.dhl.de/en/privatkunden/pakete-empfangen/an-einem-abholort-empfangen/packstation.HTML
[5] https://www.7eleven.com.au/get-to-know-us/stories/news/Wonderfully-easier-new-options-for-Australia-Post-customers.HTML
[6] https://www.starship.xyz/
[7] https://www.scmp.com/tech/big-tech/article/3195587/jdcoms-logistics-arm-pushes-ahead-unmanned-robotic-grocery-deliveries

- We propose PRiME-DQN, a reinforcement learning approach that integrates behavior-Priority Resource filtering, action bIasing, and Model-Efficient temperature tuning for efficient problem-solving;
- We validate the cost-effectiveness and computational efficiency of the proposed model and solution approach using real-world data from Amazon's parcel delivery system and synthetic crowd-shipper network in the City of Seattle.

The remainder of this paper is organized as follows. In Section 2, we review the relevant literature and highlight the research gap. In Section 3, we introduce the main components and operational processes of the system, and formulate the problem. Section 4 propose the PRiME-DQN algorithm. Section 5 presents a case study and describes the experimental results. Finally, in Section 6, we conclude our work and propose several directions for future research.

2. Related works

In increasingly flexible last-mile delivery systems, multiple delivery modes, ranging from traditional delivery and crowd-shipping to customer self-pickup, are often integrated to meet diverse and rapidly evolving service demands. Within this complex landscape, explicitly accounting for the behavioral uncertainty of system participants has been shown to play a critical role in improving operational efficiency, participants satisfaction, and overall system responsiveness (Punel et al., 2018; Zhou et al., 2020). To position our work within this context, we review the existing literature on urban delivery optimization under participant behavioral uncertainty. Specifically, we organize the discussion by actor type, distinguishing between supply-side agents (e.g., couriers) who provide delivery capacity and demand-side agents (e.g., customers) who request services.

2.1. Courier behavior

Given the limited flexibility and low uncertainty typically associated with traditional delivery networks staffed by professional couriers, recent research has increasingly focused on crowd-sourced delivery systems, where occasional drivers or in-store customers contribute delivery capacity. In this subsection, we review optimization models and algorithmic strategies developed for such systems, with particular emphasis on works that explicitly capture behavioral uncertainty.

Some studies incorporate human behavioral uncertainty by simply assuming fixed probability distributions, which are then embedded into their models without explicitly learning or estimating these behaviors from data. For example, Fan et al. (2022) assume fixed compliance and oversight probabilities to construct a two-player model in which a food-delivery platform chooses strict versus passive management while crowd-sourced riders choose compliance versus violation of traffic rules, coupling managerial and behavioral decisions in a unified framework. The objective is to identify evolutionarily stable equilibrium that minimizes combined safety penalties and management cost. However, this paper does not construct an explicit behavioral model of uncertainty but assumes fixed compliance and oversight probabilities. Gdowska et al. (2018) introduce a bi-level stochastic optimization framework for same-day last-mile delivery that first selects a subset of customers to propose to occasional couriers and then routes a professional fleet to serve any rejected orders, jointly optimizing matching, compensation propositions, and backup routing decisions. The model's objective is to minimize expected total delivery cost under behavioral uncertainty captured by fixed acceptance probabilities for each occasional courier-order pairing. Mousavi et al. (2022) capture crowd-shipper availability uncertainty by modeling each driver's appearance as a Bernoulli random variable across a discrete set of demand scenarios. These stochastic realizations are embedded in a two-stage mixed-integer programming formulation, where first-stage decisions (mobile depot placements and tentative order assignments) are linked to second-stage recourse actions (actual parcel allocations or postponements) once driver availability is revealed. To the best of our knowledge, Dos Santos et al. (2022) is the first study to integrate multiple delivery modes while accounting for participant behavioral uncertainty. The paper pioneers a hybrid order delivery system that combines crowd-sourced delivery by occasional drivers with a customer self-pickup option, assuming a fixed probability that occasional drivers fail to appear. A mixed-integer linear programming model is formulated to minimize total operational cost and to derive door-to-door routing decisions for both the professional fleet and local fleet vehicles.

Other studies go further by explicitly modeling or training predictive models of human behavior, allowing for more accurate and data-driven representations of decision-making under uncertainty. For instance, Hou et al. (2022) employ a binomial logit discrete-choice model calibrated on real-world delivery data to estimate each driver's probability of accepting an assigned order. They develop a two-stage stochastic optimization framework for crowd-sourced delivery services. In the first stage, they solve a bipartite matching problem to assign crowd-shippers to orders so as to minimize total detour distances and unassigned orders; in the second stage, they determine individual compensation levels to maximize expected order acceptance and thus minimize the retailer's expected total delivery cost. Building on their earlier framework, Hou et al. (2025) conduct a systematic comparison of several machine-learning classifiers and identify XGBoost as the most accurate predictor of crowd-shipper acceptance behavior. They then embed the XGBoost-derived acceptance probabilities into a bilateral matching game and design customized incentive schemes to promote task acceptance, thereby ensuring stable matching and minimizing total delivery costs. Similarly, Wang et al. (2022) utilize XGBoost algorithm to predict the willingness of riders to grab each order in an limited-length recommendation list. And this study formulates a mixed-integer optimization problem in which the platform decides which orders to expose to each rider and in what position within a personalized recommendation list. The objective is to minimize a weighted sum of expected incremental travel cost and exposure imbalance.

In summary, given the increased complexity introduced by behavioral uncertainty, most of the above studies employ heuristic solution approaches, such as greedy algorithms (Gdowska et al., 2018), neighborhood search (Wang et al., 2022), or rule based

sequential heuristics (Hou et al., 2022), to achieve computational efficiency. Mousavi et al. (2022) pursue exact solutions by leveraging more advanced techniques, such as decomposition algorithms that combine branch-and-cut with cut-generation methods.

2.2. Customer behavior

In this subsection, we review studies that explicitly incorporate customer behavioral uncertainty into urban delivery system design and optimization.

Similar to the studies on supply-side participants in Section 2.1, customer behavior modeling can also be broadly categorized into two types. The first includes studies that adopt simplified probabilistic assumptions, where customer behaviors are modeled using fixed or exogenous probability distributions. For example, Sungur et al. (2010) propose a stochastic mixed-integer programming model to determine optimal master routing and daily scheduling plans, with the objective of maximizing the number of customers served while minimizing total travel time and deviations from baseline routes. Customer presence behavior is explicitly modeled as a scenario-based stochastic variable, with empirical probabilities derived from historical appearance patterns integrated into the optimization framework. Wu et al. (2019) consider a dynamic last-mile pickup routing problem involving both scheduled and on-demand customers. The model jointly optimizes sequential routing decisions, where the key decision variable at each step is the selection of the next customer to visit, with the objective of minimizing expected total travel distance under real-time system uncertainty. To explicitly capture behavioral uncertainty, the model assumes Poisson-distributed cancellations from on-demand customers and Poisson-distributed spontaneous arrival of scheduled customers. Specially, Ghannadpour et al. (2014) consider behavioral uncertainty in both customer request arrivals and service time preferences. Request uncertainty is handled dynamically as requests are revealed in real time, while service preference uncertainty is modeled using fuzzy time windows with triangular membership functions to quantify satisfaction levels. And this paper formulates a multi-objective dynamic vehicle routing and scheduling model with fuzzy time windows, aiming to compute the optimal assignment and sequencing of vehicles to dynamically arriving customer requests. The objective is to minimize fleet size, total travel distance, and waiting time while maximizing customer satisfaction. Liang et al. (2023) propose a bi-objective multi-period vehicle routing problem for perishable goods that simultaneously minimizes total transportation costs and maximizes customer satisfaction, which is modeled through freshness deterioration during delivery. Customer satisfaction is quantified through a zero-order freshness decay function, reflecting temporal degradation of perishables over multi-slot planning horizons with customer-specific time windows.

The second category comprises studies that explicitly model customer behavior using data-driven or choice-based approaches, such as discrete choice models or machine learning predictors. These works aim to capture customer preferences and decision-making processes more accurately. Specifically, Yang et al. (2024) propose a novel Restaurant Area Sizing Optimization framework that optimizes the Customer Service Area and Driver Dispatch Area for each restaurant in on-demand food delivery services. The objective is to maximize the total number of served orders while ensuring delivery time constraints are satisfied. the authors model customer ordering behavior using a multinomial logit model. Zhang et al. (2023b) propose a mathematical program with complementarity constraints to solve a joint location and pricing optimization problem for self-service stations in last-mile urban logistics. The model aims to maximize the total profit of logistics service providers by determining the optimal placement of self-service stations and corresponding service prices, while capturing the delivery service choice and route selection behavior of customers. Customer behavior uncertainties are rigorously modeled using nested logit choice models for delivery mode selection and Wardrop equilibrium for path choice

Most studies that incorporate customer behavior into last-mile delivery optimization rely on heuristic solution methods to address the resulting computational complexity. These include local search–based metaheuristics (Liang et al., 2023; Ghannadpour et al., 2014; Sungur et al., 2010), rule-based heuristics (Wu et al., 2019), and occasionally commercial solvers (Yang et al., 2024) applied to mixed-integer programming formulations. Additonally, A smaller subset of works addresses customer behavior through equilibrium-based modeling, particularly when customer choices influence system-wide outcomes (Zhang et al., 2023b). However, to the best of our knowledge, there is a notable absence of exact solution methods specifically tailored to behavior-aware models, particularly when customer choices are modeled via discrete choice or data-driven frameworks. This gap highlights both the inherent complexity introduced by behavioral modeling and the need for scalable, yet principled, optimization techniques.

2.3. Position of our work

Distinct from most of previous studies that typically focus on either crowd-shipping or self-service pickup independently, our work explicitly addresses a hybrid delivery setting where heterogeneous participant groups compete for limited mobile depot capacities. Although, Dos Santos et al. (2022) investigates a similar hybrid delivery system, it does not further consider or model participant behavior.

Moreover, considering the increase of problem complexity introduced by incorporating participant behavioral uncertainty, the majority of studies resort to heuristic methods, such as greedy algorithms, neighborhood search algorithms, or rule-based heuristics, which, while guaranteeing computational efficiency, do not achieve solution optimality. While Mousavi et al. (2022) achieve exact solutions by embedding Benders decomposition inside a branch-and-cut tree and further accelerating it with a cut-and-project enhancement, our work explicitly addresses a more complex hybrid setting involving heterogeneous participant groups whose behavioral uncertainties and interdependence are quantified through discrete-choice models. This additional complexity, arising from the nonlinear interactions between participant decisions and multiple delivery modes, significantly increases computational difficulty.

Table 1
Position of our work.

Reference	Delivery Modes Integrated	Endogenous Behavior	Behavior modeling	Behavior–decision coupling	Solution approach
Fan et al. (2022)	No	Driver traffic compliance	Bounded rationality	No	Evolutionary game
Gdowska et al. (2018)	No	Crowd-shipper acceptance	Probability distribution	Partial	Heuristic
Mousavi et al. (2022)	Mobile depot + Crowd-shipper	Crowd-shipper availability	Probability distribution	Partial	Benders decomposition
Dos Santos et al. (2022)	Fixed locker + Self-pickup + Occasional drivers	Occasional drivers show-up	Scenario-based	Partial	CPLEX
Hou et al. (2022)	No	Crowd-shipper acceptance	Binomial Logit	Partial	Heuristic
Hou et al. (2025)	No	Crowd-shipper acceptance	XGBoost	Partial	Stable matching
Wang et al. (2022)	No	Rider grab preference	XGBoost	No	Heuristic
Sungur et al. (2010)	No	Customer occurrence	Scenario-based	Partial	Tabu search
Wu et al. (2019)	No	Customer cancellation	Probability distribution	Partial	MDP
Ghannadpour et al. (2014)	No	Service-time preference	Fuzzy time windows	Partial	Genetic algorithm
Liang et al. (2023)	No	Customer satisfaction	Time-based function	Partial	Simulated annealing
Yang et al. (2024)	No	Customer ordering choice	XGBoost	Partial	Gurobi
Zhang et al. (2023b)	No	Customer service choice	Logit equilibrium	Partial	KKT
Our Paper	Mobile depots + Self-pickup + Crowd-shipping	Customer self-pickup + Crowd-shipper acceptance	Binomial logit	Yes	RL-based

Consequently, data-driven methods such as reinforcement learning have emerged as more efficient approaches to solve highly nonlinear and combinatorial problems involving interactions among heterogeneous participant groups, which are precisely the challenges posed by our hybrid last-mile delivery model (Yan et al., 2024).

In summary, as shown in Table 1, to the best of our knowledge, the existing literature on last-mile delivery optimization with human behavior typically considers only a single behavioral component, and often relies on partial coupling, where behavior-related decisions are separated from other operational decisions and solved in a staged manner to balance system efficiency and individual incentives (such as Mousavi et al. (2022), Dos Santos et al. (2022), Ghannadpour et al. (2014)). In contrast, this paper jointly models two endogenous behaviors, in particular customer self-pickup choice and crowd-shipper acceptance, using logit-based formulations. These behaviors exhibit conflicting interests due to competition for limited mobile depot capacity, and jointly influence multiple operational decisions, including mobile depot allocation and order assignment. By explicitly capturing these interactions, we formulate a unified stochastic mixed-integer programming model that integrates both behavioral responses and system-level objectives. Given the resulting computational complexity, we further develop a reinforcement learning–based solution approach, which has been shown to be effective for solving such strongly coupled large-scale stochastic optimization problems (Eysenbach et al., 2023).

3. Problem statement

In this section, we propose an innovative last-mile delivery system that leverages mobile depots to enhance the flexibility and reliability of order delivery. Customers can either pick up their orders from these depots or have a crowd-shipper, such as a neighbor, bring the orders to them. Orders that are neither picked up by customers nor delivered by crowd-shippers will be assigned to unmanned drones to ensure fulfillment. First, we introduce the system structure to provide an overview of its components and interactions. Then, we present a discrete event system model to capture the system's dynamics and simulate the order delivery process. Based on this representation, the allocation of mobile depots and order dispatch under uncertainty is formulated as a stochastic mixed-integer programming model to minimize overall operational cost. To incorporate behavioral uncertainty, we introduce two binomial logit models that estimate customer and crowd-shipper choices, ensuring a realistic representation of decision-making within the system. Table 2 lists all the notations used in the paper.

3.1. System overview

This system enables hybrid last-mile delivery by integrating four key operational components: platform, mobile depots (equipped with backup delivery capability via drones), customers, and crowd-shippers. The mobile depots are platform-controlled and serve as redistribution hubs to coordinate fulfillment tasks across customer self-pickup and crowd-sourced delivery. The interactions among these components form the foundation of the proposed behavior-aware, hybrid delivery framework. The overall system structure is illustrated in Fig. 1.

- Platform (logistics center): offers parcel delivery services, equipped with a fleet of mobile depots. Let $\mathcal{L}$ be the location of the logistics center. Decisions regarding the allocation of the mobile depots, orders dispatch, whether assigned to crowd-shippers, fulfilled by customer self-pickup, or scheduled for postponed delivery, are centrally managed by the platform. Furthermore, considering the possibility that customers or crowd-shippers may decline self-service or reject to deliver orders, each mobile depot is equipped with backup delivery drones to ensure service quality.

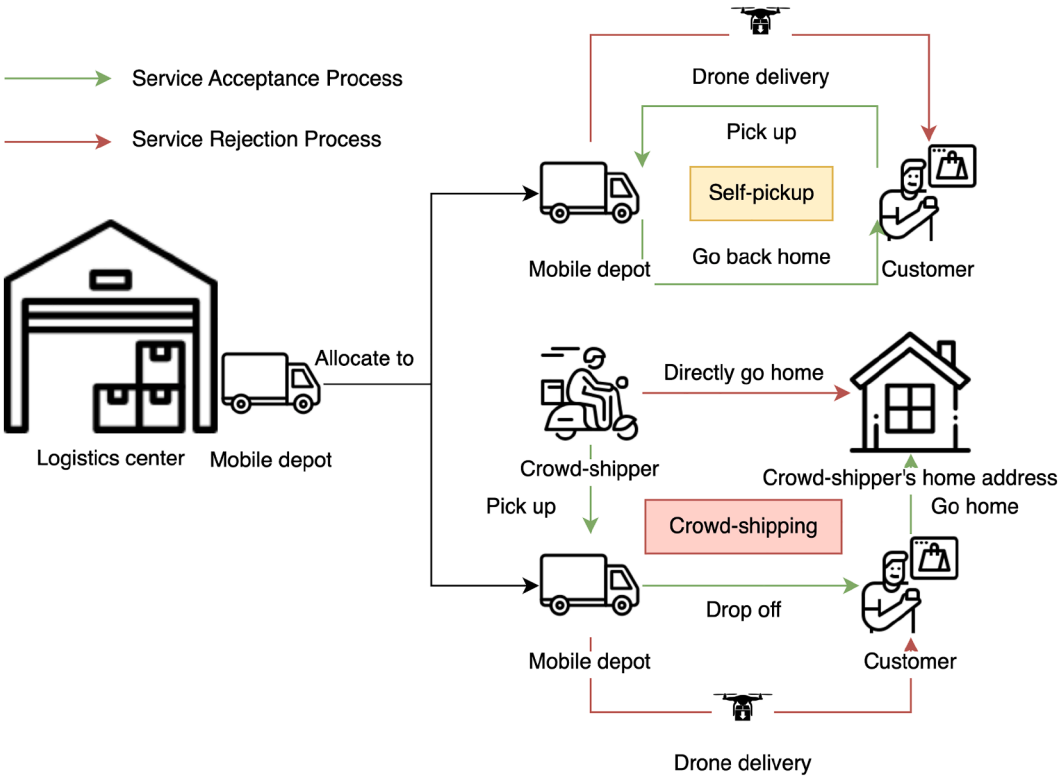


Fig. 1. System overview of mobile depots and hybrid order delivery model.

- Mobile depots: Each mobile depot has the same fixed capacity of Cap orders. They are loaded with specific orders at the logistics center and allocate to designated locations, where they wait for customers or crowd-shippers to pick up the parcels. The declined or rejected orders are delivered by the backup delivery drones. In this setup, drones are assumed to be available in unlimited numbers but are modeled as a costly fallback option, reflecting their role as a backup service, invoked only when crowd-shipping and self-pick-up fail, thereby discouraging their use through cost rather than explicit capacity constraints.We note that real-world drone deployment faces well-documented challenges, including limited payload, battery and charging constraints, regulatory restrictions, and safety concerns (Murray and Chu, 2015; Otto et al., 2018) Accordingly, drones are introduced here purely as a behavior-independent backup resource, rather than a primary delivery mode.
- Online customers (orders): Let J be the set of all orders. Each customer of order $j \in J$ shop on the platform (e.g. Amazon) uploads their personal addresses $d_j \in K$, and pay a delivery fee ρ_j to obtain last-mile delivery services. Customer addresses serve as candidate locations for mobile depot allocation. Additionally, some online customers are willing to travel a short distance to pick up their parcels in exchange for enhanced parcel security (stored in the depots to prevent loss). However, a portion of customers are unwilling to opt for self-pickup, preferring door-to-door delivery instead.
- Crowd-shippers: Let I be the set of crowd-shippers, partially overlap with online customers. They may slightly deviate from their usual route on the way home to pick up parcels from the mobile depot and deliver them to neighbors in exchange for a monetary incentive. Let o_i and d_i be the location of origin and destination of each crowd-shipper i. They first register as crowd-shippers via the platform's app or website. Delivery tasks are then assigned to their mobile devices along with an order pick-up code. Crowd-shippers can freely choose whether to accept the task. If they reject, they must notify the platform, allowing it to instruct the mobile depot to complete the delivery using drones.

3.2. Discrete event system modeling

To capture the system dynamics, and life cycle of an order, a discrete event system simulation model is introduced. For an order $j \in J$, the main events affecting its states are e^d, e^l, e^c, e^s, e^{a_c} (e^{a_s}), and e^{r_c} (e^{r_s}), which represents the order is *Postponed* to next decision epoch, the order is *Loaded and allocated* to a location, the order is assigned to a *crowd-shipper*, the order is given to customer *Self-service*, the order is *Accepted* by the crowd-shipper (customer), and the order is *Rejected* by the crowd-shipper (customer).

Let Ξ be the state space which consists of the set of possible values of the vector $\boldsymbol{\Xi}(n) = [\xi_1(n) \ \ldots \ \xi_j(n) \ \ldots \ \xi_{|J|}(n)]^T$, where $\xi_j(n)$ represents the state of the online order j after the n^{th} event happened. The value of the state variable $\xi_j(n)$ indicates if the order j:

- is available: $\xi_j(n) = \Xi_0$;

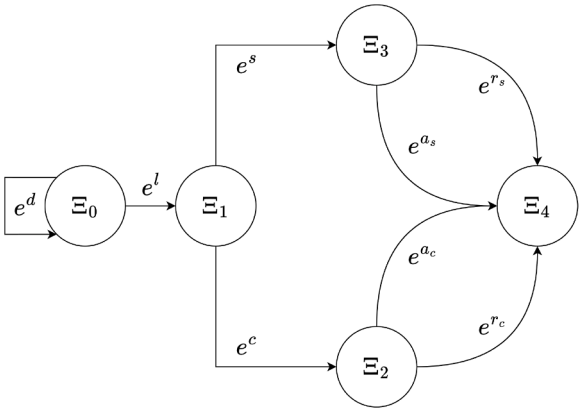


Fig. 2. State transition diagram.

- is at the allocated mobile-depot location: $\xi_j(n) = \Xi_1$;
- is evaluated by a crowd-shipper: $\xi_j(n) = \Xi_2$;
- is evaluated by the customer: $\xi_j(n) = \Xi_3$;
- is delivered: $\xi_j(n) = \Xi_4$.

The state transitions diagram of an order is shown in Fig. 2 and the details of some important events are illustrated below.

3.2.1. Order loading and allocating

A generic *order loading and allocating* event e^l, occurring at t, transits the order state from Ξ_0 to Ξ_1, based on the solution of the optimization model, described in Section 3.3. We assume that the logistic center operates the optimization model periodically, with each execution occurring at time intervals of T. The allocating costs C_k for each mobile depot include a fixed cost c_0 and a distance-based cost c_1, as defined by the following formula:

$$C_k = c_0 + c_1 \times D_{\mathcal{L},k}, \tag{1}$$

where $D_{\mathcal{L},k}$ denotes the distance between logistics center $\mathcal{L}$ and the location $k \in K$ where the mobile depot is allocated.

3.2.2. Order dispatch, acceptance and rejection events

The occurrences of order dispatch events e^c (e^s), change the order states from Ξ_1 to Ξ_2 (Ξ_3). Both crowd-shippers and customers are free to accept or decline the system assigned order delivery or self-service requests. The freedoms are defined as probability of accepting delivery, p^{a_c}, and accepting self-service, p^{a_s}. Let $p^{r_c} = 1 - p^{a_c}$, and $p^{r_s} = 1 - p^{a_s}$, be the rejection probabilities of delivery and self-service requests. The accepted orders are either delivered by crowd-shippers or self-picked up by customers. For the orders that are rejected, delivery is completed by the drones equipped with the mobile depot.

The compensation $C_{i,j,k}$ paid to each crowd-shipper i who accepts the assigned order j allocated in location k is defined as:

$$C_{i,j,k} = c_2 + c_3 \times \mathcal{D}_{i,j,k} \tag{2}$$

where c_2 denotes a fixed compensation paid to each crowd-shipper, c_3 represents the unit payment rate, and $\mathcal{D}_{i,j,k}$ is the detour distance for the crowd-shipper to deliver order j from location k, which is defined as:

$$\mathcal{D}_{i,j,k} = D_{o_i,k} + D_{k,d_j} + D_{d_j,d_i} \tag{3}$$

where $D_{o_i,k}$ denotes the distance between crowd-shipper i's origin o_i and location k, D_{k,d_j} denotes the distance between location k and order j's home address d_j, and D_{d_j,d_i} denotes the distance between d_j and crowd-shipper's destination d_i. It should be noted that the location k for allocating a mobile depot, determined by the solution of the Mobile Depot Allocation and hybrid Order Dispatch Model introduced in Section 3.3.

Orders rejected by crowd-shippers and customers are assigned to drones, incurring a cost $C_{jk} = c_4 + c_5 \times D_{d_j,k}$, where c_4 represents the fixed costs dispatch a drone, c_5 refers to the unit distance cost, and $D_{d_j,k}$ denotes the distance between order j's home address d_j and the location k of an allocated mobile depots.

3.3. Stochastic mixed integer programming formulation

In this study, the proposed optimization model is formulated for a specific decision epoch within a general last-mile delivery setting. At each decision epoch, orders with a status of Ξ_0 are considered available and are incorporated into the system's decision-making process. The key decisions include the allocation of mobile depots, determining the number of mobile depots to each location, and order dispatch decisions, consisting of whether to postpone certain orders, assign them to customer self-pickup, or delegate them

Table 2
Notation and description.

Set	Description
I, J, K	The set of all crowd-shippers (crowd-shippers), customers (orders), and available locations
$\hat{J}_{postponed}$, $\hat{J}_{self}$, $\hat{J}_{crowd}$	The set of postponed orders, self-service orders, and crowd-shipping orders
Γ	The set of location clusters, indexed by $\varsigma \in \Gamma$
Event	Description
e^c	The order is allocated to crowd-shippers
e^s	The order is allocated to customer self-service
e^p	The order dispatch is proposed to crowd-shipper/customer self-service
e^d	The order dispatch is postponed to next decision epoch
e^{a_c}	The order delivery task is accepted by the crowd-shipper
e^{a_s}	The order self-service dispatch is accepted to the customer
e^{r_c}	The order delivery dispatch is rejected by the crowd-shipper
e^{r_s}	The order self-service dispatch is rejected by the customer
Decision variable	**Description**
$x_{i,j,k}$	Binary variables: 1, if a crowd-shipper i is dispatched to a location k to pick up and deliver a specific order j; 0, otherwise
y_j	Binary variable: 1, if an order j is postponed to next decision epoch; 0, otherwise
z_k	Integer variable, denoting the number of mobile depots allocated at location k
$w_{j,k}$	Binary variable: 1, if an order j is loaded to one mobile depot sent to location k; 0, otherwise
Parameter	**Description**
ρ_j	The fee that a customer pays for same-day delivery
N_ς	Number of location clusters, pre-determined
N_ς^K	Number of mobile depots sent to a location cluster ς
$\mathcal{L}$	The location of logistics center
Cap	Maximum parcel capacity of each mobile depot
H	The penalty cost of each postponed order
C_k	The cost of allocating one mobile depot to a location k
$C_{i,j,k}$	The compensation paid to a crowd-shipper i who pick up and deliver order j from location k
$C_{j,k}$	The cost of delivering an order j from location k by a drone
$D_{i,j,k}$	Detour distance of a crowd-shipper i to pick up and deliver order j from mobile depot k
$D_{d_j,k}$	Distance between location k and home address of customer d_j
$p_{i,j,k}^{a_c}(p_{i,j,k}^{r_c})$	The probability that a crowd-shipper i accepts (rejects) to pick up and deliver order j from location k
$p_{j,k}^{a_s}(p_{j,k}^{r_s})$	The probability that a customer j accepts (rejects) to self-pickup his order from location k
o_i, d_i, d_j	The origin and destination of each crowd-shipper i, and customer j's home address
c_0, c_1	Fixed and distance-based cost rate of allocating a mobile depot;
c_2, c_3	Fixed and detour distance-based cost rate of compensating a crowd-shipper;
c_4, c_5	Fixed and distance-based delivery cost of drones

to crowd-shippers. Considering the uncertainties in customer and crowd-shipper behavior, this study formulates a stochastic mixed-integer programming model to compute the optimal mobile depot allocation and order dispatch solution, aiming to minimize overall operational costs. For each crowd-shipper $i \in I$, there is a probability that they will either accept or decline their matched order $j \in J$ at location $k \in K$. The cost of delivering this order can be visualized as a binary stochastic variable. This variable follows a Bernoulli distribution $\sim \mathcal{B}(p_{i,j,k}^{a_c})$. If a driver declines an order, which happens with a probability of $p_{i,j,k}^{r_c}$, the delivery cost becomes $C_{j,k}$. However, if a driver accepts the order (which happens with a probability $p_{i,j,k}^{a_c}$), the delivery cost is denoted by $C_{i,j,k}$. Similarly, for each customer $j \in J$, there is a probability that they will either accept or decline the self-pickup requests. The cost of fulfilling the order can be visualized as a binary stochastic variable. This variable follows a Bernoulli distribution $\sim \mathcal{B}(p_{j,k}^{a_s})$. If a customer declines the self-pickup, which happens with a probability of $p_{j,k}^{r_s}$, the delivery cost becomes $C_{j,k}$. However, if a customer accepts to self-pickup order (which happens with a probability $p_{j,k}^{a_s}$), the delivery cost is 0. Please refer to Section 3.4 for the computation of the probabilities. In addition, to account for the cost implications of orders that cannot be fulfilled immediately, we introduce a postponement penalty H. This penalty serves as a proxy for anticipated future costs, such as warehouse holding, late-delivery penalties, and the risk of service failure. By incorporating H, the model keeps postponement feasible yet costly, thereby limiting excessive deferral and preserving overall system efficiency.

Given a set of crowd-shippers I, a set of orders J, and a set of locations K. The optimization model is given below:

$$\min \quad F(z,y) + \mathbb{E}[Q(x)] + \mathbb{E}[G(w)] \tag{4}$$

$$\text{s.t. } y_j + \sum_{\forall k \in K} w_{j,k} + \sum_{\forall i \in I} \sum_{\forall k \in K} x_{i,j,k} = 1, \quad \forall j \in J \tag{5}$$

$$\sum_{\forall j \in J} w_{j,k} + \sum_{\forall i \in I} \sum_{\forall j \in J} x_{i,j,k} \le Cap \times z_k, \quad \forall k \in K \tag{6}$$

$$\sum_{\forall j \in J} \sum_{\forall k \in K} x_{i,j,k} \le 1, \quad \forall i \in I \tag{7}$$

$$x_{i,j,k}, y_j, w_{j,k} \in \{0,1\}, \quad \begin{cases} \forall i \in I \\ \forall j \in J \\ \forall k \in K \end{cases} \tag{8}$$

$$z_k \in \mathbb{N}, \quad \forall k \in K, \tag{9}$$

where,

$$F(z,y) = \sum_{\forall k \in K} C_k z_k + \sum_{\forall j \in J} H y_j$$

$$\mathbb{E}[Q(x)] = \sum_{\forall i \in I} \sum_{\forall j \in J} \sum_{\forall k \in K} (p^{a_c}_{i,j,k} C_{i,j,k} x_{i,j,k} + p^{r_c}_{i,j,k} C_{j,k} x_{i,j,k})$$

$$\mathbb{E}[G(w)] = \sum_{\forall j \in J} \sum_{\forall k \in K} p^{r_s}_{j,k} C_{j,k} w_{j,k}$$

In this formulation, the objective function is divided into three components. Let $F(z,y)$ be the sum of mobile depot allocation costs ($\sum_{\forall k \in K} C_k z_k$), and order postponed costs ($\sum_{\forall j \in J} H y_j$). Let $\mathbb{E}[Q(x)]$ be the expected order delivery costs fulfilled by crowd-shippers, and $\mathbb{E}[G(w)]$ be the expected order costs fulfilled by customer self-service. In addition, Eq. (5) ensure that each order must either be postponed, or fulfilled by customer self-service, or crowd-shippers. Eq. (6) restrict the maximum number of orders in each mobile depot. Eq. (7) guarantee that each crowd-shipper can only deliver at most one order from one mobile depot. Eq. (8) and Eq. (9) define the decision variables.

3.4. Discrete choice models for customers and crowd-shippers

In the optimization model, we have introduced two behavior probabilities: $p^{a_s}_{j,k}$, representing the probability of customers accepting self-service, and $p^{a_c}_{i,j,k}$, representing the probability of crowd-shippers accepting delivery requests. In this section, we propose two binomial logit discrete choice models, which are based on utility maximization theory and the rational economic agent assumption (McFadden, 1981), to analyze and predict the binary choice behaviors of customers and crowd-shippers.

3.4.1. Customers self-service acceptance model

Zhou et al. (2020) point out that customers' adoption of self-service parcel delivery is affected by perceived cost-effectiveness and service convenience. In particular, the study discusses cost savings as a key concern for customers and identifies walking distance as part of the facilitating conditions influencing self-service delivery adoption. Therefore, to simplify the model without sacrificing generality, we assume that the acceptance behavior of self-service pickup is primarily influenced by ρ_j, denoting the delivery fee paid by customer j, and $D_{d_j,k}$, denoting the distance between the mobile depot and the customer's home. The utility function for customer j choosing self-service option k is given by:

$$U_{j,k} = V_{j,k} + \epsilon_{j,k}, \tag{10}$$

where $V_{j,k}$ represents the deterministic component of utility, and $\epsilon_{j,k} \sim Gumbel(0,1)$, is an iid stochastic error term following a standard Gumbel distribution. The deterministic component $V_{j,k}$ is defined in Eq. (11),

$$V_{j,k} = \omega_0 + \omega_1 \rho_j + \omega_2 D_{d_j,k} \tag{11}$$

where, ω_0, denote the Alternative-Specific Constant (ASC), capturing the average effects of factors that influence utility but cannot be explicitly observed or are not included in the model specification.ω_1, and ω_2 are the marginal effects of a one-unit increase in the delivery fee and travel distance, respectively, on utility. In practical deployments, these parameters should be estimated from empirical data. However, due to the absence of publicly available datasets for this hybrid scenario, this study adopts a synthetic baseline calibrated with reference to Hou et al. (2022) and validates the robustness of the results through sensitivity analysis. The probability that customer j accepts the self-service option at mobile depot k follows a logistic function:

$$p^{a_s}_{j,k} = \frac{1}{1 + e^{-V_{j,k}}}. \tag{12}$$

3.4.2. Crowd-shipper order acceptance model

Hou et al. (2022) estimated a utility function describing crowd-shippers' order acceptance behavior using stated-preference survey data. In this paper, we adopt their utility specification, the utility function for a crowd-shipper i accepting order j from location k is given by:

$$U_{i,j,k} = V_{i,j,k} + \epsilon_{i,j,k}, \tag{13}$$

where $V_{i,j,k}$ represents the deterministic component of utility, and $\epsilon_{i,j,k} \sim Gumbel(0,1)$ is an iid stochastic error term following a standard Gumbel distribution. The deterministic component is defined as:

$$V_{i,j,k} = \lambda_0 + \lambda_1 \mathcal{D}_{i,j,k} + \lambda_2 C_{i,j,k}, \tag{14}$$

where $\mathcal{D}_{i,j,k}$ denotes the detour distance, $C_{i,j,k}$ represents the compensation offered to the crowd-shipper, and λ_0 denote the Alternative-Specific Constant, λ_1 and λ_2 represent the marginal effects of a one-unit increase in detour distance and compensation amount,

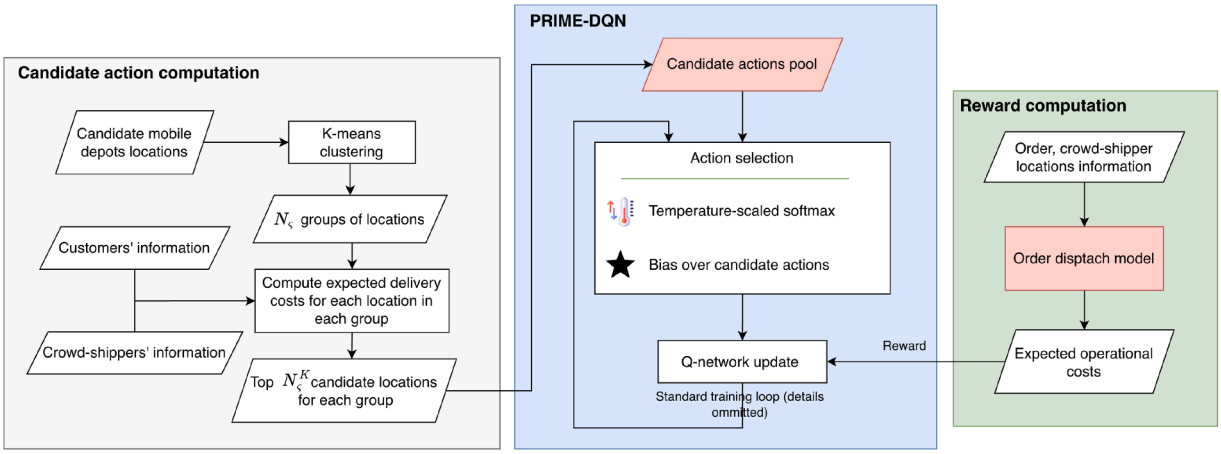


Fig. 3. Solution framework..

respectively, on utility. In practical deployments, these parameters should be estimated from empirical data. However, due to the absence of publicly available datasets for this hybrid scenario, this study adopts a synthetic baseline calibrated with reference to Hou et al. (2022) and validates the robustness of the results through sensitivity analysis. The probability of a crowd-shipper i accepts order delivery j from location k follows a logistic function:

$$p_{i,j,k}^{a_c} = \frac{1}{1+e^{-V_{i,j,k}}}. \tag{15}$$

4. Solution approach

Reinforcement learning (RL) is a learning paradigm in which an agent interacts with an environment and learns a decision-making policy by maximizing cumulative rewards through trial-and-error experience (Sutton et al., 1998). At each decision step, the agent observes a system state, selects an action, receives a reward, and transitions to a new state. A central concept in value-based RL is the action-value function, or Q-function, which measures the expected cumulative reward of taking a given action under a given state and following a certain policy thereafter. Classical Q-learning updates this action-value function based on the Bellman optimality principle (Watkins and Dayan, 1992). Deep Q-Networks (DQN) extend Q-learning by using a neural network to approximate the Q-function, denoted by $Q(s,a;\theta)$, where s is the state, a is the action, and θ represents the network parameters (Mnih et al., 2015). Instead of storing Q-values explicitly for all state-action pairs, DQN learns a functional approximation that can generalize across large state and action spaces. To improve training stability, the standard DQN framework commonly uses experience replay and a target network. Experience replay stores past transitions and samples them during training, while the target network provides a relatively stable target for Q-value updates.

The optimization model formulated in Section 3.3 incorporates various uncertainties, binary & integer decision variables, and capacity constraints that make traditional optimization techniques computationally expensive and challenging to apply in real-time environments. To address the computational challenges, we adopt a DQN-based learning framework as the core solution approach. Recent studies have shown that reinforcement learning methods, particularly value-based approaches, are effective for solving complex combinatorial optimization and mixed-integer decision problems by learning adaptive policies from data and system feedback (Eysenbach et al., 2023; Chen et al., 2022; Boysen et al., 2021). By integrating a custom-designed environment, heuristic actions filtering, and an embedded reward computation module, the DQN-based solution effectively addresses the mobile depots allocation and order dispatch problem while adapting to the uncertainties present in real-world scenarios. In our implementation, the reinforcement learning is modeled as a single-step decision process, where each episode consists of a one-shot allocation of all available orders. In other words, each episode has only one time step, during which all orders are simultaneously assigned. The overall framework, as illustrated in Fig. 3, consists of three components. The first component, Algorithm 1, is used to compute the "candidate action pool" and serves as one of the primary means to accelerate the convergence of the DQN-based algorithm. The second component, the PRIME-DQN algorithm, introduces bias values for high-priority actions and dynamic temperature tuning to balance exploitation and exploration. The third component involves a customized reward computation tailored to the optimization problem discussed in this paper.

4.1. State and action definition

We denote by $|J|$ the total number of orders. A state $\mathbf{s} \in S$ is a vector of length $|J|$, where each entry $s_j \in \{0\} \cup \{1,\dots,A-1\}$ represents the dispatch decision for order j.

$$\mathbf{s} = (s_1, s_2, \dots, s_{|J|}), \quad s_j \in \{0,1,\dots,A-1\}, \tag{16}$$

where A is the total number of discrete actions for a single order. At each step, the agent selects an action $\mathbf{a} = (a_1, \ldots, a_{|J|})$ from a multi-discrete set $\mathcal{A}^{|J|}$, where $\mathcal{A} = \{0, 1, \ldots, A-1\}$.

- $a_j = 0$ indicates that order j is postponed.
- $a_j = 1 + 2k$ or $a_j = 2 + 2k$ (for $k \in \{0, \ldots, |K| - 1\}$) assigns order j to location k under either a crowd-sourced or a self-service option.

Thus, if there are $|K|$ potential locations, each order has $A = 1 + 2K$ possible actions.

Algorithm 1 Location Clustering and Candidate Action Pool Generation.

Input: Set of mobile depot locations K, set of customer orders J, number of clusters N_c, mobile depot capacity Cap
Step 1: Clustering
Cluster K locations into N_c clusters
for each cluster $\varsigma \in \Gamma$ **do**
    $J_\varsigma \leftarrow$ customer orders in cluster ς
    $I_\varsigma \leftarrow$ shippers in cluster ς
    $N_\varsigma^K \leftarrow \lceil \frac{|J_\varsigma|}{Cap} \rceil$ ▷ Calculate the number of required depots based on Cap
end for
Step 2: Cost-Sorting and Location Selection
for each cluster $\varsigma \in \Gamma$ **do**
    Calculate the expected operation cost $\mathbb{E}[C_k]$ for each location $k \in K_\varsigma$
    Sort locations in ascending order of $\mathbb{E}[C_k]$
    Select the top N_ς^K locations with the lowest $\mathbb{E}[C_k]$ as the set of potential mobile depots $\mathbb{K}$
end for
Step 3: Candidate action pool generation
for each location $k \in \mathbb{K}$ **do**
    Add a_{2k+1} and a_{2k+2} into the candidate action pool $\mathcal{G}$
end for
Output: $\mathcal{G}$

4.2. Candidate action pool

Cheng et al. (2021) suggest that integrating heuristic information derived from domain knowledge or offline data into agents' decision-making processes can accelerate policy learning. This approach introduces heuristic functions to facilitate more effective exploration and exploitation during decision-making, thereby enhancing learning efficiency. To enhance exploration efficiency, this paper defines a "candidate action pool" $\mathcal{G}$ comprising actions with potential high priorities. This pool is dynamically integrated into the action selection process to bias exploration toward promising regions of the action space. We leverage user behavior patterns and the geographic attributes of all available locations to narrow the search space before the agent begins exploration. Specifically, we analyze user acceptance and cost performance across various location–service assignments, then combine these insights with geospatial considerations (e.g., travel distance, regional traffic patterns, and location capacity) to derive a ranked list of potentially superior actions for each order. This "candidate pool" reflects a balance between user preferences and operational feasibility. By focusing exploration on these high-value actions, we reduce computational overhead and accelerate convergence, while still maintaining the flexibility to discover novel, potentially lower-cost assignments outside the candidate set.

To this end, we introduce a K-Means-based clustering algorithm to categorize all possible locations into N_ς groups, as shown in Algorithm 1. Within each cluster $\varsigma \in \Gamma$, we count the number of customer orders $|J|$ and shippers $|I|$ whose home addresses l_j and l_i fall within that cluster, respectively. The number of mobile depots assigned to a specific cluster is determined as $N_\varsigma^K \leftarrow \lceil \frac{|J_\varsigma|}{Cap} \rceil$. Given that mobile depots are deployed at locations within the cluster $\varsigma \in \Gamma$, the expected operational cost is calculated as follows:

$$\mathbb{E}[C_k] = \sum_{\forall i \in I_\varsigma} \sum_{\forall j \in J_\varsigma} (p_{i,j,k}^{a_c} C_{i,j,k} + p_{i,j,k}^{r_c} C_{j,k} + p_{j,k}^{r_s} C_{j,k}) \tag{17}$$

where $\sum_{\forall i \in I_\varsigma} \sum_{\forall j \in J_\varsigma} (p_{i,j,k}^{a_c} C_{i,j,k} + p_{i,j,k}^{r_c} C_{j,k})$ denotes the expected costs delivered by shippers, and $\sum_{\forall i \in I_\varsigma} \sum_{\forall j \in J_\varsigma} p_{j,k}^{r_s} C_{j,k}$ denotes the delivery costs of rejected self-service orders. Based on expected delivery costs, all locations are sorted in ascending order, and the top N_ς^K locations are selected as potential locations for deploying mobile depots. These selected locations serve as substitutes within the candidate action pool $\mathcal{G}$.

4.3. Reward computation

4.3.1. Operational cost computation

Given the action **a**, the overall operational costs $C(\mathbf{a})$ can be computed through the following processes. The operational cost consists of three components: order postponement, mobile depot allocation, and order fulfillment. In turn, order fulfillment costs can be subdivided into crowd-shipping and drone-shipping.

- **Order postponement cost**: In our setting, any order j with action $a_j = 0$ is considered a *postponed* order.

$$\bar{J}_{postponed} = \left\{ j \;\middle|\; a_j = 0 \right\}. \tag{18}$$

Each postponed order incurs a per-order holding cost H. We denote by $\bar{J}_{\text{postponed}}$ the subset of orders for which $a_j = 0$, the total order postponement costs are the number of postponed orders multiplied by the unit holding cost H, denoted by $C_{\text{holding}} = \left|\bar{J}_{\text{postponed}}\right| \cdot H$.

- **Mobile depot allocation cost**: Suppose we have $|J|$ orders, each of which is assigned an action $a_j \in \{0, 1, \ldots, A-1\}$, if $a_j = 0$, j is postponed, or otherwise (for $a_j \neq 0$), $\left\lfloor \frac{a_j - 1}{2} \right\rfloor$ indicates the *location index* chosen for order j. We denote this mapping more explicitly:

$$k_j = \begin{cases} \left\lfloor \dfrac{a_j - 1}{2} \right\rfloor, & \text{if } a_j \neq 0, \\ -1, & \text{if } a_j = 0 \quad \text{(postponed)}. \end{cases} \tag{19}$$

For each location k, the set of orders assigned is denoted by $J_k = \left\{ j \;\middle|\; k_j = k \right\}$. Suppose each mobile depot can handle Cap orders, where Cap denotes the maximum number of orders a mobile depot can be loaded. Then, the total number of mobile depots dispatched to location k is the integer division of $|J_k|$ by the capacity: $\text{MobileDepots}(k) = \left\lceil \frac{|J_k|}{Cap} \right\rceil$. Hence, for every location k, we batch together Cap orders per depot. The costs of allocating mobile depots can be represented as: $C_{\text{allocation}} = \sum_{\forall k \in K} C_k \left\lceil \frac{|J_k|}{Cap} \right\rceil$.

- **Order dispatch cost**: Let $\dot{J}_{self}$ be the set of orders assigned to customer self-pick up, defined as follows:

$$\dot{J}_{self} = \left\{ j \;\middle|\; a_j \neq 0, \text{and } (a_j - 1) \; mod \; 2 = 1 \right\}. \tag{20}$$

For each order $j \in \dot{J}$, allocated to location k_j, let $p^{a_s}_{j,k_j}$ denote the probability that this order can be fulfilled by self-service. The cost incurred for an order $j \in \dot{J}_{self}$ is defined as follows:

$$\mathbb{E}^s[C_j] = \begin{cases} 0, & \text{if } p^{a_s}_{j,k} > threshold_s, \\ C_{j,k_j}, & \text{otherwise}. \end{cases} \quad \forall j \in \dot{J}_{self} \tag{21}$$

where $threshold_s$ refers to the tolerance parameter used to measure customers' willingness for self-service.

Let $\tilde{J}_{crowd}$ be the set of orders available to be matched with shippers:

$$\tilde{J}_{\text{crowd}} = \left\{ j \;\middle|\; a_j \neq 0, \text{and } (a_j - 1) \; mod \; 2 = 0 \right\}. \tag{22}$$

Given $\tilde{J}_{crowd}$ and the set of available crowd-shippers I, we formulate an optimization model to minimize the expected crowd-shipping costs.

$$\min_x \quad \mathbb{E}[Q(x)] + \sum_{\forall j \in \tilde{J}_{crowd}} C_{j,k_j} (1 - \sum_{\forall i \in I} x_{i,j}) \tag{23}$$

$$\text{s.t.} \sum_{\forall i \in I} x_{i,j} \leq 1, \quad \forall j \in \tilde{J}_{crowd} \tag{24}$$

$$\sum_{\forall j \in \tilde{J}_{crowd}} x_{i,j} \leq 1, \quad \forall i \in I \tag{25}$$

$$x_{i,j} \in \{0, 1\}, \quad \forall i \in I, \; \forall j \in \tilde{J}_{crowd}, \tag{26}$$

where, $\mathbb{E}[Q(x)] = p^{a_c}_{i,j,k_j} C_{i,j,k_j} x_{i,j} + p^{r_c}_{i,j,k_j} C_{j,k_j} x_{i,j}$.

The optimization model (order dispatch model) can be solved using the Hungarian algorithm or commercial solvers such as Gurobi and CPLEX. The solution yields a set of optimal values $\{x^*_{i,j}\}$. For each $j \in \tilde{J}_{\text{crowd}}$, the order j is *matched* to shipper i when there exists an $i \in I$ with $x^*_{i,j} = 1$. The order j is *unmatched* when $\sum_{\forall i \in I} x^*_{i,j} = 0$.

Let $p^{a_c}_{i,j,k_j}$, and $p^{r_c}_{i,j,k_j}$ be the probability that an order j is accepted and rejected by an assigned shipper i, respectively. The cost incurred for an order $j \in \tilde{J}_{crowd}$ is defined as follows:

$$\mathbb{E}^c[C_j] = \begin{cases} C_{i,j,k_j}, & \text{if } p^{a_c}_{i,j,k_j} > threshold_c, \\ C_{j,k_j}, & \text{otherwise}. \end{cases}, \quad \forall j \in \tilde{J}_{crowd} \tag{27}$$

where $threshold_c$ refers to the tolerance parameter used to measure shippers' willingness for delivery. The costs of order fulfillment can be represented as:

$$C_{\text{fulfillment}} = \sum_{\forall j \in \bar{J}_{self}} \mathbb{E}^s[C_j] + \sum_{\forall j \in \bar{J}_{crowd}} \mathbb{E}^c[C_j] \tag{28}$$

To this end, the overall operational cost $C(\mathbf{a})$ is represented as follows:

$$C(\mathbf{a}) = C_{\text{holding}} + C_{\text{allocation}} + C_{\text{fulfillment}} \tag{29}$$

4.3.2. Reward function

Our reward function $R(\mathbf{a})$ is designed to encourage the agent to find actions that reduce overall cost relative to both the *previous* cost (from the last episode) and the *historical best* cost (the lowest cost observed so far). The reward function $R(\mathbf{a})$ consists of three components:

- The penalty term $-C(\mathbf{a})$, which encourages the agent to minimize the operational cost;
- The relative improvement term $\Delta_{\text{prev}}(\mathbf{a})$, which measures the improvement relative to the previous cost C_{prev}; and
- The global improvement term $\Delta_{\text{best}}(\mathbf{a})$, which measures the improvement relative to the best cost C_{best} found so far.

Specifically, to provide a reward signal for actions that reduce the delivery cost relative to the previous solution, the relative improvement term $\Delta_{\text{prev}}(\mathbf{a})$ is formulated as follows:

$$\Delta_{\text{prev}}(\mathbf{a}) = \begin{cases} +\iota_{\text{prev}}, & \text{if } C(\mathbf{a}) < C_{\text{prev}}, \\ -\eta_{\text{prev}}, & \text{otherwise.} \end{cases} \tag{30}$$

The global improvement allows the agent to receive positive feedback upon discovering a new global minimum by updating C_{best}. Thus, the global improvement $\Delta_{\text{best}}(\mathbf{a})$ is represented:

$$\Delta_{\text{best}}(\mathbf{a}) = \begin{cases} +\iota_{\text{best}}, & \text{if } C(\mathbf{a}) < C_{\text{best}}, \\ -\eta_{\text{best}}, & \text{otherwise,} \end{cases} \tag{31}$$

where ι_{best}, η_{best} ι_{prev}, and η_{prev} are the penalty parameters. To this end, the reward function $R(\mathbf{a})$ is represented as follows:

$$R(\mathbf{a}) = C(\mathbf{a})(-1 + \Delta_{\text{prev}}(\mathbf{a}) + \Delta_{\text{best}}(\mathbf{a})). \tag{32}$$

4.4. PRiME-DQN Algorithm

To address our problem, this paper proposed a DQN-based method (named PRiME-DQN) to calculate optimal order assignment policy, as shown in Algorithm 2. The PRiME-DQN algorithm is initialized by defining the Q-network parameters, denoted by θ, and an experience replay memory $\mathcal{M}$. The algorithm also sets the initial action bias term, B_{init}, and temperature parameter, τ_{init}, which regulate the exploration-exploitation balance. Additionally, the bias decay rate, β_B, and temperature decay rate, β_τ, are initialized to control the rate at which these values diminish over episodes. The discount factor γ and learning rate α are set to ensure numerical stability in Q-learning updates. The input to the algorithm includes a set of candidate actions $\mathcal{G}$. The algorithm outputs an optimized order assignment policy π^*. The algorithm operates in an episodic manner, iterating over a predefined number of episodes, $E_{\max}$. The action bias B_e and temperature T_e are computed based on their respective decay functions, as shown in Eq. (33) and Eq. (34), respectively. Specifically, B_e follows an exponential decay schedule, ensuring that action biasing is effective in early episodes but diminishes over time. Similarly, T_e controls the degree of exploration and decays gradually to transition towards a greedy policy.

For each state, the Q-values corresponding to all possible actions are computed using the Q-network. These values are utilized within a Softmax function that incorporates the action bias term B_e to adjust the selection probability of certain actions. Eq. (35) ensures that initially, actions favored by Algorithm 1 receive higher selection probabilities while maintaining exploration. It should be noted that, in Eq. (35), the SoftMax is applied to the entire action set $\mathcal{A}$, while actions belonging to the candidate pool $\mathcal{G}$ are given an additional bias term to increase their selection probability. Actions outside $\mathcal{G}$ remain unaffected, and the normalization in the denominator ensures that the probabilities sum to one. Over time, as B_e and T_e decay, the selection probability converges to a standard greedy policy. The agent samples an action from the probability distribution $P_t(\mathbf{a})$ and executes the selected action $\mathbf{a}_t$. Upon execution, the environment returns an immediate reward r_t and a new state $\mathbf{s}_{t+1}$. The transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ is stored in the experience replay memory $\mathcal{M}$ for later training.

To stabilize learning, the algorithm performs batch updates when the size of the replay memory exceeds a predefined threshold BS. A mini-batch of transitions is sampled from $\mathcal{M}$, and the target Q-value for each transition is computed using the Bellman equation, refer to Eq. (36). The loss function is defined as the mean squared error between the current Q-values and the target Q-values, refer to Eq. (37). Gradient descent is applied to update the Q-network parameters. This ensures that the Q-network progressively refines its estimates of action values, leading to policy improvement. The entire training process continues until all episodes have been executed, yielding an optimized policy π^* that maps states to optimal actions.

Algorithm 2 PRiME-DQN.

Input: Q-network parameters θ, replay memory $\mathcal{M}$, bias B_{init}, temperature τ_{init}, bias decay rate β_B, temperature decay rate β_τ, discount factor γ, learning rate α, maximum training episode $E_{\max}$, and candidate action pool $\mathcal{G}$.
Training:
for each episode $e = 1, \dots, E_{\max}$ **do**
  Calculate action bias B_e and temperature T_e

$$B_e = \begin{cases} B_{\text{init}}, & e < E_B \\ B_{\text{init}} \cdot e^{-\beta_B (e - E_B)}, & e \geq E_B \end{cases} \tag{33}$$

$$T_e = \begin{cases} \tau_{\text{init}}, & e < E_\tau \\ \tau_{\text{init}} \cdot e^{-\beta_\tau (e - E_\tau)}, & e \geq E_\tau \end{cases} \tag{34}$$

  Calculate estimated Q-value $Q(\mathbf{s}, \mathbf{a}; \theta)$
  Calculate action probability $P(\mathbf{a})$

$$P(\mathbf{a}) = \begin{cases} \dfrac{\exp\left((Q(\mathbf{s},\mathbf{a};\theta)+B_e)/T_e\right)}{\sum_{\forall \mathbf{a} \in \mathcal{G}} \exp\left((Q(\mathbf{s},\mathbf{a};\theta)+B_e)/T_e\right) + \sum_{\forall \mathbf{a} \in \mathcal{A} \setminus \mathcal{G}} \exp\left((Q(\mathbf{s},\mathbf{a};\theta))/T_e\right)}, & \forall \mathbf{a} \in \mathcal{G} \\ \dfrac{\exp\left((Q(\mathbf{s},\mathbf{a};\theta))/T_e\right)}{\sum_{\forall \mathbf{a} \in \mathcal{G}} \exp\left((Q(\mathbf{s},\mathbf{a};\theta)+B_e)/T_e\right) + \sum_{\forall \mathbf{a} \in \mathcal{A} \setminus \mathcal{G}} \exp\left((Q(\mathbf{s},\mathbf{a};\theta))/T_e\right)}, & \forall \mathbf{a} \in \mathcal{A} \setminus \mathcal{G} \end{cases} \tag{35}$$

  Select a random action $\mathbf{a}_t$ based on $P(\mathbf{a})$
  Observe reward r_t and next state $\mathbf{s}_{t+1}$ by executing $\mathbf{a}_t$
  Store $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in $\mathcal{M}$
  if $|\mathcal{M}| > BS$ **then**
    Sample random minibatch $\{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})\}$ from $\mathcal{M}$
    Calculate target Q-value Q_{target}

$$Q_{\text{target}} = r_t + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}_t, \mathbf{a}'; \theta) \tag{36}$$

    Calculate training loss $\mathcal{L}(\theta)$

$$\mathcal{L}(\theta) = \left(Q_{\text{target}} - Q(\mathbf{s}_t, \mathbf{a}_t; \theta)\right)^2 \tag{37}$$

    Perform gradient descent on $\mathcal{L}(\theta)$ with respect to θ
  end if
end for
Output: Optimal order assignment policy π^*

5. Computational study

In this section, we compare the proposed mobile depots-based hybrid order delivery model (*Hybrid*) with two single order delivery benchmark models: the self-service model (*Self*) and the crowd-sourced delivery service model (*CDS*), which are introduced in Appendix A and Appendix B, respectively. Additionally, we demonstrate the computational efficiency of the proposed *PRiME-DQN* algorithm relative to the commercial solver *Gurobi*. The experiments in this study are based on last-mile delivery data from Amazon in Seattle, USA, and crowd-shipper location data from the Pronto Cycle Share system, which serve as order and crowd-shipper geographic information, respectively. Section 5.1 provides an overview of these datasets. Section 5.2 compares the cost-effectiveness of the proposed *Hybrid model* against *Self model* and *CDS model*, as well as analyzes the impact of mobile depot capacity on cost-effectiveness. In Section 5.3, we present the convergence curve of the *PRiME-DQN* algorithm on small-scale instances and compare both the optimization performance and computational efficiency on larger-scale datasets.

5.1. Dataset

This paper uses two sets of real-world data to validate the feasibility of the proposed approach. For the demand side, represented by online customers, we use Amazon's last-mile delivery data from the city of Seattle (Merchán et al., 2024), which includes the geographic locations of one logistics center and 259 delivery points. For the supply side, we utilize data from the Pronto Cycle Share system[8] in Seattle, which contains the locations of each cycle docker (58 unique locations) and records of 178,105 trips.

[8] https://www.kaggle.com/datasets/pronto/cycle-share-dataset

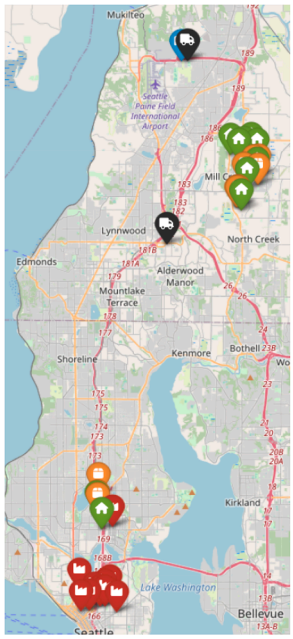


Fig. 4. The blue marker represents the logistics center, the orange markers denote the locations of Amazon customers, the red markers indicate the crowd-shippers' workplaces, the green markers represent the crowd-shippers' destinations, and the black markers are the mobile depots. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
Model parameter settings.

Parameter	Description	Value
H	Holding costs of an order	15
c_0	Fixed dispatching cost of a mobile depot	20
c_1	Energy consumption cost of mobile depot per kilometer	1
c_2	Fixed compensation paid to each crowd-shipper	4
c_3	Compensation rate per detour distance	1
c_4	Fixed delivery cost of dispatching a drone	5
c_5	Drone delivery rate per kilometer	1.1
Cap	Defaulted capacity of each mobile depot	20
$\omega_0, \omega_1, \omega_2$	Coefficients corresponding to customers' attributes	3, -0.1, -0.5
$\lambda_0, \lambda_1, \lambda_2$	Coefficients corresponding to crowd-shippers' attributes	-2, -0.4, 0.7

Specifically, the destination of each order is uniformly selected from 259 locations, and similarly, the original trip of each crowd-shipper is uniformly selected from 178,105 trips. The uniformly sampled geographic data (10×10) are illustrated in Fig. 4, where the blue marker represents the logistics center, specifically Amazon's *DSE4* warehouse, the orange markers represent the Amazon customer locations, the red markers represent the workplaces of the crowd-shippers, and the green markers represent the destinations of the crowd-shippers. Furthermore, the black markers are the mobile depots, and the allocation decisions of the mobile depots are determined by the optimization model, described in Section 3.3.

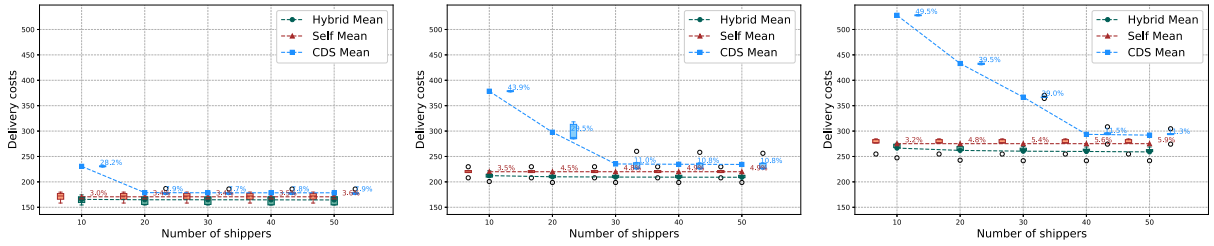
The experiments were conducted on a MacBook Pro, equipped with an Apple M4 Pro-chip, 24 GB of memory, running macOS 15.1.1. The reinforcement learning environment was implemented using the OpenAI Gym framework, and the neural network models were built and trained using PyTorch. The optimization model was implemented in Gurobi 11. And the aforementioned parameters, including their descriptions and values, are illustrated in Table 3. To ensure reproducibility, five different random seeds ($[26, 42, 13, 17, 36]$) were used for sampling during the experiments.

5.2. The potential commercial value of the hybrid model compared to the single-order delivery model

In this section, we compare the cost-effectiveness of the proposed hybrid order delivery model with crowd-sourced delivery and self-pickup service models. Additionally, we examine the impact of different mobile depot capacities on overall operational costs.

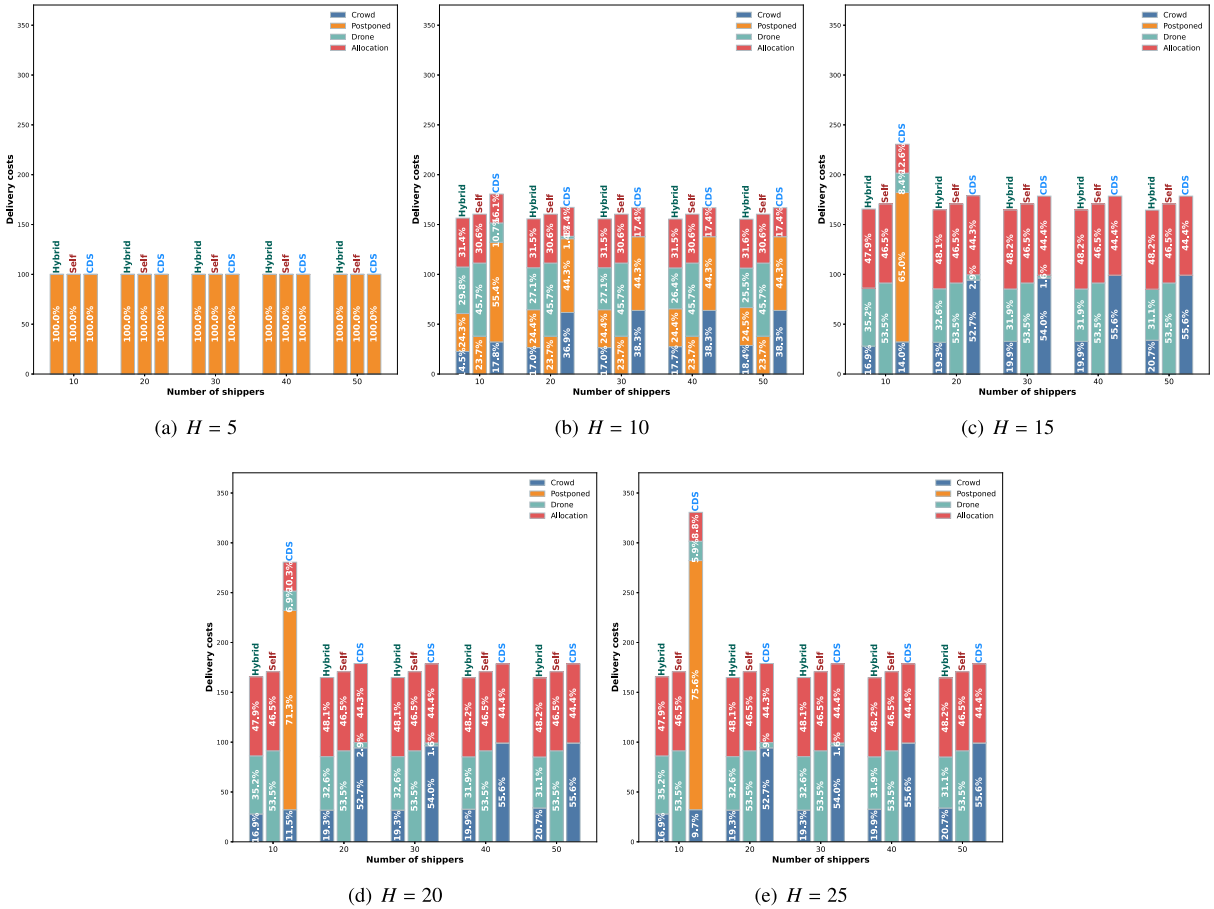
5.2.1. Delivery costs comparison

Fig. 5 presents a comparative analysis of delivery costs among three models, *Hybrid, Self, and CDS* across varying order quantities ($|J| = 20, 30$, and 40) and different numbers of crowd-shippers. The key observations are summarized as follows. The Hybrid model



(a) $|J| = 20$ (b) $|J| = 30$ (c) $|J| = 40$

Fig. 5. Comparison results of the three models (Hybrid, Self, and CDS) in terms of Delivery costs across different order quantities $|J|$ (from 20 to 40).



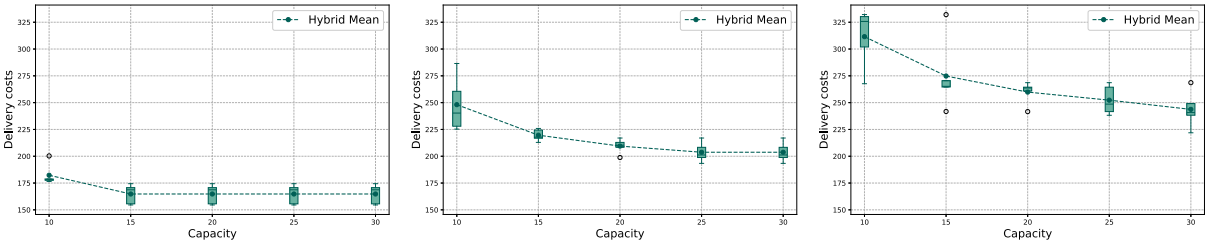
(a) $H = 5$ (b) $H = 10$ (c) $H = 15$

(d) $H = 20$ (e) $H = 25$

Fig. 6. Comparison results of the three models (Hybrid, Self, and CDS) in terms of delivery costs across different postpone penalty H (from 5 to 25).

consistently demonstrates superior cost-effectiveness, achieving the lowest delivery costs across all scenarios and emerging as the most effective solution. In comparison, the Self model incurs slightly higher delivery costs, with the cost difference between the two models ranging from approximately 3.0% to 5.9%, depending on the number of crowd-shippers and order quantities. On the other hand, the CDS model shows a sharp decline in delivery costs as the number of crowd-shippers increases, indicating its sensitivity to crowd-shipper availability. However, despite this improvement, the CDS model remains the least cost-effective option, with delivery costs significantly higher than those of both the Hybrid and Self models across all configurations.

In conclusion, the Hybrid model consistently outperforms the Self and CDS models by achieving the lowest delivery costs and maintaining stability across varying operational conditions. The CDS model, while benefiting from increased crowd-shipper availability, remains the least cost-effective option overall. These findings highlight the Hybrid model's potential for optimizing delivery operations in dynamic and scalable logistics environments.



(a) $|J| = 20$ (b) $|J| = 30$ (c) $|J| = 40$

Fig. 7. Impact of mobile depots capacity on delivery costs across varying number of orders ($|J| = 20, 30, 40$).

5.2.2. *Postponement penalty analysis*

To investigate the impact of different penalty values on the overall delivery cost and its components, including mobile allocation cost (Allocation), order postponement cost (Postponed), drone delivery cost (Drone), and crowd-sourced delivery cost (Crowd), we conducted a sensitivity analysis of this parameter. The number of orders was fixed ($|J| = 20$), and we examined the variations in the average delivery cost and cost composition across five random seeds as the number of shippers increased.

As shown in Fig. 6, when $H = 5$, postponement dominates the cost composition across all models. This outcome reflects that when the penalty is negligible, the optimization systematically postpones orders to reduce current-epoch delivery costs, thereby suppressing the use of crowd-shipping and self-service. As the penalty increases to $H = 10$, the share of postponed costs decreases substantially. This structural change is intuitive: a higher penalty makes postponement less attractive, forcing the system to complete more orders immediately, even at the expense of higher operational costs. When $H \geq 15$, postponement is almost eliminated, especially for Hybrid and Self models.

5.2.3. *Mobile depots capacity analysis*

As shown in Fig. 7 , we observe that as capacity increases, delivery costs exhibit a sharp downward trend until capacity approaches the order quantity, at which point the costs gradually converge and remain stable. Moreover, with an increase in order quantity, the impact of capacity on delivery costs becomes more pronounced. When the $|J| = 20$, increasing capacity from 10 to 30 results in an approximate 8.79% cost reduction, whereas at $|J| = 40$, the reduction reaches approximately 21.47%. In conclusion, the capacity size is negatively correlated with delivery costs, and larger mobile depots offer better cost effectiveness. However, it is important to note that factors such as mobile depot procurement costs, parking constraints, and traffic congestion, which are beyond the scope of this cost analysis, may reduce the marginal benefits of increasing capacity.

5.2.4. *Sensitivity analysis of behavioral parameters*

Since historical data and transferable parameter estimates for the proposed hybrid delivery setting are not available, we use empirically motivated baseline values for these behavioral parameters, introduced in Section 3.4. In particular, the scale and signs of the baseline coefficients are selected with reference to the behavioral modeling framework in Hou et al. (2022), which also characterizes acceptance behavior using price- and distance-related attributes. Specifically, both settings involve the trade-off between monetary incentives or costs and travel-related burden. Therefore, we adopt coefficients with comparable utility scales as the baseline setting and further evaluate the robustness of the computational results through a systematic sensitivity analysis.

For each of the six behavioral parameters, including the customer self-pickup parameters ω_0, ω_1, and ω_2, and the crowd-shipper acceptance parameters λ_0, λ_1, and λ_2, we consider three levels: low, baseline, and high. The baseline values are those reported in Table 3. The low and high levels are generated by decreasing and increasing the baseline value by 20%, respectively. For negative-valued parameters, “low” and “high” refer to the lower and higher numerical values, respectively. This design allows us to examine whether the main conclusions are sensitive to moderate perturbations of the behavioral coefficients. The sensitivity analysis is conducted in a global manner. When evaluating a given parameter, we fix this parameter at one of its three levels and vary the remaining five behavioral parameters over all their low, baseline, and high levels. For each complete parameter combination, we run the experiment under five random seeds and first compute the seed-averaged delivery cost on a 20×20 instance with 20 crowd-shippers and 20 customer orders. The boxplots in Fig. 8 report the distribution of these seed-averaged delivery costs across all combinations of the remaining five behavioral parameters. The connected markers indicate the mean of the seed-averaged delivery costs at each parameter level.

Fig. 8 shows that the delivery costs are more sensitive to the customer-side behavioral parameters than to the crowd-shipper-side parameters. In particular, ω_0 and ω_2 exhibit clear decreasing trends in mean delivery costs as their parameter levels increase. This result suggests that customers’ general willingness to adopt self-pickup and their sensitivity to pickup distance have a significant impact on system performance. By contrast, ω_1 has a relatively limited effect on delivery costs within the tested range. For the crowd-shipper acceptance parameters, λ_0 and λ_2 show moderate impacts, while λ_1 has only a minor influence. Overall, the results indicate that the main conclusions remain robust under moderate variations of the behavioral parameters, although the system is particularly responsive to parameters governing customers’ self-pickup acceptance behavior.

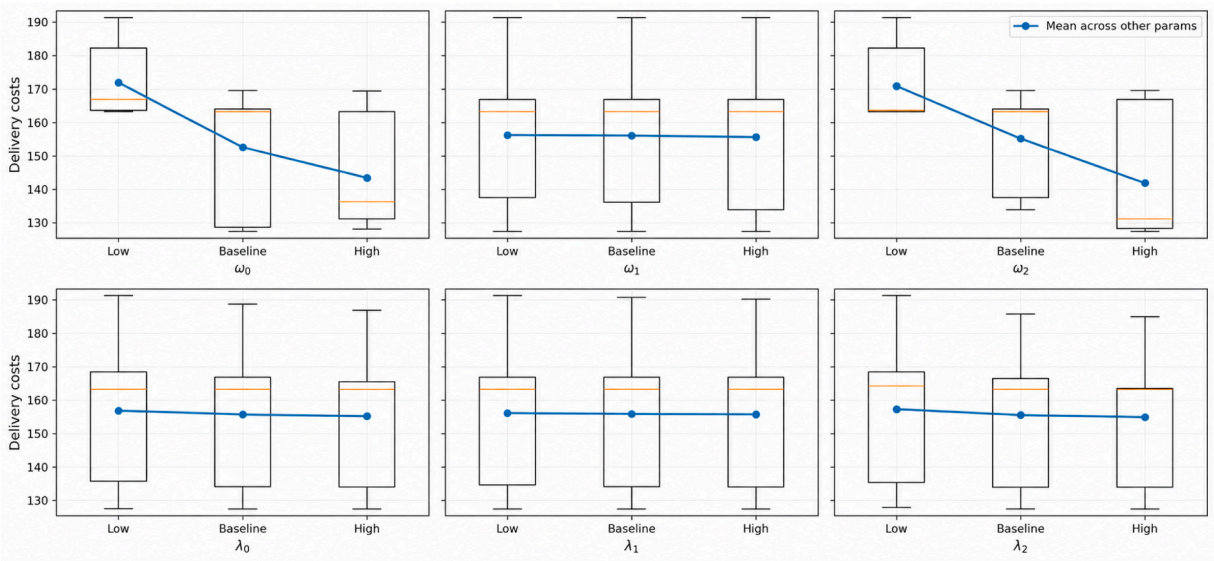


Fig. 8. Sensitivity analysis of behavioral parameters on delivery costs.

5.3. PRiME-DQN: Algorithm performance

The DQN network is trained using the Adam optimizer with a learning rate of 0.5×10^{-4}. The training process leverages GPU (mps) acceleration to speed up gradient computations, and regularization techniques are employed to prevent over-fitting. And the hyperparameter settings are given in Table 4.

It should be noted that the general DQN-related settings follow standard practices widely adopted in the deep reinforcement learning literature (Mnih et al., 2013, 2015). For the problem-specific parameters in PRiME-DQN, considering a full grid search over these reinforcement-learning hyperparameters is computationally prohibitive, we adopt a fixed empirical configuration that reflects the design logic of PRiME-DQN and apply it consistently across all computational experiments.

Specifically, the initial bias weight setting is to moderately increase the selection probability of actions in the candidate action pool during the early training stage. This value is chosen to provide sufficient guidance from the heuristic candidate pool while avoiding an overly strong bias that would prevent the agent from exploring actions outside the pool. The initial temperature provides a balanced Softmax exploration level at the beginning of training. A larger temperature would make the policy close to random and slow down convergence, whereas a very small temperature would make the policy too greedy at an early stage and increase the risk of premature convergence.

The decay rates design for the bias term and the temperature term, makes the heuristic action bias decay faster than the temperature. The rationale is that the candidate action pool should strongly guide the agent in the early stage to accelerate learning, but this heuristic intervention should diminish relatively quickly so that the learned Q-values can take over the decision process. By contrast, the temperature decays more gradually to maintain sufficient exploration even after the action bias becomes weaker, thereby reducing the risk of being trapped in local optima. The decay episodes and total training episodes are scaled with the number of customer orders $|J|$, because $|J|$ directly affects the size of the state-action space and the difficulty of the order assignment problem. Specifically, we set $E_B = E_\tau = 220 \times |J|$, $E_{\max} = 270 \times |J|$. This scaling rule allows larger instances to receive a proportionally larger training budget. The gap between the decay episodes and the total episodes also ensures that the algorithm has a sufficiently long early-stage exploration period, followed by a later-stage refinement period in which the policy becomes more exploitative and stable.

The global improvement component is assigned a stronger weight to reward actions that improve the solution relative to previous solutions, while the relative improvement component provides additional guidance toward improving the best solution found so far. This reward design helps provide informative learning signals in the combinatorial optimization environment.

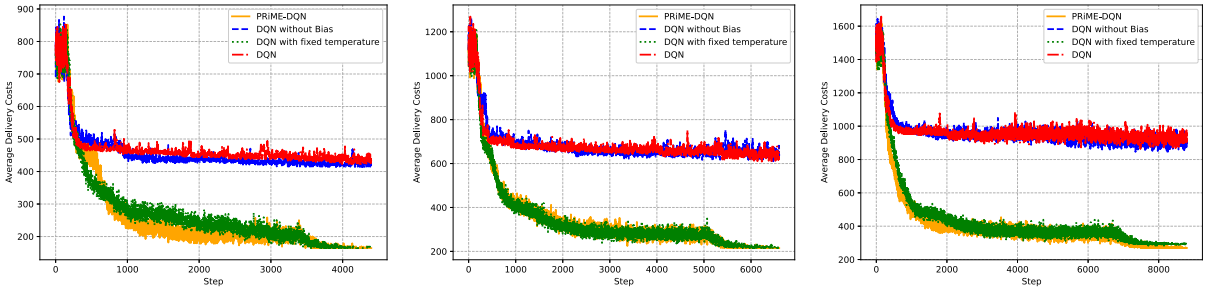
Although these problem-specific values are not claimed to be theoretically optimal, their reasonableness is supported by the ablation study in Section 5.3.1. Under the same fixed configuration, PRiME-DQN consistently outperforms the vanilla DQN and the ablation variants, indicating that the empirical settings provide stable and effective training behavior for the instances considered. In addition, the behavioral decision thresholds $threshold_s$ and $threshold_c$ are examined separately in the sensitivity analysis in Section 5.3.2, where the results show that moderate changes in these thresholds do not substantially alter delivery costs. This further indicates that the main conclusions are not driven by a single arbitrary threshold setting.

5.3.1. Ablation study

To evaluate the effectiveness of the different components embedded in PRiME-DQN, we conducted ablation studies by disabling biasing and temperature scheduling. As shown in Fig. 9, PRiME-DQN demonstrates better convergence performance compared to the

Table 4
DQN algorithm hyperparameter settings.

Hyperparameter	Description	Value
Learning Rate (η)	Controls the step size during gradient descent	5×10^{-4}
Batch Size (BS)	Number of samples per training iteration	128
Replay Memory Size ($\mathcal{M}$)	Maximum number of transitions stored in the replay buffer	10,000
Discount Factor (γ)	Balances immediate and future rewards	0.95
Initial temperature (τ_{init})	Softmax exploration parameter controlling randomness	1
Initial Bias Weight (B_{init})	Additional weight assigned to actions in the good action pool	0.2
Decay episode (E_B, E_τ)	The episode at which both Bias and Temperature begin to exhibit a declining trend	$220 \times \lvert J \rvert$
Total episode ($E_{\max}$)	Total number of episode for training	$270 \times \lvert J \rvert$
Decay rate (β_B, β_τ)	The decay rate of both bias and temperature curve	22,8
Gradient Clipping	Maximum gradient norm to prevent exploding gradients	1.0
Optimizer	Algorithm used for weight updates	Adam
Loss Function	Measures Q-value prediction error	Mean Squared Error
Relative Improvement (Δ_{best})	Measure the reward of the current action based on consistent insights	$\iota_{best}, \eta_{best} = 5, 1$
Global Improvement (Δ_{prev})	Measure the reward of the current action based on global insights	$\iota_{prev}, \eta_{prev}, = 25, 0.5$
Decision threshold ($threshold_s, threshold_c$)	Determine the acceptance of self-pickup and crowd-shipping	0.5, 0.5



(a) $|J| = 20$ (b) $|J| = 30$ (c) $|J| = 40$

Fig. 9. Ablation study: compare the convergence results among proposed PRiME-DQN algorithm and benchmark algorithms (DQN, DQN without Bias, DQN with fixed temperature).

other three variants, DQN without bias, DQN with fixed temperature ($T = 1$), and vanilla DQN. The introduction of bias has a more significant impact on both the convergence speed and the eventual stable convergence outcome of the algorithm. The incorporation of temperature regulation further facilitates the exploration of optimal solutions, particularly as the number of orders increases. For instance, when $|J| = 20$, PRiME-DQN and DQN with fixed temperature achieve similarly stable convergence after approximately 3500 steps ($175 \times |J|$), whereas the remaining two variants fail to escape suboptimal solutions within a comparable timeframe. When $|J| = 40$, it is observed that PRiME-DQN continues to explore in the later stages of training due to temperature adjustment, ultimately identifying superior solutions compared to DQN with fixed temperature. Both approaches reach stable convergence at approximately 7500 steps ($187.5 \times |J|$).

5.3.2. Sensitivity analysis of acceptance thresholds

In the main experiments, the decision thresholds for customer self-pickup acceptance and crowd-shipping acceptance are both set to 0.5, which is a commonly used default cutoff for converting predicted probabilities from binary logit models into binary decisions when no asymmetric decision cost is imposed (Hosmer et al., 2013). Under this setting, a participant is regarded as accepting the option if the predicted acceptance probability is greater than or equal to 0.5, indicating that acceptance is more likely than rejection.

To examine whether the computational results are sensitive to this threshold setting, we conduct an additional sensitivity analysis on the two acceptance thresholds. Let $threshold_s$ denote the customer self-pickup acceptance threshold and $threshold_c$ denote the crowd-shipping acceptance threshold. We vary $threshold_s$ over $\{0.4, 0.5, 0.6\}$ and $threshold_c$ over $\{0.3, 0.4, 0.5, 0.6, 0.7\}$. For each threshold combination, the experiment is repeated under five random seeds ($|J| = 20$). Fig. 10 reports the resulting delivery costs. Each panel corresponds to a fixed value of $threshold_s$, while the horizontal axis represents different values of $threshold_c$. The boxplots show the distribution of delivery costs across random seeds, and the connected markers indicate the mean delivery costs.

The results indicate that the delivery costs remain relatively stable under different threshold combinations. Although minor fluctuations are observed, there is no strong monotonic trend as either $threshold_s$ or $threshold_c$ changes within the tested ranges. This suggests that the system performance is not driven by the specific baseline threshold values of $threshold_s = 0.5$ and $threshold_c = 0.5$. The stability should be understood together with the behavioral-parameter sensitivity analysis in Section 5.2.4, and the two results are complementary. The behavioral parameters ω and λ govern the magnitude and shape of the underlying acceptance probabilities themselves, that is, how likely customers and crowd-shippers are to accept an offered option. The acceptance thresholds, by contrast, only determine where the cutoff is placed when these already-computed probabilities are converted into binary accept/reject labels.

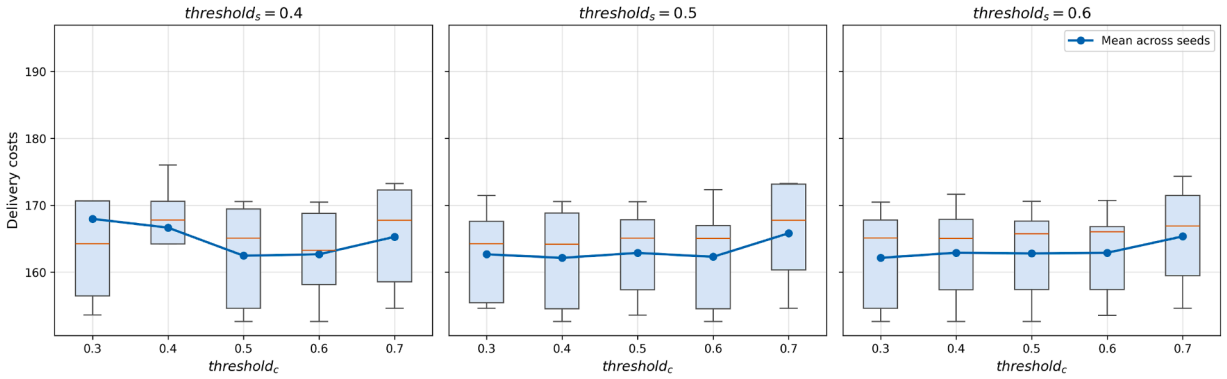


Fig. 10. Sensitivity analysis of acceptance thresholds on delivery costs.

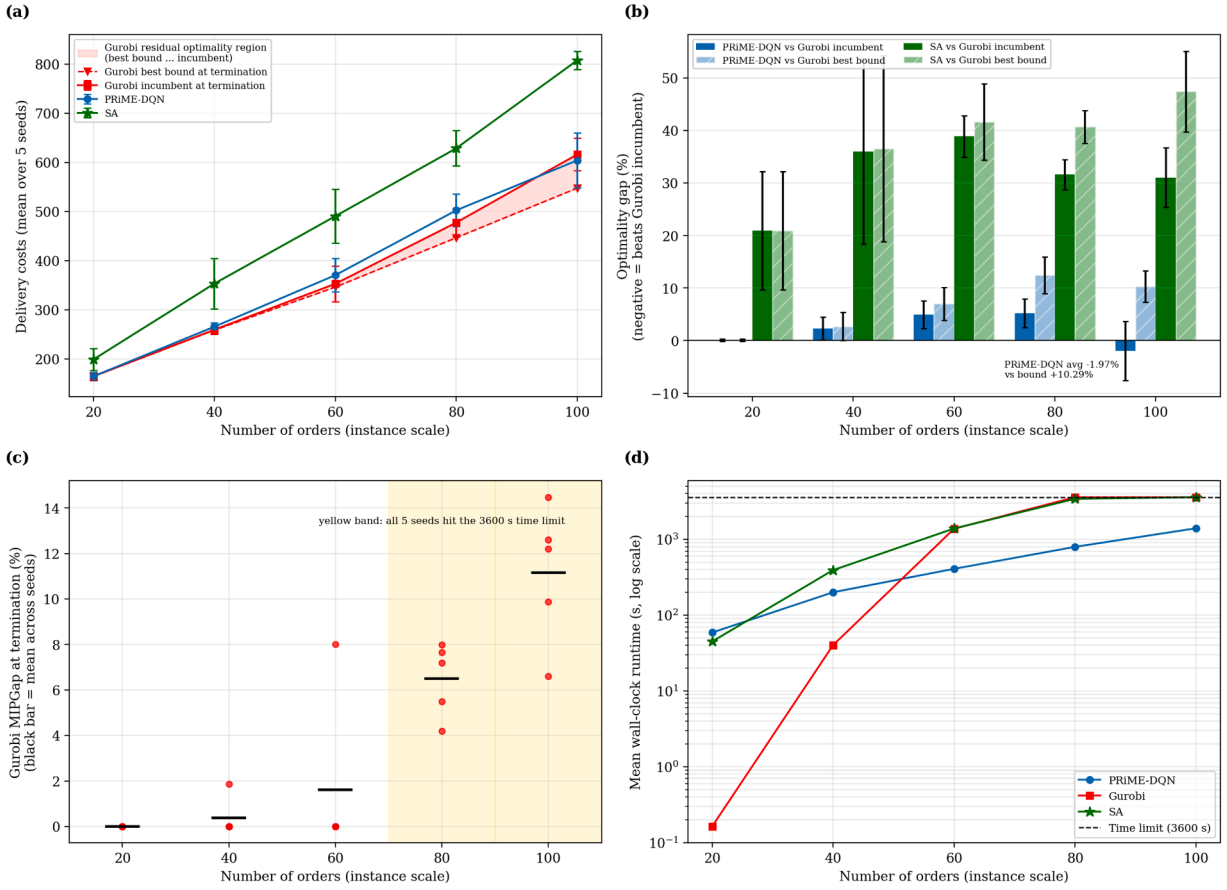


Fig. 11. Computational comparison of PRiME-DQN, Gurobi, and simulated annealing across different instance scales.

Moving the threshold within a moderate range around 0.5 only re-classifies the small set of orders whose acceptance probability lies near the cutoff, while the classification of orders whose probabilities are clearly above or below the cutoff remains unchanged. Consequently, the system performance is sensitive to the behavioral parameters that determine acceptance propensity, but robust to the specific value of the decision threshold within a reasonable range.

5.3.3. Scaling analysis

To rigorously evaluate the proposed PRiME-DQN algorithm, the experimental design is structured around two central dimensions: a comprehensive scaling analysis and a comparative performance benchmark against both the Gurobi mathematical solver and the Simulated Annealing (SA) heuristic. To ensure a fair comparison, the SA benchmark, detailed in Appendix C, uses the same candidate action pool generated by the heuristic filtering procedure, so that both SA and PRiME-DQN exploit the same problem-specific structural

information. The SA parameters are fixed across all instances and random seeds and are not tuned separately for individual cases. By systematically increasing the problem complexity from 20 shippers and 20 orders to a high-density 100x100 configuration, and validating each scenario across five independent random seeds, this setup ensures statistical robustness. This multi-faceted approach allows for a precise assessment of how solution quality, computational efficiency, and optimality gaps evolve as the system scales, highlighting the algorithm's reliability in increasingly complex logistical environments.

As shown in Fig. 11, Panel (a) reports, for each instance scale, the mean delivery cost returned by PRiME-DQN, Gurobi and SA, with error bars equal to one standard deviation across the five seeds. The red shaded band is Gurobi's *residual optimality region*: its upper edge is Gurobi's incumbent at termination and its lower edge is Gurobi's best bound at termination, so the true minimum delivery cost is guaranteed to lie inside this band. The band collapses to a line on the smaller instances (20 and 40 orders), where Gurobi proves optimality, and widens visibly from 60 orders onwards as Gurobi starts to leave the optimality gap open.

Panel (b) decomposes the optimality gap of PRiME-DQN and SA into two complementary quantities. Solid bars give $\mathrm{gap}_{\mathrm{inc}}$, the gap with respect to Gurobi's incumbent at termination; hatched bars give $\mathrm{gap}_{\mathrm{bnd}}$, the gap with respect to Gurobi's best lower bound at termination. A negative value indicates that the method achieves a lower delivery cost than Gurobi's incumbent. The only negative bar in the figure is the PRiME-DQN solid bar at 100 orders (−1.97%): on those instances PRiME-DQN returns a feasible solution whose delivery cost is roughly 1.97% below the incumbent Gurobi can produce within the time limit. The corresponding hatched bar (+10.29%) shows, however, that the same PRiME-DQN solution still sits at most 10.29% above Gurobi's best lower bound, so it is not certified as optimal. SA is positive in both metrics at every scale.

Panel (c) shows Gurobi's per-instance `MIPGap` at termination. Red dots correspond to the five individual seeds and the black tick marks give the mean. The mean gap grows monotonically with the instance scale, from 0% at 20 orders to 11.15% at 100 orders. The yellow band highlights the two scales (80 and 100 orders) at which all five seeds exhaust the 3600 s time limit without closing the optimality gap; together with panel (a), this explains why the residual optimality region expands so quickly beyond 60 orders.

Panel (d) reports the mean wall-clock runtime per scale on a logarithmic axis. The dashed black line marks the 3600 s time limit. Gurobi saturates the time limit from 80 orders onwards; SA also reaches the limit on the largest instances (mean ≈ 3429 s at 80 orders, with one seed at the cap, and exactly 3600 s for all five seeds at 100 orders), whereas PRiME-DQN finishes in ∼ 1400 s on the 100-order instances, roughly 2.6× faster than the budget granted to the other two methods.It should be noted that the gap is structural rather than a tuning artifact: as shown in Fig. 11, SA is positive in both gap_inc and gap_bnd at every scale and its disadvantage grows with instance size, whereas PRiME-DQN even improves on Gurobi's incumbent on the largest instances

6. Conclusion and future work

This paper presents a behavior-aware, hybrid last-mile delivery framework that coordinates mobile depots, crowd-shipping, customer self-pickup, and unmanned delivery under realistic behavioral uncertainty. By formulating the problem as a stochastic mixed-integer program and modeling agent behaviors through binomial logit choice models, we capture heterogeneous participant preferences and competitive interactions over limited depot resources. To efficiently solve the problem, we propose the PRiME-DQN algorithm, incorporating location filtering and action bias to enhance computational efficiency. Experimental results, based on real-world datasets from Seattle, USA, demonstrate that the hybrid delivery model significantly outperforms single-method approaches (self-pickup and CDS), achieving cost reductions of approximately 5.9% and 11.3%, respectively. Additionally, PRiME-DQN attains near-optimal solutions compared with Gurobi's, while requiring less computational time compared to Gurobi.

While our current model focuses on mode selection and behavioral decision-making, it does not explicitly integrate routing optimization for rejected orders, which could be consolidated and served via a centralized fallback plan. Addressing this would require modeling dynamic routing over residual demand scenarios, which would significantly increase model complexity and is therefore left for future work. Nevertheless, the existing structure allows for modular extension by treating rejected orders as a separate scenario and embedding cost evaluation through external routing algorithms or modifying the reward structure within the DQN. This highlights an important direction for future research that combines behavior-aware mode allocation with adaptive routing under rejection risk.

In addition, although the behavioral parameters are calibrated from prior studies and sensitivity analyses indicate that the main conclusions remain robust, future research could further enhance the framework by estimating these parameters from context-specific empirical data and validating the model across additional geographical and operational settings.

CRediT authorship contribution statement

Shixuan Hou: Writing – original draft, Software, Methodology, Data curation, Conceptualization; **Chengming Hu:** Software; **Annabathuni Sandeep Chowdary:** Software, Data curation; **Bissan Ghaddar:** Writing – review & editing, Supervision, Funding acquisition; **Joe Naoum-Sawaya:** Writing – review & editing, Supervision, Funding acquisition; **Jie Gao:** Writing – review & editing, Validation, Data curation.

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.

Data availability

Data will be made available on request.

Appendix A. Single order delivery model: self-service

Given the set of orders J, and set of locations K, the objective of the optimization model is to minimize the sum of mobile depots allocation costs, order postponed costs, and expected order delivery costs of self-service.

$$\min \quad F(z,y) + \mathbb{E}[G(w)] \tag{A.1}$$

$$\text{s.t. } y_j + \sum_{\forall k \in K} w_{j,k} = 1, \quad \forall j \in J \tag{A.2}$$

$$\sum_{\forall j \in J} w_{j,k} \le Cap * z_k, \quad \forall k \in K \tag{A.3}$$

$$y_j, w_{j,k} \in \{0,1\}, \quad \begin{cases} \forall j \in J \\ \forall k \in K \end{cases} \tag{A.4}$$

$$z_k \in \mathbb{N}, \quad \forall k \in K, \tag{A.5}$$

where,

$$F(z,y) = \sum_{\forall k \in K} C_k z_k + \sum_{\forall j \in J} H y_j \tag{A.6}$$

$$\mathbb{E}[G(w)] = \sum_{\forall j \in J} \sum_{\forall k \in K} p^{r_s}_{j,k} C_{j,k} w_{j,k} \tag{A.7}$$

Eq. (A.6) define the mobile depots allocation costs, and order postponed costs. Eq. (A.7) define the expected delivery costs of self-service orders. Eq. (A.2) ensure that each order can either be postponed or be fulfilled by customer self-service. Eq. (A.3) ensure that the number of orders allocated to each location can not exceed the capacity of each mobile depot. Eq (A.4) and Eq (A.5) define the decision variables.

Appendix B. Single order delivery model: crowd-sourced delivery

Given the set of crowd-shippers I, set of orders J, and set of locations K, the objective of the optimization model is to minimize the sum of mobile depots allocation costs, order postponed costs, expected order delivery costs by crowd-shippers.

$$\min \quad F(z,y) + \mathbb{E}[Q(x)] \tag{B.1}$$

$$\text{s.t. } y_j + \sum_{\forall i \in I} \sum_{\forall k \in K} x_{i,j,k} = 1, \quad \forall j \in J \tag{B.2}$$

$$\sum_{\forall i \in I} \sum_{\forall j \in J} x_{i,j,k} \le Cap * z_k, \quad \forall k \in K \tag{B.3}$$

$$\sum_{\forall j \in J} \sum_{\forall k \in K} x_{i,j,k} \le 1, \quad \forall i \in I \tag{B.4}$$

$$y_j, x_{i,j,k} \in \{0,1\}, \quad \begin{cases} \forall i \in I \\ \forall j \in J \\ \forall k \in K \end{cases} \tag{B.5}$$

$$z_k \in \mathbb{N}, \quad \forall k \in K, \tag{B.6}$$

where,

$$F(z,y) = \sum_{\forall k \in K} C_k z_k + \sum_{\forall j \in J} H y_j \tag{B.7}$$

$$\mathbb{E}[Q(x)] = \sum_{\forall i \in I} \sum_{\forall j \in J} \sum_{\forall k \in K} (p^{a_c}_{i,j,k} C_{i,j,k} x_{i,j,k} + p^{r_c}_{i,j,k} C_{j,k} x_{i,j,k}) \tag{B.8}$$

Eq. (B.7) define the mobile depots allocation costs, and order postponed costs. Eq. (B.8) define the expected delivery costs of crowd-shipping. Eq. (B.2) ensure that each order can either be postponed or fulfilled by crowd-shippers. Eq. (B.3) ensure that the number of orders allocated to each location can not exceed the capacity of each mobile depot. Eq. (B.4) ensure that each crowd-shipper can only deliver at most one order from one mobile depot. Eq. (B.5) and Eq (B.6) define the decision variables.

Algorithm 3 Simulated Annealing benchmark.

Input: Number of orders $|J|$, action space $\mathcal{A}$, candidate action pool $\mathcal{G}$, maximum iterations $N_{\max}$, initial temperature T_{start}, final temperature T_{end}, initial bias probability p_b^0, time limit $T_{\text{limit}} = 3600$ seconds
Randomly initialize an order-level action vector $\mathbf{a}$, where $a_j \in \mathcal{A}$ for each order $j \in J$
Evaluate the initial delivery cost $C(\mathbf{a})$ using the same cost-evaluation function as PRiME-DQN
Set $\mathbf{a}^* \leftarrow \mathbf{a}$ and $C^* \leftarrow C(\mathbf{a})$
Record the starting time t_0
for $n = 1, \ldots, N_{\max}$ **do**
  if elapsed time since t_0 exceeds T_{limit} **then**
    break
  end if
  Update temperature:

$$T_n = T_{\text{start}} \left(\frac{T_{\text{end}}}{T_{\text{start}}} \right)^{n/N_{\max}}$$

  Update bias probability:

$$p_b(n) = p_b^0 \left(1 - \frac{n}{N_{\max}} \right)$$

  Create a neighboring solution $\mathbf{a}' \leftarrow \mathbf{a}$
  if $n \le 0.8 N_{\max}$ **then**
    Randomly select the number of changed orders m from $\{1, 2\}$
  else
    Set the number of changed orders $m = 1$
  end if
  Randomly select m orders from J
  for each selected order j **do**
    if random number $< p_b(n)$ and $\mathcal{G} \neq \emptyset$ **then**
      Select a new action a_j' randomly from the candidate action pool $\mathcal{G}$
    else
      Select a new action a_j' randomly from the full action space $\mathcal{A}$
    end if
  end for
  Evaluate the delivery cost $C(\mathbf{a}')$ using the same cost-evaluation function as PRiME-DQN
  Compute $\Delta = C(\mathbf{a}') - C(\mathbf{a})$
  if $\Delta < 0$ **then**
    Accept the neighboring solution: $\mathbf{a} \leftarrow \mathbf{a}'$
  else
    Accept $\mathbf{a}'$ with probability $\exp(-\Delta / T_n)$
  end if
  if $C(\mathbf{a}) < C^*$ **then**
    Update $\mathbf{a}^* \leftarrow \mathbf{a}$ and $C^* \leftarrow C(\mathbf{a})$
  end if
end for
Output: Best solution $\mathbf{a}^*$ and best delivery cost C^*

Appendix C. Simulated Annealing benchmark

This appendix provides the implementation details of the Simulated Annealing (SA) benchmark used in the computational comparison. The SA benchmark uses the same order-level action representation and cost-evaluation function as PRiME-DQN. Specifically, a candidate solution is represented by an action vector $\mathbf{a}$, where each element a_j denotes the dispatch decision for order j, including postponement, self-pickup, or crowd-shipping assignment. Given an action vector, the delivery cost is evaluated using the same cost-computation module as PRiME-DQN, including mobile-depot allocation, order dispatch, crowd-shipper matching, postponement, and backup drone delivery costs.

Unlike PRiME-DQN, SA does not learn Q-values and therefore cannot use the Q-value-based Softmax action-selection rule. Instead, SA explores the same action space through local perturbations and accepts candidate solutions according to the Metropolis criterion. To ensure a fair comparison, SA is evaluated on the same problem instances, under the same random seeds, using the same objective function, and with the same 3600-second computational time limit. The key SA parameters in Table C.1 were determined through small-scale pilot runs to ensure a stable cooling schedule, then fixed and applied uniformly across all instances and seeds without

instance-level retuning, exactly the same protocol as PRiME-DQN, whose settings were likewise fixed once and applied consistently. Both methods thus operate under a single non-instance-specific configuration, so the comparison reflects intrinsic performance rather than tuning effort, and the detailed procedure is presented in Algorithm 3.

Table C.1
Parameter settings of the Simulated Annealing benchmark.

Parameter	Value	Rationale
$N_{\max}$	20,000	Sufficient search steps within the time limit
T_{start}	300	Encourages early exploration by accepting worse moves
T_{end}	10^{-3}	Makes the final search nearly greedy
p_b^0	0.3	Uses candidate action pool without fully restricting exploration
T_{limit}	3600 seconds	Same computational time limit as the benchmark comparison
Neighborhood size	$m \in \{1,2\}$ if $n \leq 0.8N_{\max}$; $m = 1$ otherwise	Early exploration and later local refinement

References

Beck, K., Esquillor, J., Zarei, M.M., Froes, I., Hauswald, I., Giannakopoulou, A., Flämig, H., 2025. Making last mile logistics models aware of customer choices, demand sustainability and data economy. Eur. Transp. Res. Rev. 17 (1), 1–14.

Boysen, N., Fedtke, S., Schwerdfeger, S., 2021. Last-mile delivery concepts: a survey from an operational research perspective. Or Spectrum 43 (1), 1–58.

Chen, W., Park, S., Tanneau, M., Van Hentenryck, P., 2022. Learning optimization proxies for large-scale security-constrained economic dispatch. Electr. Power Syst. Res. 213, 108566.

Cheng, C.-A., Kolobov, A., Swaminathan, A., 2021. Heuristic-guided reinforcement learning. Adv. Neural Inf. Process. Syst. 34, 13550–13563.

Deutsch, Y., Golany, B., A parcel locker network as a solution to the logistics last mile problem, Int. J. Prod. Res. 56, (2018) 251—261.

Dos Santos, A.G., Viana, A., Pedroso, J.P., 2022. 2-Echelon lastmile delivery with lockers and occasional couriers. Transp. Res. E: Logist. Transp. Rev. 162, 102714.

Eysenbach, B., Geist, M., Levine, S., Salakhutdinov, R., 2023. A connection between one-step RL and critic regularization in reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 9485–9507.

Fan, J., Yao, X., Zhou, L., Wood, J., Wang, C., 2022. Food-delivery behavior under crowd sourcing mobility services. J. Traffic Transp. Eng. 9 (4), 676–691.

Gdowska, K., Viana, A., Pedroso, J.P., 2018. Stochastic last-mile delivery with crowdshipping. Transp. Res. Procedia 30, 90–100.

Ghaderi, H., Zhang, L., Tsai, P.-W., Woo, J., 2022. Crowdsourced last-mile delivery with parcel lockers. Int. J. Prod. Econ. 251, 108549.

Ghannadpour, S.F., Noori, S., Tavakkoli-Moghaddam, R., 2014. A multi-objective vehicle routing and scheduling problem with uncertainty in customers' request and priority. J. Comb. Optim. 28, 414–446.

Hosmer, D.W., Lemeshow, S., Sturdivant, R.X., 2013. Applied Logistic Regression. John Wiley & Sons. 3rd ed.

Hou, S., Gao, J., Wang, C., 2022. Optimization framework for crowd-sourced delivery services with the consideration of shippers' acceptance uncertainties. IEEE Trans. Intell. Transp. Syst. 24 (1), 684–693.

Hou, S., Wang, C., Gao, J., 2025. Reinforced stable matching for crowd-sourced delivery systems under stochastic driver acceptance behavior. Transp. Res. C: Emerg. Technol. 170, 104916.

Joerss, M., Schröder, J., Neuhaus, F., Klink, C., Mann, F., 2016. Parcel delivery: the future of last mile. McKinsey Co. , 1–32.

Liang, X., Wang, N., Zhang, M., Jiang, B., 2023. Bi-objective multi-period vehicle routing for perishable goods delivery considering customer satisfaction. Expert Syst. Appl. 220, 119712.

Macrina, G., Pugliese, L. D.P., Guerriero, F., Laporte, G., 2020. Crowd-shipping with time windows and transshipment nodes. Comput. Oper. Res. 113, 104806.

McFadden, D., 1981. Econometric models of probabilistic choice. Struct. Anal. Discrete Data Econom. Appl. 198272.

Merchán, D., Arora, J., Pachon, J., Konduri, K., Winkenbach, M., Parks, S., Noszek, J., 2024. 2021 Amazon last mile routing research challenge: data set. Transp. Sci. 58 (1), 8–11.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv:1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Mohri, S.S., Ghaderi, H., Nassir, N., Thompson, R.G., 2023. Crowdshipping for sustainable urban logistics: a systematic review of the literature. Transp. Res. E: Logist. Transp. Rev. 178, 103289.

Mohri, S.S., Ghaderi, H., Van Woensel, T., Mohammadi, M., Nassir, N., Thompson, R.G., 2024. Contextualizing alternative delivery points in last mile delivery. Transp. Res. E: Logist. Transp. Rev. 192, 103787.

Moroz, M., Polkowski, Z., 2016. The last mile issue and urban logistics: choosing parcel machines in the context of the ecological attitudes of the y generation consumers purchasing online. Transp. Res. Procedia 16, 378–393.

Mousavi, K., Bodur, M., Roorda, M.J., 2022. Stochastic last-mile delivery with crowd-shipping and mobile depots. Transp. Sci. 56 (3), 612–630.

Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery. Transp. Res. C: Emerg. Technol. 54, 86–109.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey. Networks 72 (4), 411–458.

Peppel, M., Spinler, S., Winkenbach, M., 2024. Integrating mobile parcel lockers into last-mile delivery networks: an operational design for home delivery, stationary, and mobile parcel lockers. Int. J. Phys. Distrib. Logist. Manag. 54 (4), 418–447.

Punel, A., Ermagun, A., Stathopoulos, A., Studying determinants of crowd-shipping use, Travel Behav. Soc. 12, (2018) 30–40.

Sajid, M., Mittal, H., Pare, S., Prasad, M., 2022. Routing and scheduling optimization for UAV assisted delivery system: a hybrid approach. Appl. Soft Comput. 126, 109225.

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.

Stokkink, P., Geroliminis, N., 2023. A continuum approximation approach to the depot location problem in a crowd-shipping system. Transp. Res. E: Logist. Transp. Rev. 176, 103207.

Sungur, I., Ren, Y., Ordóñez, F., Dessouky, M., Zhong, H., 2010. A model and algorithm for the courier delivery problem with uncertainty. Transp. Sci. 44 (2), 193–205.

Sutton, R.S., Barto, A.G., et al., 1998. Reinforcement Learning: An Introduction. Vol. 1. MIT press Cambridge.

Wang, X., Wang, L., Wang, S., Pan, J., Ren, H., Zheng, J., 2022. Recommending-and-grabbing: a crowdsourcing-based order allocation pattern for on-demand food delivery. IEEE Trans. Intell. Transp. Syst. 24 (1), 838–853.

Wang, Y., Bi, M., Chen, Y., 2020. A scheduling strategy of mobile parcel lockers for the last mile delivery problem. Promet-Traffic&Transp. 32 (6), 875–885.

Watkins, C.J., Dayan, P., 1992. Q-Learning. Mach. Learn. 8 (3), 279–292.

Wen, J., Li, Y., 2016. Vehicle routing optimization of urban distribution with self-pick-up lockers. In: 2016 International Conference on Logistics, Informatics and Service Sciences (LISS). IEEE, pp. 1–6.

Wu, Y., Zeng, B., Huang, S., 2019. A dynamic strategy for home pick-up service with uncertain customer requests and its implementation. Sustainability 11 (7), 2060.
Yan, H., Tan, H., Wang, H., Zhang, D., Yang, Y., 2024. Robust route planning under uncertain pickup requests for last-mile delivery. In: Proceedings of the ACM Web Conference 2024, pp. 3022–3030.
Yang, J., Lau, H.C., Wang, H., 2024. Optimization of customer service and driver dispatch areas for on-demand food delivery. Transp. Res. C: Emerg. Technol. 165, 104653.
Yuen, K.F., Wang, X., Ng, L. T.W., Wong, Y.D., 2018. An investigation of customers' intention to use self-collection services for last-mile delivery. Transp. Policy 66, 1–8.
Zhang, S., Le, T.V., Roorda, M., 2023a. Optimizing parcel locker locations in a city crowd logistics network. Transp. Res. Rec. 2677 (12), 267–283.
Zhang, W., Xu, M., Wang, S., 2023b. Joint location and pricing optimization of self-service in urban logistics considering customers' choice behavior. Transp. Res. E: Logist. Transp. Rev. 174, 103128.
Zhou, M., Zhao, L., Kong, N., Campy, K.S., Xu, G., Zhu, G., Cao, X., Wang, S., 2020. Understanding consumers' behavior to adopt self-service parcel services for last-mile delivery. J. Retail. Consum. Serv. 52, 101911.