International Journal of Scientific Engineering and Science
Volume 10, Issue 5, pp. 130-134, 2026. ISSN (Online): 2456-7361

Real-Time Delivery Dispatching in Dense Urban Environments: A Deep Reinforcement Learning Approach

Duyen Bui Thi[1], Duong Vu Bui Thuy[2], Huyen Ho Minh[3]
[1,2,3]Thuong mai University, Ha Noi, Viet Nam

***Abstract*—*Coordination in delivery operations has been a topic of interest in recent years thanks to the development of e-commerce and logistics, as coordination is one of the factors that helps businesses improve operational efficiency and optimize their resources. Therefore, businesses need the most optimal coordination solutions. This paper addresses the Random-Real-Time Delivery Driver Dispatching Problem (R-RDP). This problem has high practical applicability, especially in urban delivery with high traffic density and a large volume of orders, which requires strong flexibility and adaptability in the coordination process. This study proposes intelligent dispatching methods based on reinforcement learning algorithms to solve the R-RDP problem - Proximal Policy Optimization (PPO) to build the most optimal order dispatch schedule for the driver team. The experimental results show that reinforcement learning is superior and achieves significant efficiency in solving the R-RDP problem.*

Keywords*—*Deep Reinforcement Learning, Delivery Driver Dispatching, Proximal Policy Optimization, Last-mile Delivery.

I. INTRODUCTION

In the context of urbanization and the boom in the logistics industry, the problem of coordinating orders and optimizing schedules in real time in densely populated cities is facing huge challenges. Traditional scheduling methods based on fixed rules often reveal many limitations, failing to adapt quickly to the random fluctuations of customer demand as well as urban traffic congestion [1]. To solve this problem, the shift to Deep Reinforcement Learning (DRL) application combined with spatio-temporal graph structures is increasingly proving its superior effectiveness, helping the system make optimal routing decisions even in highly uncertain environments [1].

To improve operational efficiency, modern research has focused on building Markov decision process models to optimize routes and increase economic benefits for last-mile delivery drivers [2]. In addition, the application of neural network structures to process asynchronous dynamic vehicle dispatch decisions helps the system respond at super speed and minimize order delays [3]. These advanced approaches not only thoroughly solve the resource allocation problem but also open up intelligent, comprehensive dispatch solutions for today's complex urban transport systems [2, 3].

This research has three main contributions as follows:

- Researching an extended real-time delivery driver dispatching problem (R-RDP) model by integrating random factors arising in a real-world operating environment, which is called R-RDP.
- Applying PPO to find the optimal solution to the problem.
- Conduct an empirical assessment in Hanoi, a typical urban area with complex transportation and logistics. Analyzing the results comparing reinforcement learning to demonstrate the superior practical effectiveness of RL.

The paper is structured as follows. The following section presents an overview of related works. Section 3 provides a detailed description of the real-time delivery driver dispatching model, R-RDP. Next, the proposed algorithm, dataset and experimental results in Hanoi are presented in Sections 4, 5 and 6, respectively. Finally, Section 7 provides conclusions, recommendations, and directions for future research.

II. RELATED WORKS

Numerous impressive studies have addressed issues related to the development and application of reinforcement learning models for optimizing dispatching in the real-time delivery field.

Traditional mathematical programming and hyper-heuristic methods often face a significant trade-off between computation time and solution quality. To address the randomness of urban demand, Zamal et al. (2025) proposed an SDOA-DP model combining Cost Function Approximation (CFA) and Adaptive Large Neighborhood Search (ALNS) methods, which reduced actual operating costs by up to 30.5% [4]. In more complex logistics spaces, Liu et al. (2026) applied a mixed-integer second-order cone programming (MISOCP) model coordinating electric vertical takeoff and landing (eVTOL) mother ships and drones to avoid no-fly zones, while using vision graphs to refine trajectories [5]. Despite their high accuracy, these algorithms often take from tens of minutes to hours to compute, failing to meet the sub-second decision-making needs in an extremely dynamic urban environment.

To meet real-time operating conditions, building the problem in the form of a Markov Decision Process (MDP) solved using DRL is becoming the main trend. To encode urban spatial structure, graph neural networks (GNNs) are often integrated into DRLs. Li et al. (2026) demonstrated that a GNN-DRL model using a softmax discovery strategy reduces scheduling time from tens of minutes to just a few seconds [6]. Saleh et al. (2025) used the DQN (Deep Q-Network) algorithm to dynamically adjust fixed driver shifts to absorb sudden demand spikes [7].

For asynchronous events, Cordeiro and Pitombeira-Neto (2025) proposed a semi-Markov model (SMDP) with two independent Double DQN agents handling order generation and

International Journal of Scientific Engineering and Science
Volume 10, Issue 5, pp. 130-134, 2026. ISSN (Online): 2456-7361

vehicle release events, reducing waiting time by 50.6% [3]. In addition, Dehghan et al. (2026) applied the NeurADP framework (combining ADP and DRL) to solve the ultra-fast delivery problem through a flexible order consolidation and queuing mechanism [8].

When the system scale reaches thousands of drivers and orders simultaneously, the multi-agent model (MARL) is applied to increase dispersion and cooperation. Sha et al. (2026) introduced DualG-MARL using dual graphs (vehicle state and task) combined with a Top-K filter to optimize order response rate (ORR) [9]. To manage mixed tasks (delivery, pickup, service), Lyu et al.'s HeroCS system (2025) integrates a top-k action pruning mechanism based on distance and hides state-aware actions to ensure performance and fairness among drivers [10].

In terms of core algorithm selection, Saleh et al. (2025) compared PPO, Advantage Actor Critic (A2C), and DQN, showing that PPO with the pruning objective function achieved the highest efficiency in optimizing shifts [11]. To solve the multi-objective problem, Wang et al. (2025) combined Inverse Optimization (IO) and PPO, demonstrating that an active order holding strategy during low-density periods will create better order consolidation opportunities [12]. Finally, to protect privacy and reduce communication burden, the Federated Multi-Agent Deep Reinforcement Learning (FedMARL4OD) system (2025) combines local MARL with global Federated Learning, which increases drivers' cumulative earnings by 9.17% [13].

III. THE R-RDP PROBLEM

The random real-time delivery dispatching problem (R-RDP) is defined as the task of efficiently allocating drivers operating in an urban area to continuously generated orders over time, with the goal of maximizing operational efficiency and service quality.

The R-RDP problem consists of three main components:

- Drivers: Each driver has a status that includes their current location, carrying capacity, current schedule, and level of readiness to take on new assignments.
- Orders: Each order includes a delivery location, requested delivery time, desired delivery timeframe, weight, and other relevant details.
- Environment state: This represents dynamic environmental factors influencing dispatch decisions, including real-time traffic information, order density by area, weather, and the current allocation of drivers by region. To simulate Hanoi city, we integrated these dynamic factors using real-world data collected from the map API at a specific point in time. This data provides real-time travel times between pairs of points within the city, accurately reflecting the impact of the aforementioned dynamic factors on the dispatch decision-making process.

The problem is described as follows: At each time *t,* the system receives a set of orders, each order including information about the recipient's location, the current location of the order, the desired delivery time, and the weight. Simultaneously, the system also tracks the operational status of the delivery drivers, such as their current location and maximum load capacity. A key feature of R-RDP is its ability to handle new orders randomly generated while drivers are delivering their initial orders. When a random order is generated, the order assignment process follows this principle:

- If one or more drivers are available (no orders on their schedule), the order will be assigned to the driver with the shortest travel time from their current location to the pickup point.
- If drivers are busy (all drivers are on schedule), the system will insert new orders at feasible locations along the route. The optimal insertion location (e.g., between existing pickup/delivery points or at the end of the route) is determined by recalculating the total travel time and rearranging the route to ensure the best possible solution.

The driver pickup and delivery process is illustrated in Figure 1 below.

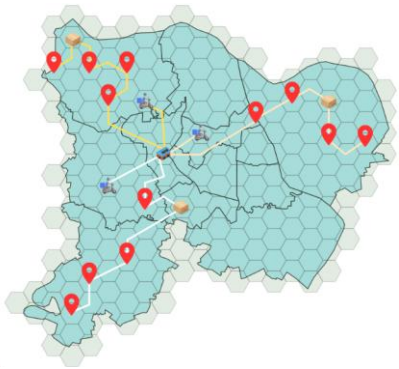

Fig. 1. Simulating the delivery driver dispatch process.

Before delving into the study of R-RDP, we consider the following symbols in Table 1:

TABLE I. Mathematical symbols.

S. No.	Symbol	Explanation
1	D	The set of drivers, $d \epsilon D$
2	O	The set of orders, $o \epsilon O$
3	x_d^o	Driver d is assigned to the order o with $x_d^o = 1$: The driver accepts the order, $x_d^o = 0$: The driver did not accept the order
4	t_d^o	Order o processing time by driver d
5	t_o	Order completion time o
6	r_o	The time when the order o was placed
7	l_o	Deadline for completing the order o
8	w_o	The weight/volume of the order o
9	C_d	Driver's maximum carrying capacity d
10	n	Total orders under responsibility
11	n'	The order has not been delivered

The goal of the problem is to minimize the total delivery time:

$$f(Obj) \rightarrow Min[\sum_{d=1}^{n} \sum_{o=1}^{m} t_d^o] \quad (1)$$

The problem must satisfy the following constraints:

$$t_d^o \leq l_o \quad \forall d \epsilon D, o \epsilon O \quad (2)$$

$$i = \sum_{i=1}^{n} w_o . x_d^o \leq C_d \quad \forall d \epsilon D, x_d^o = 1 \quad (3)$$

International Journal of Scientific Engineering and Science
Volume 10, Issue 5, pp. 130-134, 2026. ISSN (Online): 2456-7361

$\exists: x_d^o = 1$	$\forall o \in O, \forall d \in D$	(4)
$w_o + \sum_{d \in Od}^{n'} w_o . x_d^{o'} \leq C_d$	$\forall o, o' \in O_d, \forall d \in D$	(5)
$t_d^o + t_d^{o'} \leq l_o$	$\forall o, o' \in O_d, \forall d \in D$	(6)

The specific details of the constraints are as follows:

- Constraint (2): The order o completion time must be less than or equal to the latest allowable delivery deadline l_o.
- Constraint (3): Each driver has a maximum load limit; the total weight of all orders currently assigned to driver d must not exceed the maximum carrying capacity C_d.
- Constraint (4): Each order o can only be assigned to one driver d.
- Constraint (5): The driver's current load d plus the weight of the newly generated order must be less than or equal to the maximum load C_d.
- Constraint (6): After adding the new order to driver d route, the delivery completion time must be earlier than the expected late delivery deadline l_o for all original orders and the random order added to the new route.

IV. Proposed Solution

A. Solution

In this study, the solution to the R-RDP problem involves assigning orders to drivers and determining the sequence of execution. Each dispatch schedule is represented by a vector whose size equals the total number of orders to be served. The value of each element in this vector indicates which driver and the corresponding order sequence will be processed.

Specifically, the solution is defined by two main components:

- Driver - Order Mapping: Identify which driver $d \in D$ is responsible for order $o \in O$.
- Route Sequence: Determine the order of service points (pickup and delivery) that each driver d follows.

Example: Consider a delivery job with 13 orders, handled by 3 drivers. Thus, we have:

- The set of orders O = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}, where orders 11, 12, and 13 are randomly generated orders.
- The set of drivers D = {1, 2, 3}

As shown in Table 2, each order in set O is assigned to a specific driver in set D:

TABLE II. Drivers – Orders assignment.

Orders	1	2	3	4	5	6	7	8	9	10	11	12	13
Drivers	1	1	2	3	3	1	2	3	2	3	1	3	2

The chart shows the assigned orders and delivery times of drivers before and after the order was generated as described in Figure 2:

B. Proposed algorithm

We propose an improved PPO (Proximal Policy Optimization with Coordination) algorithm, consisting of four main components:

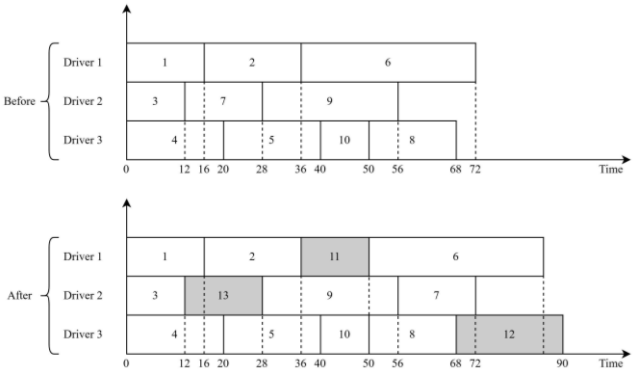

Fig. 2. A Gantt chart showing the scheduling.

First, modeling the problem as an MDP. State s_t is a fixed vector combining order information (weight, remaining time), system-wide statistics (average load, time difference between drivers), and the load of each driver. Action $a_t \in \{0,1,\dots,m\}$ where 0 means skip, $i \geq 1$ is assigned to driver i. The reward function is defined:

$$R = \begin{cases} 500 + \dfrac{t_o^{dead} - t_{arrival}}{60} \\ 200 \\ -20 \\ -50 \end{cases}$$

Secondly, there's the action masking and guided fallback mechanism. Before each decision, the algorithm identifies a pool of feasible drivers based on two constraints: sufficient remaining payload to accommodate the order and the expected delivery time not exceeding the deadline:

$$D_{feas} = \{ d \in D : w_d + w_o \leq W_{d,}\ t_{arrival}(d,o) \leq t_o^{dead} \}$$

Actions outside of D_{feas} are masked. If the Actor's action is still not feasible, the guided fallback mechanism automatically assigns the order to the driver with the earliest completion time:

$$d^* = argmin_{d \in D_{feas}} t_{arrival}(d,o)$$

This mechanism ensures that no orders are missed due to non-optimal policies during training.

Thirdly, the Actor-Critic network is trained with a clip loss function. The objective function of the PPO is:

$$L^{CLIP}(\theta) = E_t[r_t(\theta)\hat{A}_t, clip\ (r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t]$$

Where $\pi_t(\theta) = \dfrac{\pi_\theta(at|st)}{\pi_{\theta_{old}}(at|st)}$. The advantage $\hat{A}_t$ is calculated using GAE ($\gamma = 0.99, \lambda = 0.95$). The total loss function adds critic loss and entropy bonus to encourage exploration:

$$L = -L^{CLIP} + 0{,}5. \| V_\phi(s_t) - G_t \|^2 - 0{,}05. H(\pi_\theta(.\,t|st))$$

Parameter updates occur after each episode or after the buffer has reached its maximum number of steps.

Fourth, the post-processing unit is two-phase. The first phase (batch insertion) combines all early-shift orders into a single pickup, then inserts each newly generated order into the optimal position. The second phase (global repair) examines each delayed order and attempts to transfer it to another driver; the transfer is accepted if $\Delta_{late} < 0$, meaning the total number of delayed orders is reduced.

V. Dataset

To conduct the experiment, we constructed a self-generated

International Journal of Scientific Engineering and Science
Volume 10, Issue 5, pp. 130-134, 2026. ISSN (Online): 2456-7361

dataset with a structure suitable for the R-RDP problem, detailed in Figure 3 and Figure 4. The dataset was designed to realistically simulate the operational characteristics of an urban delivery system under real-world conditions, especially the complex, high-density, and unpredictable traffic environment of Hanoi's city center. Specifically, using QGIS software, the study area was partitioned into 228 uniformly sized hexagonal grid cells, each representing a delivery zone, simplifying spatial coordinate representation while maintaining inter-regional geographical structure. The distance between cell centers, both horizontally and vertically, was set at 1.5 km, corresponding to the scale of residential areas and urban routes, as illustrated in Figures 3 and 4. Each cell was assigned a unique ID for dispatch purposes.

Fig. 3. Hexagonal grid of Hanoi city center.

Based on this partitioning, we applied the OpenRouteService API in combination with Python and real-world map data of the central Hanoi area to construct a 228 × 228 travel time estimation matrix, representing the actual travel time between each pair of grid cells (Figure 4). This matrix reflects the influence of the urban road network, rather than assuming simple Euclidean distances, thus enhancing the realism of the simulation. During the experiment, the travel time estimation matrix played a crucial role in evaluating the feasibility of routes and the effectiveness of the dispatching algorithm.

	H1	**H2**	**…**	**H228**
H1	0	24.919	…	31.523
H2	24.338	0	…	30.929
…	…	…	0	…
H228	31.236	31.195	…	0

Fig. 4. Travel time estimation matrix.

The driver dataset structure includes a unique driver ID (Driver_id), the driver's departure time to the pickup point (Timestamp), the driver's maximum load (Maximum_load) in kilograms, and the current location (Current_place).

For order data, each order has a unique order ID (Order_id), the time it was created (Created_time), the origin cell (Origin_cell), and the delivery destination (Destination_cell). Additionally, each order has a weight in kilograms (Weight) and a latest delivery time (Latest_time).

The study used ten experimental datasets (DO1 to DO10) with different scales to evaluate the stability and scalability of the algorithms under different conditions.

VI. RESULT

Experimental results show that a characteristic of PPO is its ability to maintain a narrow gap between the best and average values across episodes. In dataset DO4, at episode 600 (442.07 → 449.72 minutes) and in dataset DO7 at episode 900 (376.38 → 383.98 minutes), the average delivery time did not differ significantly from the best delivery time. Conversely, the variation in results during the initial training phase was relatively large in some datasets, as evidenced by the high standard deviation at DO6 (39.46 at episode 100) and DO8 (38.25 at episode 100), indicating that the model had difficulty stabilizing parameters during the initial phase.

To evaluate the effectiveness of the PPO algorithm in real-time delivery scheduling, the study trained the model through multiple episodes and monitored metrics such as reward, order assignment rate, late delivery rate, makespan, and throughput. The experimental results are presented in the graphs below to assess the learning ability and performance of the model.

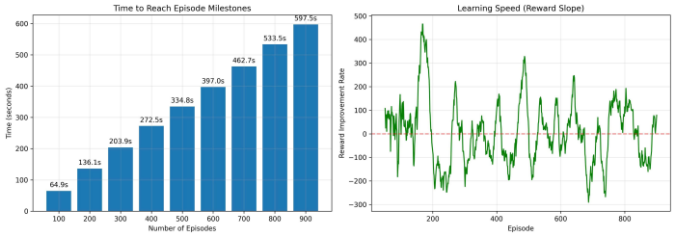

Fig. 5. Training time and learning rate of the model

Figure 5 shows that training time increases progressively with the number of episodes and tends to be near-linear, demonstrating the model's stability as the learning process expands. Simultaneously, the learning rate graph shows that rewards fluctuate sharply in the early stages due to the agent's continuous exploration of the environment, then stabilize in later episodes. While some positive and negative fluctuations remain, the model has generally learned better policies over time and significantly improved its training performance.

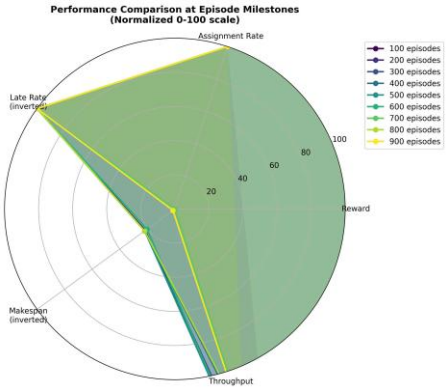

Fig. 6. Compare performance at different episode milestones

Figure 6 illustrates the model's performance improvement as the number of episodes increases. Metrics such as Assignment Rate, Throughput, and Late Rate (inverted) all reach high values at larger episode milestones, indicating efficient single-task assignment, handling more tasks, and maintaining a very low late-assignment rate. The radar area in

the later episodes is significantly larger, reflecting the model's gradual convergence and learning of a more optimal scheduling strategy after a long training period.

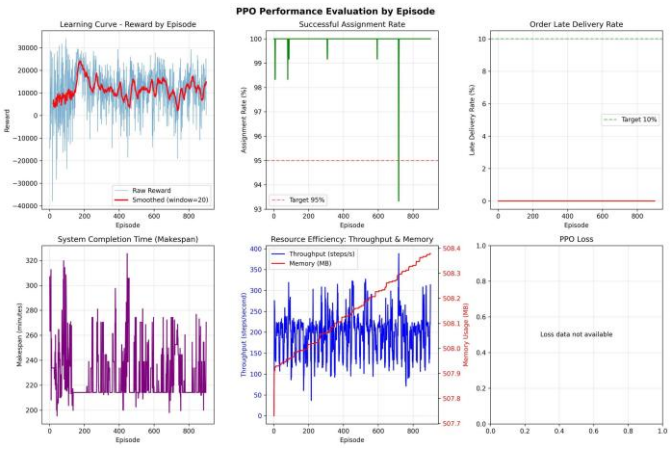

Fig. 7. Evaluating PPO performance by episodes

The reward shown in Figure 7 demonstrates an increasing and more stable trend over the training period, while the single assignment rate consistently remains near 100% and the late assignment rate is almost 0%. Furthermore, the makespan fluctuates within an acceptable range and the throughput remains stable, indicating that the system ensures both good processing performance and efficient resource utilization throughout the training process.

VII. Conclusion

In this study, we proposed an advanced approach based on an improved Proximal Policy Optimization (PPO) deep reinforcement learning algorithm to solve the real-time delivery scheduling problem in a high-density urban environment (R-RDP). By modeling the problem as a Markov Decision Process (MDP) combined with dynamic information on order status and system load, the proposed model demonstrated self-learning capabilities and flexible scheduling decisions. In particular, the integration of action masking and guide fallback mechanisms, along with a two-phase post-processing unit (batch insertion and global repair), enabled the system to comprehensively optimize both operational performance and stringent time constraints.

Experimental results on datasets from DO1 to DO10 demonstrated the superior efficiency and high reliability of the proposed PPO algorithm. The model not only maintains an ideal order fulfillment rate of approximately 100% and reduces late delivery rates to near 0%, but also excellently optimizes total system completion time (makespan) and processing throughput. The most significant characteristic of the improved PPO algorithm is its ability to narrow the optimal gap, maintaining stable performance between the best and average values as the number of training episodes increases, despite large and unstable fluctuations during the startup phase.

In the future, we will continue to expand this research direction by integrating more realistic constraints of the urban logistics industry, such as the continuous variation of weather, multimodal delivery models, or complex multi-objective distribution policies. Simultaneously, optimizing the neural network architecture to minimize standard deviation in the startup phase will also be emphasized to enhance the practical applicability of the model to intelligent transportation coordination systems in real-world scenarios.

References

[1] Kadyrov, S., Azamov, A., Abdumajitov, Y., & Turan, C. (2025). Deep reinforcement learning for dynamic vehicle routing with demand and traffic uncertainty. Operations Research Perspectives, 100351.
[2] Zhou, W., Fotouhi, H., & Miller-Hooks, E. (2025). Decision support through deep reinforcement learning for maximizing a courier's monetary gain in a meal delivery environment. Decision Support Systems, 190, 114388.
[3] Cordeiro, F. E. A., & Pitombeira-Neto, A. R. (2025). A deep reinforcement learning approach for the asynchronous dynamic vehicle dispatching problem. Artificial Intelligence for Transportation, 3, 100038.
[4] Zamal, M. A., Schrotenboer, A. H., & Van Woensel, T. (2025). End-to-end logistics in metropolitan areas: A stochastic dynamic order-assignment and dispatching problem. Transportation Research Part B: Methodological, 199, 103249.
[5] Liu, S., Yu, Y., Tian, Q., & Sun, H. (2026). Routing optimization for an eVTOL-and-drone delivery system in continuous space with no-fly zones: A reinforcement learning approach. Transportation Research Part E: Logistics and Transportation Review, 209, 104741.
[6] Li, C., Han, W., Li, H., Liu, J., Wang, X., Zhang, Y., & Su, X. (2026). Deep reinforcement learning for carrier-based aircraft flight deck operations scheduling problem. Neural Networks, 108776.
[7] Saleh, Z., Al Hanbali, A., & Baubaid, A. (2025). Enhancing courier scheduling in crowdsourced last-mile delivery through dynamic shift extensions: a deep reinforcement learning approach. Computers & Industrial Engineering, 111693.
[8] Dehghan, A., Cevik, M., & Bodur, M. (2026). Neural approximate dynamic programming for the ultra-fast order dispatching problem. IISE Transactions, 1-18.
[9] Sha, J., Song, M., Sui, G., Sun, H., & Dong, D. (2026). A multi-agent reinforcement learning scheduling algorithm integrating state graph and task graph structural modeling for ride-sharing dispatching. Scientific Reports.
[10] Lyu, W., Zhang, L., Zhong, S., Wang, H., Li, C., Wang, S., ... & Zhang, D. (2025). HeroCS: Cooperative Courier Scheduling for Heterogeneous Tasks in Last-Mile Delivery. IEEE Transactions on Mobile Computing.
[11] Saleh, Z., Baubaid, A., Al Hanbali, A., & Alromema, M. (2025). Comparing Reinforcement Learning Algorithms for Online Couriers Scheduling in Crowdsourced Last-Mile Delivery. Transportation Research Procedia, 84, 121-128.
[12] Wang, Y., He, L., Qi, Z., & Minner, S. (2025). Dispatch or Hold? An Inverse Optimization and Reinforcement Learning Approach for Multi-Objective On-Demand Delivery. An Inverse Optimization and Reinforcement Learning Approach for Multi-Objective On-Demand Delivery (November 30, 2025).
[13] Dehghan, A., Cevik, M., & Bodur, M. (2026). Neural approximate dynamic programming for the ultra-fast order dispatching problem. IISE Transactions, 1-18.