Transformer-Augmented Deep Reinforcement Learning for Multi-Objective Path Co-Optimization in Same-Day Delivery with Hybrid Fleets

Wei Tang[1,2], Wanyang Zhou[3*], Qianlan Wang[1], Cuixia Zhang[1]
[1]Anhui Research Center for Economic and Social Development, Tongling University, City of Tongling Anhui province 244061, China
[2]College of Business Administration and Accountancy, De La Salle University - Dasmariñas, City of Dasmariñas Cavite, 4115, Philippines
[3]College of Tourism and E-commerce, Baise University, City of Baise Guangxi province 533000, China
E-mail: vzaibv2285844@outlook.com
[*]Corresponding author

Keywords: same-day delivery, crowdsourced delivery, dynamic dispatching, multi-objective path co-optimization

Received: May 18, 2025

The rapid expansion of e-commerce and local services has intensified the demand for instant delivery, particularly same-day delivery (SDD), presenting significant challenges in operational efficiency and multi-objective cost management. Traditional logistics systems often struggle with dynamic order arrivals, complex routing decisions, and the effective integration of hybrid fleets comprising both dedicated and crowdsourced couriers. This paper investigates the multi-objective path co-optimization problem within logistics hyper-automation, proposing an innovative Transformer-Augmented Policy Optimization (TAPO) Double Layers Optimization Framework. This framework addresses the dynamic dispatching and routing challenges in SDD scenarios. The upper layer employs the TAPO agent, where a Transformer model processes complex spatio-temporal dependencies from dynamic order data to enrich state representations for a Deep Reinforcement Learning (DRL) agent (based on Proximal Policy Optimization - PPO). This TAPO agent learns a sophisticated policy for synergistic dispatch-delay decisions. The lower layer utilizes efficient heuristic algorithms (GCIH and VNS) for static vehicle routing and order assignment when a dispatch action is chosen. The primary goal is to co-optimize multiple objectives, focusing on minimizing total fulfillment costs (encompassing mileage and delay penalties) while enhancing service efficiency. A Markov Decision Process (MDP) models the sequential decision-making problem. Numerical experiments on diverse, realistic instances demonstrate the TAPO framework's superiority. Results show that the TAPO framework achieves an average total cost reduction of approximately 5-11% compared to a Myopic policy and 3-12% against an Urgent-Based Policy (UBP). The framework also exhibits robust generalization to varying order volumes and sensitivity to fleet composition changes, underscoring the significant potential of advanced AI techniques in achieving logistics hyper-automation.

Povzetek: Opisan je nov pristop za optimizacijo poti v sistemih za dostavo istega dne z uporabo hibridnih flot. Predlagani Transformer-Augmented Policy Optimization (TAPO) okvir združuje globoko ojačanje učenja in modeliranje zaporednih odločitev za zmanjšanje stroškov dostave.

1 Introduction

The proliferation of e-commerce and local life services has catalyzed an explosive growth in the instant delivery market, with consumer habits increasingly shifting towards online purchasing for a wide array of goods [1]. This trend has significantly propelled the demand for "same-day delivery" (SDD) services, which aim to deliver goods to customers within hours of an order being placed [2, 3]. SDD models typically utilize local fulfillment centers or front-end stores as warehousing points, processing dynamically arriving orders throughout operational hours. This paradigm, bridging the gap between traditional multi-day e-commerce fulfillment and ultra-fast meal delivery, presents novel operational challenges for service providers, primarily in balancing stringent delivery timeliness with escalating operational costs [4].

A critical aspect of managing SDD operations is the efficient utilization of delivery fleets. Traditionally, companies relied on dedicated, in-house couriers. However, with the rise of the sharing economy and crowdsourcing platforms, many service providers are now adopting hybrid fleet models that combine dedicated couriers with a flexible supply of crowdsourced delivery personnel [5, 6]. This "asset-light" approach offers scalability and can be a cost-effective supplement to existing capacity, especially during peak demand periods. Nevertheless, the integration of heterogeneous fleets adds complexity to the dispatching and routing decisions. In this dynamic environment, characterized by fluctuating

demand and evolving fleet compositions, the problems of dynamic order dispatching and multi-objective path co-optimization have become paramount for achieving cost reduction and efficiency improvements [7].

The inherent complexity of SDD, involving real-time decision-making under uncertainty with dynamically arriving orders and stochastic travel times, often renders traditional optimization methods and simple heuristics inadequate. These problems are typically characterized by large state and action spaces, making direct solutions via methods like Bellman equations computationally intractable, a phenomenon often referred to as the "curse of dimensionality" [8]. Consequently, there is a growing need for advanced intelligent algorithms that can learn effective strategies from data and adapt to changing conditions. Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm for tackling such complex sequential decision-making problems [9, 10]. DRL agents can learn optimal or near-optimal policies through interaction with the environment, making them well-suited for dynamic logistics scenarios where future information is uncertain and long-term rewards need to be considered [11]. Several studies have explored DRL for various vehicle routing problems (VRPs) and dynamic dispatching tasks, demonstrating its potential to outperform traditional approaches [12, 13].

Furthermore, logistics optimization in SDD is inherently a multi-objective problem. Service providers aim to simultaneously minimize various costs (e.g., total travel distance, labor costs), reduce delivery delays, and maximize customer satisfaction [14, 15]. These objectives are often conflicting, necessitating a multi-objective optimization (MOO) approach to find a set of Pareto-optimal solutions that represent different trade-offs. Path co-optimization, in this context, refers to the synergistic optimization of both dispatching decisions and the subsequent routing decisions for the assigned couriers, rather than treating them as separate, sequential steps.

To enhance the decision-making capabilities of DRL agents in such complex environments, advanced data processing techniques are beneficial. The logistics domain generates vast amounts of sequential and spatio-temporal data. Transformer models, originally developed for natural language processing, have shown remarkable success in capturing long-range dependencies and contextual information in sequential data due to their attention mechanisms [16]. Their application is expanding to various other fields, including logistics and transportation, for tasks like travel time estimation or demand prediction, which can provide richer state representations for DRL agents [17]. This research operates under the broader umbrella of Logistics Hyper-automation, which signifies a strategic approach to automate and optimize logistics processes end-to-end by leveraging a combination of advanced technologies, including AI and machine learning, to achieve greater efficiency and resilience [18, 19]. The Comparative Analysis of Prior Methods in SDD Dispatching is shown in Table 1.

This paper addresses the critical challenge of dynamic order dispatching and multi-objective path co-optimization in SDD systems that utilize a hybrid fleet of dedicated and crowdsourced couriers. We propose a novel Transformer-Augmented Policy Optimization (TAPO) Double Layers Optimization Framework. This framework integrates Transformer models with Deep Reinforcement Learning to learn sophisticated, adaptive strategies. The upper layer of the framework employs the TAPO agent, where a Transformer component processes complex sequential and contextual information from the operational environment, enhancing the state representation for a DRL agent (based on algorithms like Proximal Policy Optimization - PPO [20]). This TAPO agent then learns a policy to make synergistic dispatch-delay decisions. The lower layer is responsible for the path optimization for the heterogeneous fleet when a dispatch action is chosen. The primary goal is to co-optimize multiple objectives, focusing on minimizing total fulfillment costs (delay penalties and mileage costs) while ensuring service quality. This study aims to demonstrate how such an integrated AI approach can lead to significant improvements in the efficiency and cost-effectiveness of SDD operations, contributing to the advancement of logistics hyper-automation. In this paper we solve three research questions:

- Does integrating Transformer-based state representations into PPO improve dispatching efficiency in hybrid fleet logistics?
- How does the TAPO framework compare to traditional dispatching policies in terms of cost reduction and operational efficiency?
- What are the robustness and generalization capabilities of the TAPO framework across different order volumes and fleet compositions?

The remainder of this paper is organized as follows: Section 2 describes the problem and formulates the mathematical model. Section 3 details the proposed TAPO Double Layers Optimization Framework. Section 4 presents the numerical experiments and discusses the results. Finally, Section 5 concludes the paper and suggests directions for future research.

2 Problem description and mathematical model

This section formally defines the Same-Day Delivery Problem with Hybrid Fleets (SDDPHF) in the context of logistics hyper-automation. We delineate the system components, decision-making processes, and formulate a Markov Decision Process (MDP) model tailored for multi-objective path co-optimization.

Table 1: Comparative analysis of prior methods in SDD dispatching

Method Type	Description	Performance Metrics	Limitations
Rule-Based Heuristics	Simple, predefined rules for	Dispatch time, route length	Assumes static conditions, limited scalability

	dispatching and routing		
Standard DRL	Basic DRL algorithms without additional enhancements	Total cost, delivery time	Struggles with complex state spaces, limited generalization
MILP-Based Methods	Mathematical optimization using Mixed-Integer Linear Programming	Cost minimization, service level	Computationally intensive, static assumptions
TAPO (Proposed)	Transformer-Augmented Policy Optimization for dynamic dispatch-delay decisions and routing	Multi-objective cost reduction, service efficiency	Assumes certain vehicle speed and service area constraints

To model this problem effectively, we make the following assumptions:

1.Assumption on Courier Availability: We assume that courier availability is known and relatively stable within the operational time window. This assumption allows us to focus on the optimization of dispatch and routing decisions without the added complexity of dynamic courier availability changes.

2.Assumption on Order Deadlines: We assume that order deadlines are provided and are relatively uniform across customers. This assumption helps to standardize the problem and allows for a more focused analysis of the dispatch and routing optimization.

3.Assumption on Fixed Simulation Parameters: We assume that certain simulation parameters, such as vehicle speeds and service areas, are fixed and known in advance. This assumption is necessary to create a controlled environment for testing and validating the TAPO framework.

2.1 Problem setting

The Same-Day Delivery Problem with Hybrid Fleets (SDDPHF) addresses the dynamic dispatching and routing of customer orders that arrive sequentially over a finite operational horizon, denoted as T. Operations occur within a defined geographical area, represented by a graph $G = (N, E)$. Here, N is the set of nodes, including a central depot N0 (such as a front-end store or local fulfillment center), various customer locations N_C, and potential starting locations for crowdsourced couriers N_P. The set E comprises edges that represent travel paths, each associated with specific travel times or distances. Customer orders, $O = o_1, o_2, \ldots, o_{|O|}$, arrive dynamically, with each order oi characterized by its arrival time t_i^a, customer location, and a desired delivery window or latest acceptable delivery time t_i^d. The system is designed to fulfill all accepted orders [6].

Delivery operations are executed by a hybrid fleet, which includes a dedicated fleet and a crowdsourced fleet. The dedicated fleet, $F = f_1, f_2, \ldots, f_M$, consists of M in-house couriers who typically start and end their routes at the depot N_0. These couriers have specific capacities q_f and operational characteristics, such as speed v_f. Complementing this is a dynamically available pool of L crowdsourced couriers, $P = p_1, p_2, \ldots, p_L$, who can be engaged on demand. Crowdsourced couriers may possess different capacities q_p, potentially lower speeds v_p, and distinct operational patterns; for instance, they might not be required to return to the depot after completing their deliveries. The unique characteristics of dedicated versus crowdsourced couriers, including travel speed, single-trip carrying capacity, and route start/end points, are fundamental to the model. A detailed comparison is presented in Table 2.

Table 2: Comparison of dedicated and crowdsourced courier characteristics

	Dedicated Courier	Crowdsourced Courier
Driving Speed	Fast	Slow
Order Capacity per Trip	High	low
Starting Point of Delivery Route	Pre-determined Warehouse	Location at Time of Dispatch
End Point of Delivery Route	Pre-determined Warehouse	Last Order Location, Exit System

Decisions concerning order dispatching and vehicle routing are made at discrete decision epochs, t_k. These epochs can be triggered either by fixed time intervals (e.g., every Δt minutes) or by specific events, such as the return of a dedicated courier to the depot, which makes them available for new assignments. These decision epochs are shown in Figure 1. At each decision epoch t_k, the system must address several intertwined decisions: deciding which currently unassigned orders (both newly arrived and previously delayed) to dispatch immediately and which to delay for potential consolidation; assigning dispatched orders to available dedicated or crowdsourced couriers; and for each courier assigned a set of orders, determining the optimal delivery sequence (route) to co-optimize multiple objectives.

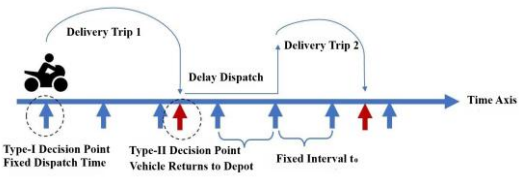

Figure 1: Dynamic dispatch decision epochs

The primary goal of the system is multi-objective optimization, aiming to enhance performance across several, often conflicting, criteria. These objectives typically include minimizing total operational costs (comprising mileage-based costs proportional to total

distance traveled and delay penalties for late deliveries) and maximizing service levels (e.g., minimizing average delivery time, maximizing on-time deliveries). Additionally, while not always mandatory, ensuring a fair distribution of workloads among delivery personnel is considered as a supplementary objective to maintain operational equity and efficiency. The overall objective function can be structured as a weighted sum of these components or approached from a Pareto-optimality perspective. A common formulation is to minimize a composite cost function:

$$\min Z = w_1 \cdot C_{\text{mileage}} + w_2 \cdot C_{\text{delay}} + \sum_{j=3}^{N_{\text{obj}}} w_j \cdot C_j \quad (1)$$

where C_{mileage} is the total mileage cost, C_{delay} is the total delay penalty, C_{j} represents other cost or performance metrics, and w_j are their respective weights reflecting their relative importance.

2.2 Markov decision process (MDP) formulation

The dynamic and stochastic nature of the SDDPHF lends itself to modeling as a Markov Decision Process (MDP). The MDP is defined by a tuple (S, A, P, R, γ):

The state $s_k \in S$ at a decision epoch t_k must encapsulate all information relevant to decision-making. This includes the current time t_k; the set of unassigned orders $O_{\text{pool}}(k)$, which comprises newly arrived orders O_{new} since the last epoch tk−1 and any orders O_{remain} held over from previous epochs (each order $o_i \in O_{\text{pool}}$ is described by its features like location, arrival time, and deadline); the status of dedicated couriers$V_F(k)$, detailing for each $f_j \in F$ their current location, remaining capacity, and estimated time of arrival back at the depot if en-route; the availability and status of crowdsourced couriers $V_P(k)$, including their current availability, location, and capacity; and information about ongoing routes $R_{\text{active}}(k)$ for couriers already dispatched, including the sequence of remaining deliveries and estimated completion times. The state can be formally represented as:

$$s_k = \left(t_k, O_{\text{pool}}(k), V_F(k), V_P(k), R_{\text{active}}(k)\right) \quad (2)$$

The component $R_{\text{active}}(k)$ is crucial as it includes information about routes assigned in previous epochs that are still in progress. For instance, for a dedicated courier f_i dispatched at an earlier epoch $k' < k$, its route $r(f_i, k')$ would be part of $R_{\text{active}}(k)$ until completed.

To capture complex dependencies and sequential patterns, such as those in order arrivals or courier availability, raw state features can be processed by a Transformer encoder. The Transformer's self-attention mechanism can effectively weigh the importance of different elements in sequences of data (e.g., features of orders in O_{pool} or courier activities) to produce a richer, context-aware state embedding $\Phi(s_k)$. For a sequence of input features $X = (x_1, x_2, \ldots, x_n)$, the self-attention mechanism computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

where Q, K, V are query,key,and value matrices derived from the input embeddings,and d_k is the dimension of the keys.This allows the model to learn inter-order relationships or temporal patterns crucial for effective dispatching and routing.The output of the Transformer,$\Phi(s_k)$,then serves as the input to the DRL agent's policy and value networks, shown as figure 2.

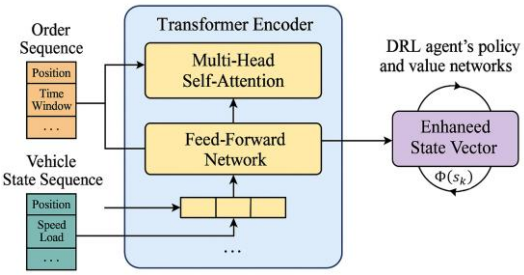

Figure 2: Transformer based state representation enhancement structure

At each state s_k ,the agent chooses an action $a_k \in A(s_k)$ .This action is multi-faceted, involving a dispatch decision $a_D(k)$ (e.g., a binary choice to dispatch all current orders or delay them),an order-courier assignment $a_A(k)$ if dispatching (matching selected orders to available couriers from $F \cup P$,considering capacities and fleet characteristics),and a routing decision $a_R(k)$ .The routing decision determines the sequence of deliveries for each assigned courier. A route r_j for courier j is a sequence of customer nodes $(o_{j1}, o_{j2}, \ldots, o_{jm_j})$.The set of all routes generated at epoch k is $R'_k = r_1, r_2, \ldots$ .For a dedicated courier f_i, a route $r(f_i, k)$ dispatched at epoch k can be represented as [21]:

$$r(f_i, k) = \left(N_0, o_{i1}, \ldots, o_{im_i}, N_0\right), t^r_{f_i}, \text{ distance } f_i^k \quad (4)$$

where $\left(N_0, o_{i1}, \ldots, o_{im_i}, N_0\right)$ is the sequence of nodes, $t^r_{f_i}$ is the courier's return time to the depot, distance $^k_{f_i}$ is the route distance, and delay f_i^k is the accumulated delay. For a crowdsourced courier p_j, a route $r(p_j, k)$ might start at their current location $N_{p'}$, proceed to the depot N_0 for pick-ups, deliver to customers $o_{j1}, \ldots, o_{jn_j}$ and end at the last customer location:

$$r(p_j, k) = \left(N_{p_j} \to N_0 \to o_{j1}, \ldots, o_{jn_j}\right), \text{ distance } ^k_j \quad (5)$$

The composite action is thus $a_k = (a_D(k), a_A(k), a_R(k))$ .The state transition function $P(s_{k+1} \mid s_k, a_k)$ defines the probability of moving from state s_k to s_{k+1} after action a_k. This transition is shaped by deterministic consequences of a_k (like courier

movements and order fulfillment) and stochastic elements (such as new order arrivals $O_{\text{new}}(k+1)$, changes in crowdsourced courier availability $V_P(k+1)$, and uncertain travel times). The next decision epoch t_{k+1} is determined by:

$$t_{k+1} = \min\left(\min_{f_i \in F_{\text{en-route}}} t^r_{f_i}, \left(\left\lfloor \frac{t_k}{\Delta t} \right\rfloor + 1\right) \cdot \Delta t\right) \quad (6)$$

where $F_{\text{en-route}}$ is the set of dedicated couriers currently on a route. The order pool $O_{\text{pool}}(k+1)$ and courier statuses $V_F(k+1), V_P(k+1)$ are updated based on s_k, a_k, and new exogenous information $w_{k+1} = (O_{\text{new}}(k+1), V_P(k+1))$.

The reward function $R(s_k, a_k, s_{k+1})$ quantifies the immediate outcome of action a_k. Given the multi-objective nature, it's a scalar value reflecting the action's desirability. Typically, it's the negative of a weighted sum of costs incurred:

$$R_k = -\left(w_1 \cdot \Delta C_{\text{mileage}}(k) + w_2 \cdot \Delta C_{\text{delay}}(k) + \sum_{j=3}^{N_{\text{obj}}} w_j \cdot \Delta C_j(k)\right) \quad (7)$$

where $\Delta C_{\text{mileage}}(k)$ and $\Delta C_{\text{delay}}(k)$ are mileage and delay costs from routes in $a_R(k)$. Reward shaping can be used to provide denser feedback. The discount factor $\gamma \in [0,1]$ balances immediate versus future rewards.

The DRL agent's goal is to learn an optimal policy $\pi^*: S \to A$ that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{i=k}^{K-1} \gamma^{i-k} R(s_i, \pi(s_i), s_{i+1}) \mid s_k\right] \quad (8)$$

for all $s_k \in S$, where K is the total number of decision epochs. The "curse of dimensionality" arising from vast state and action spaces makes exact solutions to the Bellman optimality equation infeasible. The Bellman equation for the optimal action-value function $Q^*(s,a)$ is:

$$Q^*(s,a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')] \quad (9)$$

This necessitates DRL algorithms using function approximators (e.g., deep neural networks) to learn the policy $\pi(a|s;\theta)$ or value functions, where θ are network parameters.

2.3 Illustrative scenario

To clarify the decision-making process and state transitions, consider a simplified scenario. An illustration of the system state at a specific decision epoch is shown as Figure 3 at decision epoch t_k =120 min.

The DRL agent observes the system state S_k.Based on this observation, potentially enhanced by a Transformer model, the agent decides on an action a_k.One option is to delay dispatch, in which case all orders currently in $O_{\text{pool}}(k)$ remain,and the system transitions to the next decision epoch t_{k+1} .Another option is immediate dispatch, where orders are assigned to available couriers(e.g., f_1 and p_1),and routes are planned.For instance,orders o_1, o_2, o_3 might be assigned to f_1 ,and order o_4 to p_1.

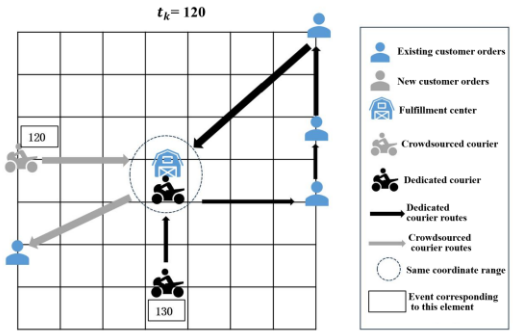

Figure 3: System state at decision epoch t_k=120 min

Assume the next decision epoch t_{k+1} occurs at 130 min, triggered by the return of courier f_2 to the depot (assuming this is earlier than $120 + \Delta t$).During the interval from 120 min to 130 min, new orders o_5, o_6, o_7 arrive (this is the exogenous information w_{k+1}),and a new crowdsourced courier p_2 becomes available.

If the agent chose to delay dispatch at $t_k = 120$ min,then at $t_{k+1} = 130$ min ,the order pool $O_{\text{pool}}(k+1)$ would now contain the original orders o_1, o_2, o_3, o_4 plus the new orders o_5, o_6, o_7.An illustration of the system state after a delay decision is shown as Figure 4. The available couriers would include f_1(who was waiting),f_2(who just returned), and the newly available p_2.The DRL agent then makes a new dispatch and routing decision for this larger pool of orders and updated fleet.

Conversely, if the agent chose immediate dispatch at $t_k = 120$ min ,then at $t_{k+1} = 130$ min , the order pool $O_{\text{pool}}(k+1)$would only contain the new orders o_5, o_6, o_7 ,as orders $o_1 - o_4$ are already enroute with couriers f_1 and p_1. An illustration of the system state after an immediate dispatch decision is shown as Figure 5. The available couriers for these new orders would be f_2 (who just returned) and p_2. Couriers f_1 and p_1 are still completing their routes initiated at t_k. The DRL agent then makes decisions for the new orders using the currently available fleet.

This illustrative scenario highlights how strategic delays, guided by a learned policy, can potentially yield superior overall solutions. By enabling order consolidation and more efficient utilization of vehicle capacity, such

strategies can reduce total costs. The DRL agent's function is to discern when such delays are advantageous by analysing the complex interplay of current state variables and future uncertainties.

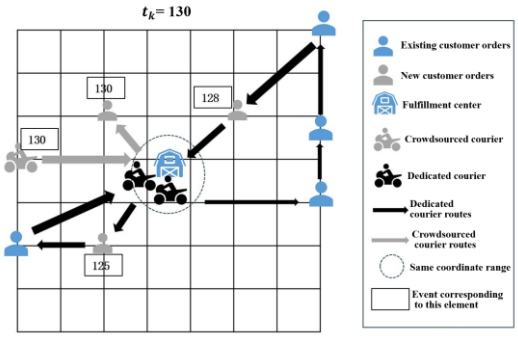

Figure 4: System State at $t_{k+1} = 130$ min after Delaying Dispatch at t_k

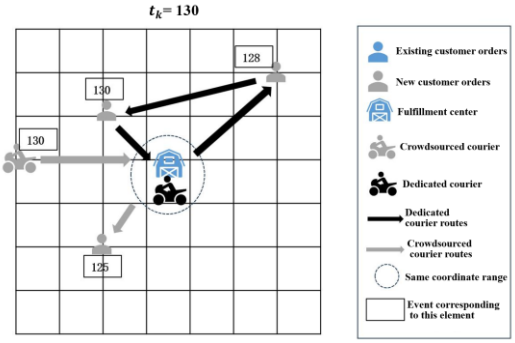

Figure 5: System State at $t_{k+1} = 130$ min after Immediate Dispatch at t_k

3 Algorithm design

With the mathematical model established, we now focus on the algorithmic design to solve the formulated problem effectively. This chapter details the hybrid algorithmic approach developed to address the Same-Day Delivery Problem with Hybrid Fleets (SDDPHF). The proposed solution employs double layers optimization framework. At its core, an advanced Deep Reinforcement Learning (DRL) method, termed Transformer-Augmented Policy Optimization (TAPO), is responsible for making dynamic, global dispatch-delay decisions. This is complemented by heuristic algorithms designed to solve the static routing subproblems that arise when a dispatch decision is made. This integrated approach aims to achieve overall optimization within the operational period of same-day delivery services, balancing immediate operational needs with long-term strategic goals.

3.1 Upper-Layer: transformer-augmented policy optimization (TAPO)

The Transformer-Augmented Policy Optimization (TAPO) agent is central to our approach for the dynamic dispatch-delay decision-making process. It learns its policy through continuous interaction with a simulated environment that accurately mimics the SDDPHF system. The design of this DRL environment, particularly its observation space, action space, and reward function, is critical for successful learning and for leveraging the capabilities of the Transformer architecture. The interaction loop between the DRL agent and the environment is depicted in Figure 6.

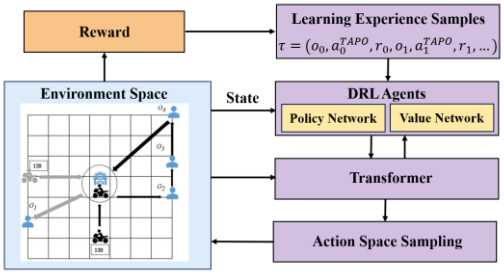

Figure 6: TAPO Agent-environment interaction loop

To manage the complexity of the full MDP state s_k and ensure that the learning process is feasible, the TAPO agent receives a carefully selected observation $o_k \in S_{\text{obs}}$ at each decision epoch t_k. This observation is a compact yet informative representation of the system's current state. It typically includes the current time t_k within the operational horizon $[0, T]$; aggregate information about the order pool $O_{\text{pool}}(k)$, such as the total number of waiting orders, and potentially specific features of the most urgent order (e.g., the time remaining until its deadline, UT_k); and information about available fleet capacity, such as the number of dedicated couriers V_k currently at the depot and ready for dispatch. Crucially, to provide richer contextual information and enable more sophisticated decision-making, features extracted by a Transformer model from recent historical order data (e.g., sequences of order locations, demand volumes, arrival times) or predicted future demand patterns can be included. The Transformer architecture comprises an embedding layer, followed by 6 encoder layers. Each encoder layer consists of a multi-head self-attention mechanism and a position-wise feed-forward network. This Transformer processed information, denoted $\Phi_{\text{context}}(k)$, allows the agent to better understand temporal dependencies and complex patterns that might influence the optimal dispatch strategy. Thus, the observation vector can be represented as $o_k = \left(t_k, \text{count}\left(O_{\text{pool}}(k)\right), UT_k, V_k, \Phi_{\text{context}}(k)\right)$.

The TAPO agent's action space A_{TAPO} is simplified for the dispatch-delay decision to maintain tractability. At each epoch t_k, the agent chooses a discrete action a_k^{TAPO}

from the set 0,1. An action $a_k^{TAPO} = 0$ signifies a decision to delay the dispatch of all orders currently in $O_{\text{pool}}(k)$; these orders then remain in the pool and are reconsidered at the next decision epoch. Conversely, an action $a_k^{TAPO} = 1$ indicates a decision to dispatch the orders currently in $O_{\text{pool}}(k)$. This action triggers the execution of the lower-layer heuristic algorithm, which then handles the detailed order assignment and routing tasks.

Designing an appropriate reward function $R_{\text{TAPO}}(o_k, a_k^{TAPO}, o_{k+1})$ is paramount for effectively guiding the TAPO agent towards achieving the desired multi-objective optimization goals. A naive reward signal, such as one based solely on the final system cost at the end of an entire episode (e.g., a full operational day), would likely be too sparse, rendering the learning process highly inefficient. Therefore, a shaped reward function is employed to provide more immediate and informative feedback to the agent. This reward is a composite function that includes the negative of the immediate operational costs incurred if the dispatch action ($a_k^{TAPO} = 1$) is taken. These costs, $\Delta C_{\text{mileage}}(k)$ and $\Delta C_{\text{delay}}(k)$, are derived from the solution provided by the lower-layer routing algorithm. Additionally, the shaped reward incorporates auxiliary rewards or penalties designed to encourage desirable intermediate behaviors or discourage undesirable ones. For instance, a small penalty might be applied for delaying orders, particularly if they are approaching their delivery deadlines, to reflect the implicit cost of waiting. Conversely, a bonus is awarded for optimal utilization of vehicle capacity during dispatch operations or for successfully completing an entire operational day within service targets. The specific formulation of this shaped reward function, r_k, often involves a sum of several components $r^{(i)}(S_t, a_t)$ minus the objective cost penalty. These components are carefully tuned to balance short-term operational efficiency with long-term strategic objectives like beneficial order consolidation, and might include rewards or penalties related to delaying dispatch based on current system load $(r^{(1)})$, vehicle return events enhancing capacity $(r^{(2)})$, the urgency or earliness of orders $(r^{(3)})$, the number of orders completed $(r^{(4)})$, and the successful completion of the service period $(r^{(5)})$.

TAPO employs two main deep neural networks: a Policy Network (Actor), $\pi(a \mid o; \theta_{\text{actor}})$, which maps the potentially Transformer-enhanced observation o_k to a probability distribution over actions (delay or dispatch), and a Value Network (Critic), $V(o; \theta_{\text{critic}})$, which estimates the expected cumulative future reward from state o_k. Both networks are commonly implemented as Multi-Layer Perceptrons (MLPs). The policy network uses a softmax output layer for discrete actions. The model training algorithm for TAPO involves the DRL agent interacting with the simulated SDDPHF environment over numerous episodes. During each episode, the agent collects trajectories of experiences $\tau = (o_0, a_0^{TAPO}, r_0, o_1, a_1^{TAPO}, r_1, \dots)$. After collecting a batch of trajectories, TAPO updates the parameters θ_{actor} and θ_{critic}. The core PPO objective function for the actor is the clipped surrogate objective:

$$L^{\text{CLIP}}(\theta_{\text{actor}}) = \hat{\mathbb{E}}_t\left[\min\left(\rho_t(\theta_{\text{actor}})\hat{A}_t, \text{clip}(\rho_t(\theta_{\text{actor}}), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (10)$$

In this equation, $\rho_t(\theta_{\text{actor}}) = \frac{\pi(a_t|o_t;\theta_{actor})}{\pi(a_t|o_t;\theta_{actorold})}$ is the probability ratio. $\hat{A}_t$ is an estimator of the advantage function at timestep t, often computed using Generalized Advantage Estimation (GAE), where $\hat{A}t = Q(o_t, a_t) - V(o_t; \theta$ critic). The hyperparameter ϵ defines the clipping range. The critic network is trained by minimizing a loss function, usually the mean squared error between its predicted state values $V(o_t; \theta_{\text{critic}})$ and the empirically estimated returns $\hat{R}_t$:

$$L^{VF}(\theta_{\text{critic}}) = \hat{\mathbb{E}}t\left[\left(V(o_t; \theta_{\text{critic}}) - \hat{R}t\right)^2\right] \quad (11)$$

An entropy bonus $[\pi(\cdot \mid o_t; \theta_{\text{actor}})]$ can be added to the actor's objective to encourage exploration. The final combined loss function often takes the form:

$$L(\theta_{\text{actor}}, \theta_{\text{critic}}) = L^{CLIP}(\theta_{\text{actor}}) - c_1 L^{VF}(\theta_{\text{critic}}) + c_2 S[\pi(\cdot \mid o_t; \theta_{\text{actor}})] \quad (12)$$

where c_1 and c_2 are weighting coefficients. The networks are updated using gradient-based optimization. This iterative cycle allows the TAPO agent to progressively learn an effective dispatch-delay strategy.

3.2 Lower-layer: static subproblem algorithm

When the upper-layer TAPO agent decides to dispatch orders at a given epoch t_k, the set of available orders in the pool $O_{\text{pool}}(k)$ must be assigned to the currently available dedicated fleet $F(k)$ and the callable crowdsourced fleet $P(k)$. Subsequently, efficient delivery routes must be planned for each courier. This static HFVRP is addressed using a twostage heuristic approach: an initial solution is first constructed, and then it is iteratively improved.

For the rapid generation of a feasible and reasonably good initial solution for the HFVRP, we employ a Greedy Costbased Insertion Heuristic (GCIH). This heuristic iteratively assigns unassigned orders to routes and inserts them into the most advantageous positions within those routes. The "best" position is determined by an insertion

cost function designed to consider multiple objectives, primarily focusing on minimizing the additional travel distance and any potential delay incurred by the insertion. The GCIH prioritizes assigning orders to dedicated couriers first, up to their capacity q_f, often due to their potentially higher efficiency or contractual obligations. The core of the GCIH involves calculating two types of costs for inserting an unassigned order v into a partially constructed route between existing nodes i and j. The first is the direct insertion cost, $c_1(i,v,j) = d_{iv} + d_{vj} - \mu \cdot d_{ij}$, where d_{xy} represents the distance between nodes x and y, and μ is a weighting parameter. The best insertion position $(i(v), j(v))$ for order v in a specific route is the one that minimizes this c_1 cost. The second is a relative insertion cost, $c_2(i(v), v, j(v)) = \gamma \cdot d_{0v} - \min c_1(i(v), v, j(v))$, where d_{0v} is the distance from the depot (node 0) to order v, and γ is another weighting factor. This c_2 cost helps in selecting which order v^* to insert next from the pool of unassigned orders, by identifying the v^* that minimizes c_2 . The algorithm proceeds by iteratively selecting the best order to insert and its best position within a route until all orders are assigned or all vehicle capacities are met. Any remaining orders are then considered for assignment to crowdsourced couriers, following a similar insertion logic but adapted for their specific operational characteristics (e.g., crowdsourced couriers might not need to return to the depot and may have different capacities q_p). Orders that cannot be assigned in the current dispatch cycle are returned to the order pool for consideration in the next decision epoch.

Once an initial solution is constructed by the GCIH, a Variable Neighborhood Search (VNS) algorithm is applied to further improve its quality. VNS is a metaheuristic that systematically explores different neighborhood structures to avoid getting trapped in local optima and to find higher-quality solutions. The VNS algorithm proceeds in iterations, typically involving a shaking phase and a local search phase. In the shaking phase, the current best solution is perturbed by applying one or more predefined neighborhood operators (moves), generating a new starting point in a potentially different region of the solution space. This helps in diversifying the search. Following the shaking phase, an intensive local search is performed from this perturbed solution. The local search phase uses a set of neighborhood operators to explore the vicinity of the current solution, aiming to find a local optimum. If this newly found local optimum is better than the current overall best solution, it replaces the current best solution, and the search process (particularly the choice of neighborhood for shaking) might reset to the first neighborhood structure. Otherwise, if no improvement is found, the VNS typically moves to the next type of neighborhood structure for the subsequent shaking phase, thereby changing the landscape of the search.

The effectiveness of VNS heavily relies on the design of its neighborhood operators. We have designed two main categories of operators specifically tailored for the HFVRP. Intra-route operators modify a single courier's route to improve its individual efficiency. Examples include a Greedy Swap, which exchanges two orders within the same route if such an exchange reduces the route's cost (e.g., total distance or lateness), and Random Insertion (Reinsertion), which involves removing an order from its current position in a route and reinserting it into a different, randomly chosen, feasible position within that same route. The Greedy Swap operator can be illustrated in Figure 7(a). The Random Reinsertion operator can be illustrated in Figure 7(b).

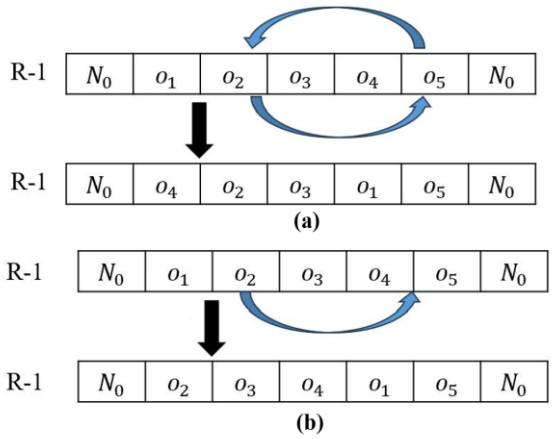

Figure 7: Intra-route operators: (a)greedy swap (b) random reinsertion

Inter-route operators, on the other hand, modify two or more routes simultaneously, allowing for a broader exploration of the solution space and facilitating the movement of orders between different couriers (including between dedicated and crowdsourced couriers). Examples include Relocate, which moves an order from one route to a feasible position in another route (respecting capacity and fleet-specific constraints such as return-to-depot requirements), and Exchange (Swap), which exchanges an order from one route with an order from another route. The Relocate operator can be illustrated in Figure 8(a). The Exchange operator can be illustrated in Figure 8(b). The careful selection and sequencing of these operators within the VNS framework are managed to effectively balance the breadth of exploration with the depth of exploitation. The VNS algorithm continues for a predefined number of iterations or until a termination criterion, such as no further improvement being found for a certain period, is met.

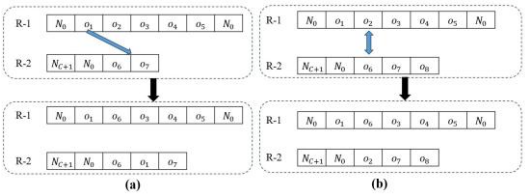

Figure 8: Inter-route operators: (a) relocate (b) exchange

3.3 Transformer-augmented policy optimization (TAPO) double layers optimization framework

The Transformer-Augmented Policy Optimization (TAPO) Double Layers Optimization Algorithm is shown as Algorithm 1:

Algorithm 1: TAPO double layers optimization

Input: Simulation environment env, training instances INS, number of training episodes EP, batch size BS, number of epochs per update BT, hyperparameters (γ_{discount} , ϵ_{clip} , λ_{GAE} , learning rates, etc.), Transformer model parameters $\theta_{\text{transformer}}$ (if pre-trained or jointly trained)

Output: Trained policy network parameters θ_{actor} , trained value network parameters θ_{critic}

1. Initialize policy network $\pi(a \mid o; \theta_{\text{actor}})$ with random weights θ_{actor}
2. Initialize value network $V(o; \theta_{\text{critic}})$ with random weights θ_{critic}
3. (Optional: Initialize/Load Transformer model $\Phi(\cdot; \theta_{\text{trans former}})$)
4. for episode = 1 to EP do
5. Initialize an empty list batch_trajectories
6. for rollout = 1 to BS do
7. Select a training instance in from INS
8. Reset environment env with instance in , get initial raw state features s_0^{raw}
9. (Optional: Process s_0^{raw} with Transformer to get o_0
10. Initialize an empty list current_trajectory
11. while episode is not done do
12. Choose action a_t^{TAPO} from $\pi(a \mid o_t; \theta_{\text{actor }_old})$
13. Execute a_t^{TAPO} in env, observe reward r_t and next raw state s_{t+1}^{raw}
14. (Optional: Process s_{t+1}^{raw} with Transformer to get o_{t+1})
15. Store $(o_t, a_t^{TAPO}, r_t, o_{t+1})$ in current_trajectory
16. $o_t \leftarrow o_{t+1}$
17. end while
18. Add current_trajectory to batch_trajectories
19. end for
20. Compute advantage estimates $\hat{A}_t$ (e.g., using GAE) for all steps in batch_trajectories
21. Compute discounted returns $\hat{R}_t$ for all steps in batch_trajectories
22. $\theta_{\text{actor }_old} \leftarrow \theta_{\text{actor}}$
23. $\theta_{\text{critic }_old} \leftarrow \theta_{\text{critic}}$
24. for epoch = 1 to BT do
25. For each trajectory in batch_trajectories :
26. Calculate probability ratio $\rho_t(\theta_{\text{actor}})$
27. Calculate policy loss L^{CLIP} (θ_{actor}) using Equation (10)
28. Calculate value loss L^{VF} (θ_{critic}) using Equation (11)
29. Calculate entropy bonus $S[\pi(\cdot \mid o_t; \theta_{\text{actor}})]$
30. Calculate combined loss $L(\theta_{\text{actor}}, \theta_{\text{critic}})$ using Equation (12)

4 Experiments

This section presents a comprehensive evaluation of the proposed Transformer-Augmented Policy Optimization (TAPO) Double Layers Optimization Framework. We first describe the experimental setup, including data generation and parameter settings. Subsequently, we analyze the performance of the lower-layer heuristic algorithms for solving the static subproblems. The core of this section then focuses on the training and validation of the TAPO agent, comparing its performance against several baseline dispatch strategies across various scenarios to demonstrate its efficacy in reducing costs and improving operational efficiency in same-day delivery (SDD) operations. Finally, generalization and sensitivity analyses are conducted to assess the robustness and applicability of the TAPO framework.

4.1 Experimental setup and data generation

To rigorously evaluate the proposed algorithms, a discrete-event simulation environment was developed to mimic the dynamic operations of an SDD system with hybrid fleets. The operational period for the SDD service is set to 600 minutes (e.g., from 9：00 AM to 7：00 PM). Dispatch decisions are made at fixed intervals of 20 minutes, or when a dedicated courier returns to the depot. The service area is represented as a 30×30 grid, approximating a 3 km × 3 km region, with the depot located at the center.For each order,the expected service time (latest delivery time after order placement) is 70 minutes.Other key parameters,including vehicle speeds (v_f for dedicated,v_p for crowdsourced),vehicle capacities (q^f, q_i^p), and objective function weights (α for distance cost,β for delay cost),are set based on realistic logistics scenarios and prior research. To enhance the reproducibility of our simulation environment, we provide additional details in the supplementary information section. The simulation parameters include seed values for random number generation, order arrival rate distributions (Poisson process with λ = 5 orders per minute during peak hours and λ = 2 orders per minute during off-peak hours), and spatial distribution parameters (grid dimensions of 50x50 with a depot at the center). For the demand curves, we have included numerical examples showing peak order rates of up to 10 orders per minute and off-peak rates of 2

orders per minute. Sample maps are described as 50x50 grids with customer locations distributed either uniformly or clustered in a 20x20 central area with obstacles. Fleet configurations include dedicated couriers with a speed of 60 km/h and a capacity of 6 orders, and crowdsourced couriers with a speed of 40 km/h and a capacity of 3 orders. These parameters are detailed in Table 3.

Table 3: Experimental parameter settings

parameter	Setting
v_f	0.2(km/h)
v_p	0.15(km/h)
q^f	6
q_i^p	3
α	0.2
β	0.8

To reflect diverse operational conditions, test instances were generated with varying characteristics in terms of order spatial and temporal distributions. Spatially, orders can be Clustered(C), with 50%of customers located in a dense 20x20 central area and the rest distributed in the periphery, or Spatially Uniform(S), where order locations are uniformly random across the map. Temporally ,order arrivals can be Homogeneous(A),following a Poisson process with a constant average rate, or Non-homogeneous(B),simulating peak and off-peak periods with order arrivals following a normal distribution centered around predefined peak times(e.g.,1/4T and 3/4T).An instance is denoted as " N-M-L- TimeType-SpaceType " ,where N is the total number of orders, M is the number of dedicated couriers, and L is the number of crowdsourced couriers available per dispatch wave. For example, " 600-6-2-B-S " represents a scenario with 600 orders, 6 dedicated couriers, 2 available crowdsourced couriers per wave, non-homogeneous(peaked)order arrivals, and spatially uniform order locations. Multiple instances were generated for each type to ensure statistical validity.

4.2 Performance of lower-layer heuristic algorithms

The efficiency of the lower-layer algorithms—Greedy Cost-based Insertion Heuristic (GCIH) for initial solution generation and Variable Neighborhood Search (VNS) for improvement—is crucial for the overall performance of the TAPO framework, as they are called repeatedly during the DRL agent's interaction with the environment. We evaluated these heuristics on a set of static subproblem instances derived from the dynamic scenarios. Performance metrics included total cost (TC, the weighted sum of distance cost DC and overtime cost OC), DC, OC, and CPU time. The comparative results of GCIH and VNS are detailed in Table 4.

The results indicated that VNS significantly improves upon the solutions generated by GCIH, typically reducing the total cost by a substantial margin (e.g., 8% to 65% across different instances) and consistently eliminating overtime costs in most tested cases. While GCIH is very fast (often under 0.005s), VNS provides much higher quality solutions within a reasonable timeframe (generally less than 1 second for instances with up to 40 orders), making it suitable for real-time decision support within the DRL loop. The convergence analysis of VNS showed that the objective function value rapidly decreases within the initial iterations (e.g., 10-20 iterations) and tends to stabilize after around 60 iterations for the tested problem sizes. This allows for setting a practical limit on VNS iterations to balance solution quality and computational speed. The convergence behavior of the VNS algorithm is shown in Figure 9.

Table 4: Performance comparison of GCIH and VNS on static subproblems

Subproblem	Algorithm	OBJ	DC	OC	Diff (%)	std	CPU Time (s)
10-2-1-A-C	VNS	36.58	182.89	0.00		0.91	0.464
	GCIH	41.66	208.29	0.00	13.89	0.00	0.001
10-2-1-A-S	VNS	33.23	166.16	0.00		0.48	0.541
	GCIH	54.99	210.35	16.16	65.48	0.00	0.002
20-3-1-A-C	VNS	53.08	265.40	0.00		0.52	0.904
	GCIH	60.39	301.94	0.00	13.77	0.00	0.002
20-3-1-A-S	VNS	54.05	270.23	0.00		1.59	0.904
	GCIH	57.14	285.72	0.00	5.73	0.00	0.001
30-4-2-A-C	VNS	73.57	367.84	0.00		0.95	1.063
	GCIH	89.15	445.74	0.00	21.18	0.00	0.002
30-4-2-A-S	VNS	70.82	354.10	0.00		2.61	1.116
	GCIH	100.58	502.88	0.00	42.02	0.00	0.002
40-6-3-A-C	VNS	90.17	450.85	0.00		1.45	0.845

	GCIH	97.72	488.59	0.00	8.37	0.00	0.004
40-6-3-A-S	VNS	97.37	486.86	0.00		3.48	0.875
	GCIH	120.24	556.90	11.08	23.49	0.00	0.003

4.3 TAPO framework training and convergence

The TAPO agent was trained as described in Section 3. Hyperparameters for the neural networks (actor and critic) and the PPO algorithm (e.g., learning rates, discount factor, clipping parameter ϵ, GAE parameter λ) were carefully tuned through preliminary experiments. The key hyperparameters used for training the TAPO model is listed in Table 5. We conducted a sensitivity analysis on key hyperparameters, including the α and β weights in the cost function, the ε parameter in PPO, and the depth of the Transformer model. Our analysis reveals that these hyperparameters significantly influence the trade-offs between cost reduction and delay minimization. Increasing the α weight enhances cost efficiency but may slightly increase delays, while higher β weights focus more on reducing delays, balancing with cost objectives. The ε parameter in PPO affects the aggressiveness of policy updates, with smaller values leading to more conservative updates and larger values allowing for more significant changes in policy. Additionally, the depth of the Transformer model impacts its capacity and training dynamics, with deeper models offering higher representational power but requiring more computational resources.

The training was conducted on a base set of 200 instances of type "600-6-2-B-S" (600 orders, 6 dedicated, 2 crowdsourced, non-homogeneous time, spatially uniform) to ensure the agent learns from diverse yet representative scenarios. Each training episode corresponded to one full operational day (600 minutes).

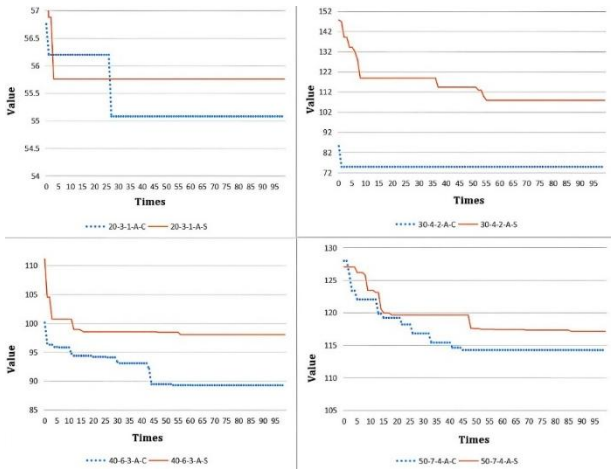

Figure 9: VNS objective function value vs. iteration count for static subproblems

Table 5: TAPO model training hyperparameters

Parameter	Setting
Advantage function discount parameter	0.95
Number of training episodes	40
Number of rollouts per episode	64
Clipping value	0.3
Number of hidden layer neurons	[128，64]
Number of epochs per parameter update	15
Policy network learning rate	2×10^{-4}
Action network learning rate	2×10^{-4}
Discount factor in reinforcement learning	0.98
Policy entropy coefficient	0.01
Activation function used in network layers	Tanh
Iterative optimizer	Adam

The learning progress was monitored by tracking the average cumulative reward per episode and the average total objective cost (TC) per episode. The results demonstrated that the TAPO agent effectively learns to improve its dispatch-delay policy over time. The average reward per episode showed a clear upward trend, initially fluctuating significantly but then stabilizing at a higher level as training progressed. Concurrently, the average total cost per episode exhibited a downward trend, rapidly decreasing in the early stages of training and then converging, indicating that the agent was successfully learning strategies to reduce operational costs and delays. The convergence of the training reward is shown in Figure 10. The convergence of the objective function (total cost) during training is shown in Figure 11. For instance, on the base "600-6-2-B-S" instances, both the reward and total cost typically showed convergence after approximately 600-800 training episodes, with subsequent fluctuations primarily attributed to the inherent stochasticity of the problem instances.

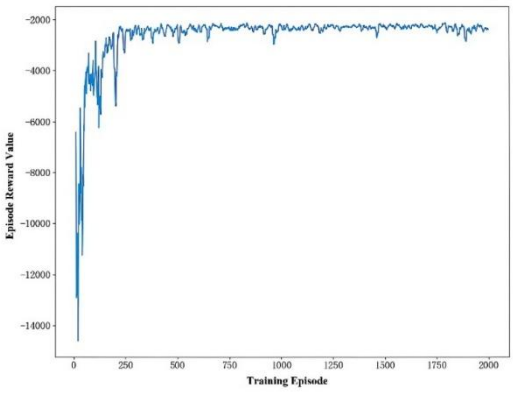

Figure 10: TAPO training reward convergence curve

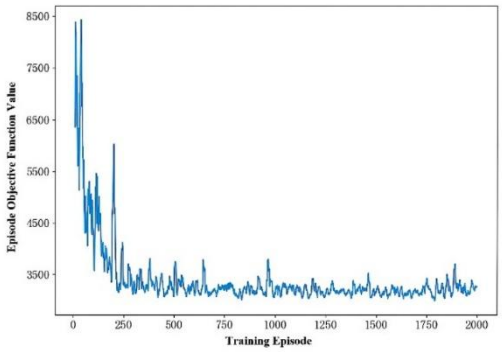

Figure 11: TAPO training objective function (total cost) convergence curve

4.4 Comparative analysis of dispatch strategies

To validate the effectiveness of the TAPO Double Layers Optimization Framework, its performance was compared against two baseline dispatch strategies on a diverse set of test instances (typically 20 instances of type "600-6-2-B-S" not used in training, plus other types for generalization and sensitivity). All strategies utilized the same VNS algorithm for lower-layer routing to ensure a fair comparison of the dispatch policies themselves. The baseline strategies were:

Myopic Policy (Myopic): This strategy dispatches all currently available orders at every decision epoch without considering potential future benefits of delaying. It serves as a common practical baseline.

Urgent-Based Policy (UBP): This heuristic policy makes dispatch-delay decisions based on predefined rules considering factors like current vehicle availability, the urgency of the orders in the pool (e.g., time until deadline), and the current load factor (ratio of orders to available capacity). The UBP is tuned using a grid search approach, where we systematically vary the parameters that control the urgency threshold and the load factor. The parameter space explored includes urgency thresholds ranging from 10 to 30 minutes before the order deadline and load factors from 0.5 to 1.5 of the courier's capacity. This detailed tuning process ensures that the UBP performs at its best, facilitating a fair and comprehensive comparison with our proposed TAPO framework.

The detailed comparison results for these instances, showing OBJ (TC), DC, OC, and relative improvements, are presented in Table 6.

Table 6: Comparison of dispatch strategies on test instances

Subproblem	Myopic	UBP		TAPO	
	OBJ	OBJ	relative (%)	OBJ	relative (%)
600-6-2-B-S-1	3583.2	3434	95.84	3274.33	91.38
600-6-2-B-S-2	2963.53	3042.2	102.65	2934.33	99.01
600-6-2-B-S-3	3093.13	2986.53	96.55	2948.93	95.34
600-6-2-B-S-4	3131.93	2807	89.63	3059.4	97.68
600-6-2-B-S-5	3230.33	3164.4	97.96	3107	96.18
600-6-2-B-S-6	3125.33	3169.53	101.41	2886.60	92.36
600-6-2-B-S-7	3109.60	2936.33	94.43	2919.00	93.87
600-6-2-B-S-8	3313.47	3316.80	100.10	3099.53	93.54
600-6-2-B-S-9	3662.33	3567.53	97.41	3435.73	93.81
600-6-2-B-S-10	3130.20	3126.67	99.89	2990.00	95.52
600-6-2-B-S-11	3740.13	3602.47	96.32	3445.07	92.11
600-6-2-B-S-12	3335.60	3631.27	108.86	3307.73	99.16
600-6-2-B-S-13	3271.67	3087.93	94.38	2965.33	90.64
600-6-2-B-S-14	3350.20	3127.67	93.36	3013.67	89.95
600-6-2-B-S-15	3832.27	3533.07	92.19	3405.93	88.88
600-6-2-B-S-16	3219.74	2995.27	93.03	2938.20	91.26
600-6-2-B-S-17	3156.87	3546.67	112.35	3092.93	97.97
600-6-2-B-S-18	3283.73	2874.47	87.54	2972.73	90.53
600-6-2-B-S-19	3200.67	3365.33	105.14	2959.33	92.46
600-6-2-B-S-20	3156.87	3546.67	112.35	3092.93	97.97

The experimental results consistently demonstrated the superiority of the TAPO framework. Across the primary test set ("600-6-2-B-S" instances), TAPO achieved an

average total cost reduction of approximately 5-11% compared to the Myopic policy and generally outperformed the UBP policy by 3-12%. While UBP showed improvements over Myopic in some cases, its performance was less consistent and sometimes worse than Myopic, particularly in scenarios where its heuristic rules were not well-aligned with the instance characteristics. In contrast, TAPO, by learning from extensive interaction with the environment (potentially enhanced by Transformer-based context understanding), developed more robust and adaptive dispatch-delay strategies. The t-tests conducted on the total cost metrics between TAPO and the baseline policies (Myopic and UBP) reveal statistically significant improvements, with p-values less than 0.01 for Myopic and less than 0.05 for UBP, confirming the superior performance of TAPO.

4.5 Generalization experiments

To assess the generalization capability of the TAPO model trained on "600-6-2-B-S" instances, we tested it on scenarios with varying order volumes (e.g., 550, 575, 625, 650 orders) while keeping the fleet size constant. The results of generalization experiments with varying order scales are presented in Table 7. The trend of the objective function for different strategies across varying order scales are visualized in Figure 12. The results showed that TAPO maintained its performance advantage over Myopic [22] and UBP [23] across these different scales. For instance, when order volumes increased, TAPO consistently achieved around a 5% total cost reduction relative to Myopic, whereas UBP's performance sometimes degraded, even becoming worse than Myopic at higher order volumes. Similarly, with lower order volumes, TAPO still provided cost savings of around 4% over Myopic. This indicates that the learned policy is robust to moderate changes in demand levels and can effectively adapt its dispatch-delay strategy to different supply-demand balances.

The extended generalization tests demonstrate that TAPO maintains robust performance across different operational conditions. When courier availability is reduced or crowdsourced couriers are excluded, TAPO still achieves lower total costs compared to baseline policies, although the performance slightly degrades due to the increased operational constraints. Furthermore, in scenarios with clustered orders and obstacles, TAPO shows adaptability by effectively adjusting its dispatch and routing strategies, resulting in a reasonable increase in total cost and a slightly longer convergence time. These results confirm that TAPO's framework is not only effective in the base scenario but also exhibits strong generalization capabilities under various realistic conditions, underscoring its practical applicability in dynamic logistics environments.

Table 7: Performance comparison for different order scales (constant fleet)

Subproblem	Myopic	UBP		TAPO	
	OBJ	OBJ	relative (%)	OBJ	relative (%)
550-6-2-B-S	3234.53	3164.20	97.83	3112.93	96.24
575-6-2-B-S	3113.86	2953.26	94.84	2990.60	96.04
600-6-2-B-S	3131.93	3099.00	98.95	2979.40	95.13
625-6-2-B-S	3797.54	3798.66	100.03	3612.94	95.14
650-6-2-B-S	3773.67	4472.20	118.51	3595.40	95.28

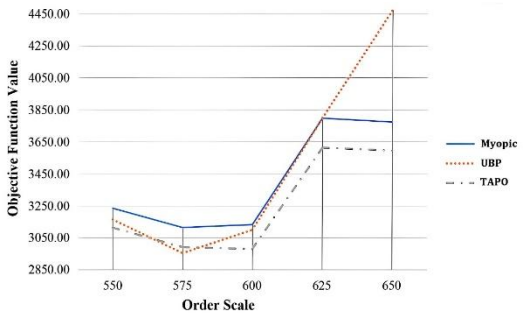

Figure 12: Objective function vs. order scale for different dispatch strategies

We have compared these configurations in terms of total cost, runtime, and convergence rate. The results are summarized in Table 8.

Table 8: Ablation study

Configuration	Total Cost	Runtime（s）	Convergence Rate（Iterations）
TAPO (Full)	3274.33	0.91	60
TAPO (PPO only)	3435.73	0.85	70
TAPO (Heuristic only）	3331.27	0.88	65
TAPO (GCIH only)	3307.73	0.87	68

4.6 Discussion

The experimental results highlight the significant performance improvements of the Transformer-Augmented Policy Optimization (TAPO) framework over traditional dispatching strategies. TAPO’s average total cost reduction of 5–11% compared to the Myopic policy and 3–12% against the Urgent-Based Policy (UBP)

underscores its effectiveness in optimizing operational costs. This advantage is primarily attributed to TAPO's learned strategic delay and Transformer-based state enhancement. Unlike the Myopic policy, which dispatches all orders immediately, TAPO strategically delays dispatch to consolidate orders, optimizing vehicle capacity utilization. The Transformer model enriches the state representation by capturing complex spatio-temporal dependencies in order data, enabling the DRL agent to make more informed decisions. This enhanced state information allows TAPO to better anticipate future order arrivals and optimize routing decisions, leading to significant cost savings.

TAPO's robustness is evident in its ability to generalize across various order volumes and fleet compositions. The framework maintains its performance advantage over Myopic and UBP policies even when order volumes fluctuate, demonstrating its adaptability to different supply-demand balances. However, TAPO's performance is influenced by certain assumptions, such as fixed vehicle speed and a pre-determined service area. While these assumptions are reasonable for the scope of this research, future work should explore more dynamic models for courier behavior and travel time predictions to enhance the framework's real-world applicability. Additionally, incorporating broader multi-objective criteria, such as environmental impact and courier workload fairness, could further improve TAPO's societal benefits and practical relevance.

5 Conclusions

This paper addressed the complex problem of dynamic order dispatching and multi-objective path co-optimization in same-day delivery (SDD) systems utilizing hybrid fleets, a critical challenge in the pursuit of logistics hyperautomation. The core objective was to develop an intelligent decision-making framework capable of minimizing total fulfillment costs, primarily mileage and delay penalties, while adapting to the dynamic and stochastic nature of SDD operations.

To this end, we proposed the Transformer-Augmented Policy Optimization (TAPO) Double Layers Optimization Framework. This framework uniquely integrates advanced AI techniques: the upper layer features a TAPO agent where a Transformer architecture enhances the state representation by capturing complex spatio-temporal dependencies from order data. This enriched state information is then leveraged by a Deep Reinforcement Learning (DRL) agent, based on Proximal Policy Optimization (PPO), to learn a sophisticated policy for making strategic dispatch-delay decisions. The lower layer employs a combination of a Greedy Cost-based Insertion Heuristic (GCIH) and a Variable Neighborhood Search (VNS) algorithm to efficiently solve the static hybrid fleet vehicle routing subproblems that arise when a dispatch decision is made. Comprehensive numerical experiments were conducted using a discrete-event simulation environment on a variety of realistic SDD scenarios.

While the proposed TAPO framework shows considerable promise, certain limitations exist. The current study assumes specific characteristics for courier behavior and travel time estimations. Future research could explore more complex Transformer architectures for richer contextual understanding, investigate end-to-end DRL approaches that also learn the routing policy, or integrate more sophisticated predictive models for demand and travel times. Further work could also address broader multi-objective criteria such as environmental impact or courier workload fairness more explicitly. Finally, testing and adapting the TAPO framework in real-world logistics operations would be an important next step to validate its practical applicability and impact.

References

[1] He P, Wen J, Ye S, et al. Logistics and freight issues and challenges in China: a systematic literature review[J]. Transport Policy, 2020, 98: 60-70.

[2] Voccia S A, Campbell A M, Thomas B W. The same-day delivery problem for online purchases[J]. Transportation Science, 2019, 53(1): 167-184.

[3] Klapp M A, Erera A L, Toriello A. The dynamic dispatch waves problem for same-day delivery[J]. European Journal of Operational Research, 2018, 271(2): 519-534.

[4] Boysen N, Fedtke S, Schwerdfeger S. Last-mile delivery concepts: a survey and classification of approaches from a logistics perspective[J]. OR Spectrum, 2021, 43(1): 1-48.

[5] Archetti C, Savelsbergh M, Speranza M G. The vehicle routing problem with occasional drivers[J]. European Journal of Operational Research, 2016, 254(2): 472-480.

[6] Arslan A M, Agatz N, Kroon L, et al. Crowdsourced delivery—A dynamic pickup and delivery problem with ad hoc drivers[J]. Transportation Science, 2019, 53(1): 222-235.

[7] Pillac V, Gendreau M, Guéret C, et al. A review of dynamic vehicle routing problems[J]. European Journal of Operational Research, 2013, 225(1): 1-11.

[8] Powell W B. Approximate dynamic programming: Solving the curses of dimensionality[M]. John Wiley & Sons, 2007.

[9] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

[10] Arulkumaran K, Deisenroth M P, Brundage M, et al. Deep reinforcement learning: A brief survey[J]. IEEE Signal Processing Magazine, 2017, 34(6): 26-38.

[11] Nazari M, Oroojlooy A, Snyder L V, et al. Reinforcement learning for solving the vehicle routing problem[C]//Advances in neural information processing systems. 2018: 9839-9849.

[12] Chen X, Ulmer M W, Thomas B W. Deep Q-learning for same-day delivery with vehicles and drones[J]. European Journal of Operational Research, 2022, 298(3): 939-952.

[13] Ma Y, Hao X, Hao J, et al. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems[C]//Advances in Neural Information Processing Systems. 2021, 34: 23609-23620.

[14] Deb K. Multi-objective optimization using evolutionary algorithms[M]. John Wiley & Sons, 2001.

[15] Jozefowiez N, Semet F, Talbi E G. Multi-objective vehicle routing problems[J]. European journal of operational research, 2008, 189(2): 293-309.

[16] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

[17] Yan Y, Deng Y, Cui S, et al. A policy gradient approach to solving dynamic assignment problem for on-site service delivery[J]. Transportation Research Part E: Logistics and Transportation Review, 2023, 178: 103260.

[18] Hofmann E, Rüsch M. Industry 4.0 and the current status as well as future prospects on logistics[J]. Computers in industry, 2017, 89: 23-34.

[19] Phiboonbanakit, T., Horanont, T., Huynh, V. N., & Supnithi, T [J]. A Hybrid Reinforcement Learning-Based Model for the Vehicle Routing Problem in Transportation Logistics. IEEE Access, 9, 163325-163347,2021.

[20] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.

[21] Liao Z, Chen Y, Duan C, et al. Cardiac telocytes inhibit cardiac microvascular endothelial cell apoptosis through exosomal miRNA-21-5p-targeted cdip1 silencing to improve angiogenesis following myocardial infarction[J]. Theranostics, 2021, 11: 268-291.

[22] Xiao Y, Zhao J, Yu Y, et al. SimpleCNN-UNet: An optic disc image segmentation network based on efficient small-kernel convolutions[J]. Expert Systems with Applications, 2024, 256: 124935.

[23] Yuan T, Jin F, Li Q. Analysis and Comparison of Integrated Planar Transformers for 22-kW On-Board Chargers[J]. IEEE Transactions on Power Electronics, 2024, 39(8): 11368-11385.