Truck-Drone Collaborative Path Planning in Large-Scale Urban Scenarios

1st Jinghang Huang, Wencan Hong, Biao Xu, Wenji Li, zhun Fan
dept. College of Engineering
Shantou University
Shantou, China
email@stu.edu.cn

2nd Dunwei Gong
dept. School of Automation and Electronic Engineering
Qingdao University of Science and Technology
Qingdao, China
email@qust.edu.cn

***Abstract*—With the vigorous development of e-commerce, the timeliness requirements for logistics distribution have been continuously elevated. Customers' specific demands for delivery time windows have rendered it crucial to complete delivery services within the specified timeframes. However, in large-scale urban scenarios characterized by dense customer distribution and complex delivery environments, traditional distribution models are struggling to meet the demands for efficient and on-time logistics. Additionally, delivery time windows, as hard constraints, narrow the scope of feasible solutions and increase the difficulty of path planning. To address these challenges, this study explores the Truck-Drone Collaborative Delivery Problem with Multiple Time Windows (TDCDP-MTW) and its solution algorithms. A hybrid distribution model is proposed, which categorizes customers into two types: dense and sparse. For dense customer clusters, fixed trucks are employed as UAV (Unmanned Aerial Vehicle) deployment centers; for sparse customer areas, trucks carry UAVs for delivery. This approach enhances UAV utilization and delivery efficiency. Meanwhile, by integrating deep reinforcement learning and attention mechanism, an encoder-decoder architecture is constructed to capture spatiotemporal dependencies. This architecture is applied to optimize path planning schemes in large-scale customer scenarios, improving computational efficiency and generalization ability. Experimental results demonstrate that this study provides a collaborative path planning solution that balances efficiency and flexibility for high-density urban logistics, and resolves the coupling dilemma between multi-objective optimization and dynamic decision-making in large-scale scenarios.**

***Index Terms*—truck-drone collaborative delivery; multi-objective optimization; deep reinforcement learning; path planning**

I. INTRODUCTION

The application of truck-drone collaborative delivery has covered multiple fields in production and daily life [1]. This collaborative model is commonly referred to as the "truck-drone system" [2], which has been extensively researched and applied in the logistics field [3]. For instance, in disaster relief, the synchronous delivery mode of trucks and drones combines the large capacity of trucks with the speed and flexibility of drones, enabling the rapid transportation of emergency supplies to disaster-stricken areas [4]. For target monitoring tasks, a truck carries multiple drones to a designated location and deploys them for real-time monitoring of multiple targets. After completing their tasks, the drones return to the truck [5]. In last-mile logistics, the collaborative delivery of trucks and drones is explored by considering both route selection and time windows to optimize the delivery process [6]. These applications highlight the potential advantages of the collaboration between trucks and drones in various fields [7].

In recent years, with the rapid development of e-commerce, people's requirements for the timeliness of logistics delivery have been continuously increasing. For example, some customers may have specific requirements for delivery time points. To meet such customer needs, it is necessary to complete delivery services within the specified time windows. Therefore, it is essential to incorporate delivery time windows into the specific model constraints. References [8] and [9] consider completing delivery services within given time windows and improve delivery efficiency through optimization algorithms, achieving the goal of completing deliveries as soon as possible while meeting customer requirements. Maghfiroh et al. [10] discussed the truck-drone path planning problem involving time windows and improved customer satisfaction and operational efficiency by balancing the minimization of travel distance and the satisfaction of time window constraints. The aforementioned studies on truck-drone collaborative systems with time windows only limit the earliest or latest service execution time through additional constraints. It can be seen that delivery time windows are treated as hard constraints of the model and must be satisfied. On the one hand, the increase in hard constraints narrows the feasible region of the problem, leading to difficulties in solution search; on the other hand, in actual delivery, violations of delivery time window constraints often occur due to the impact of external uncertain factors. Therefore, it is more reasonable to evaluate services under scenarios that allow for time window violations.

To address these challenges and considering the characteristic of concentrated customer distribution in urban scenarios, customers are divided into two categories: dense and sparse, and different delivery methods are adopted for these two types of customers. In dense customer areas, frequent short-distance movements of trucks are unfavorable for the recovery and subsequent launch of drones. In this case, trucks are more suitable for remaining stationary as mobile deployment centers for drones, thereby improving the utilization rate of drones. However, the deployment of trucks requires time; thus, for areas with sparse customer density, the mode of trucks carrying drones for joint delivery is adopted. Therefore, this hybrid truck-drone collaborative delivery mode is more suitable for high-density urban delivery environments while also meeting the delivery needs of low-density areas.

In recent years, the attention mechanism has been widely applied in the Traveling Salesman Problem (TSP). Peng et al. [11] successfully solved the Vehicle Routing Problem by combining the attention model with deep reinforcement learning algorithms, and their algorithm demonstrated excellent generalization ability. Compared with traditional heuristic algorithms, the attention mechanism can help the model focus on key information, thereby significantly improving computational efficiency. Based on this advantage, this study will adopt the method of combining deep reinforcement learning with MONSNSA to solve the truck-drone collaborative path planning problem in large-scale customer scenarios.

II. PROBLEM DESCRIPTION

In densely populated areas, by deploying trucks as mobile UAV (Unmanned Aerial Vehicle) service centers, the advantages of UAVs in flexibility and efficiency for "last-mile" delivery can be fully exploited. As shown in Figure 1, the coverage area of each service center is determined by the maximum flight distance of UAVs, with a radius set to half of the maximum flight distance. This deployment strategy enables UAVs to efficiently serve customers within their coverage areas, and then enter a collaborative delivery mode with trucks to jointly complete delivery tasks in the surrounding areas before moving to the next service center location.

A. Time Window Model

The definition of customer satisfaction is as follows: the satisfaction function $\mu_i(t)$ serves as an indicator that meets customer expectations and is related to the arrival time of delivery services. If the service time of customer i falls within the time window $[a_i, b_i]$, there is no loss of satisfaction. However, when the arrival time of the delivery service is within the intervals $[e_i, a_i]$ and $[b_i, l_i]$, the satisfaction will change linearly with the arrival time.

$$\mu_i(t) = \begin{cases} \frac{t-b_i}{l_i-b_i}, & b_i < t \leq l_i \\ 1, & a_i \leq t \leq b_i \\ \frac{t_i-e_i}{a_i-e_i}, & e_i \leq t < a_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

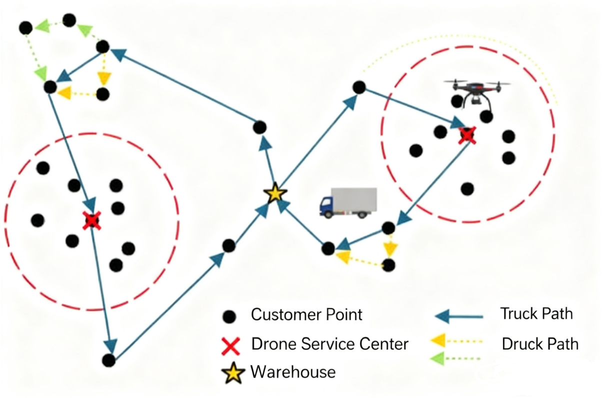

Fig. 1. Schematic Diagram of Truck-UAV Collaborative Delivery Mode.

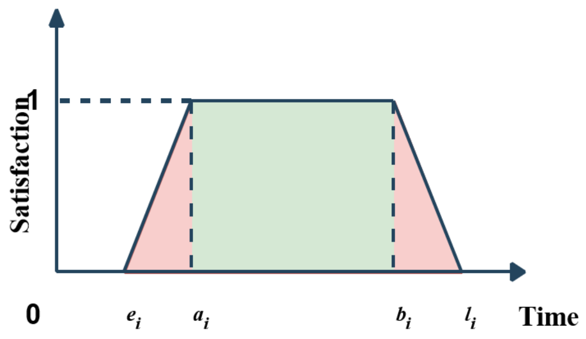

Fig. 2. Schematic Diagram of Customer Time Windows.

B. Objective Model

In the discussion of this chapter, considering the scenario introduced in Chapter 4 where a UAV service center serves customers, it is essential to emphasize the operational mechanism of the system. As the UAV service center adopts an operational model using fixed trucks as bases, the UAVs will provide delivery services to surrounding customers with these trucks as the center. This configuration offers significant operational advantages. In terms of cost analysis, the delivery cost primarily consists of two core components: the first is the collaborative routing cost between the truck and the UAV, $Cost_{collab}$, which includes the cost of the truck moving to the service center positioning point and the collaborative operational cost between the UAV and the truck, $Cost_{DS}$; the second is the flight cost of the UAV from the service center, which mainly involves the flight distance and energy consumption from the service center to the customer points.

$$\begin{aligned}\text{Cost}_{\text{collab}} =& \sum_{i\in N}\sum_{j\in N} c_t d_{ij} x_{ij} \\ &+ \sum_{d\in D}\sum_{k\in V}\left[\sum_{i\in C_L}\sum_{j\in C_L} c_d d'_{ij} y^{dk}_{ij}\right] \\ &+ \sum_{i\in N}\sum_{j\in C_L}(c_d d'_{ij} + c_l) a^{dk}_{ij} \\ &+ \sum_{i\in C_L}\sum_{j\in N} c_d d'_{ij} b^{dk}_{ij}\end{aligned} \tag{2}$$

This cost model is composed of three main components: first, $c_d d'_{ij} y^d_{ij}$ represents the basic flight cost of the UAV between customer points i and j, where c_d is the unit distance flight cost coefficient, d'_{ij} is the actual flight distance, and y^d_{ij} is the path selection decision variable; second, $(c_d d'_{ij} + c_l)a^d_{ij}$ denotes the additional takeoff cost, c_l is the additional cost coefficient for the takeoff phase, and a^d_{ij} is the takeoff decision variable; finally, $c_d d'_{ij} b^d_{ij}$ indicates the landing cost, and b^d_{ij} is the landing decision variable. Therefore, the total delivery cost can be expressed as:

$$f_1 = \text{Cost}_{\text{collab}} + \sum_{k\in N_{DS}} \text{Cost}^k_{DS} \tag{3}$$

where N_{DS} represents the total number of UAV service centers, and the sum of the costs of all UAV service centers can be expressed as:

$$\sum_{k\in N_{DS}} \text{Cost}^k_{DS} \tag{4}$$

In the analysis of customer satisfaction, it is necessary to take into account the differences in service area density and service methods. Thus, the customer satisfaction model can be divided into two main parts: the satisfaction under the service of UAV service centers, and the satisfaction under the truck-drone collaborative service in sparse areas. For the service areas of UAV service centers, customer satisfaction mainly depends on whether UAVs can complete services within the time windows specified by customers. The satisfaction formula for customers served by the truck-drone collaborative delivery mode can be expressed as follows:

$$\begin{aligned}\text{Sat}_{\text{collab}} =& \sum_{i\in N}\sum_{j\in N} x_{ij} u_j(r_j) + \\ & \sum_{d\in D}\sum_{k\in V}\sum_{i\in N}\sum_{j\in N}(a^{dk}_{ij} + b^{dk}_{ij} + y^{dk}_{ij}) u_j(r_j)\end{aligned} \tag{5}$$

The sum of satisfaction of all customers can be expressed as:

$$f_2 = Sat_{collab} + \sum_{k\in N_{DS}} Sat^k_{DS} \tag{6}$$

Specifically, $\sum_{k\in N_{DS}} Sat^k_{DS}$ represents the total satisfaction of all customers independently served by UAV service centers, which reflects the service quality of UAVs in dense areas, and Sat_{DS} denotes the sum of satisfaction of all customers under a single UAV service center. Sat_{collab} characterizes the total satisfaction of customers served by the collaborative mode in sparse areas.

This chapter also focuses on the UAV path optimization problem based on reinforcement learning in large-scale urban scenarios, aiming to construct a bi-objective optimization model: on the one hand, minimizing the maximum operational cost, and on the other hand, maximizing the overall customer satisfaction. To effectively guide resource allocation decisions, this chapter unifies these two mutually restrictive objectives into a single optimization framework by constructing a reward function, thereby achieving the overall optimization of system performance.

The planning process of UAV service paths can be represented by the Markov Decision Process (MDP). First, we define the state vector $s_t = (v_t, c_t, r_t)$. Here, v_t represents the customer points that have been visited before step t. c_t represents the current location of the drone at a customer point. r_t represents the remaining endurance of the drone for this flight. At each step t, the decision-maker, based on the current state, selects the next action a_t, which is the next customer the drone will fly to. When the drone arrives at the next customer,the path cost incurred in this process is $Cost_t$, and the satisfaction obtained after the UAV serves the customer is Sat_t. The visited customer point is incorporated into v_t.

In the scenario of multi-drone service centers, modeling the flight cost of drones requires consideration of multiple key factors. Assume there are multiple drone service centers in a certain area. A drone service center can be denoted as DS, , and the set of customers served by the k-th service center is denoted as C^k_{DS}. $Cost_{DS}$ represents the maximum operational cost within the drone service center, and Sat_{DS} represents the overall satisfaction of customers served by the drone service center. The flight cost $Cost_{DS}$ for a single drone service center can be modeled as follows:

$$\text{Cost}_{DS} = \sum_{t\in C_{DS}} \text{Cost}_t \tag{7}$$

Under this model, due to the relatively concentrated distribution of customers, the response time and service efficiency of drones are typically high. The satisfaction level of the drone service centers can be modeled as the sum of satisfaction levels across all drone service centers, which can be specifically expressed as:

$$Sat_{DS} = \sum_{t\in C_{DS}} Sat_t \tag{8}$$

In fact, there is often a trade-off between these two objectives: excessively pursuing cost minimization may lead to a decline in service quality, while blindly pursuing the improvement of customer satisfaction may cause a sharp surge in operational costs. Based on the above analysis, this chapter defines the reward function as:

$$R = -\alpha \cdot \log \frac{\mathrm{Cost}_{DS}}{\beta} + (1-\alpha)\frac{\mathrm{Sat}_{DS}}{C_{DS}} \quad (9)$$

Among these, α and β are weighting coefficients for cost and satisfaction, respectively. Their values reflect the decision-maker's relative emphasis on the two objectives. β is used to control the sensitivity to cost.

III. MONSNSA_RL FRAMEWORK

This section elaborates on the Multi-Objective Drone Service Network Scheduling Algorithm combined with Deep Reinforcement Learning (MONSNSA_RL). The core innovation of this algorithm lies in integrating the Multi-head Attention Mechanism with the Gated Recurrent Unit (GRU) and applying this combination within the MONSNSA framework to effectively enhance its efficiency in handling large-scale truck-drone collaboration problems.

Specifically, during the encoding phase, the algorithm draws on the Multi-head Attention Mechanism proposed by Kool et al. (2018). This structure enables the simultaneous capture of spatial correlation information between different drone service centers. During the decoding phase, a GRU is employed for sequential decision-making. The memory characteristics of its hidden state effectively handle the temporal dependencies between successive actions.

The design of this encoder-decoder architecture stems from the importance of relative positional information for path planning in the Traveling Salesman Problem (TSP): the hidden state of the GRU can store historical information of past decisions, offering distinct advantages over stateless attention-based decoders when dealing with sequential decision-making problems. Research by Wu et al. (Reference [5]) also validated the effectiveness of the GRU decoder in single-vehicle routing problems, further supporting our architectural choice.

The specific implementation process of the MONSNSA_RL algorithm is shown in Table I Algorithm 4.1, which employs a phased optimization strategy to achieve coordinated truck-drone delivery route planning. First, based on Kernel Density Estimation (KDE), spatial clustering analysis is performed on the customers. Multiple high-density customer regions C_{DS} are delineated according to the geographical distribution characteristics of the customers. Each region defines a service area centered around a specific customer point acting as a drone service center. This partitioning method considers the actual distribution characteristics of customers in the real-world scenario (Line 2). Simultaneously, a set C_{collab} is defined, which includes other customer points not assigned to any C_{DS} and the customer points serving as drone service centers. These points will utilize the truck-drone collaborative delivery mode.

Next, initial routes are created using the nearest-neighbor method. Route planning for the truck-drone collaborative delivery is performed for the customer points in C_{collab}, generating the initial population P_0 (Line 3). It is important to note that in this phase, only the customer points in C_{collab} are processed, excluding those within the C_{DS} regions. Subsequently,

TABLE I
ALGORITHM 4.1 MONSNSA_RL

```
Input: Population size $m$, map size $s$, initial customers $C$,
maximum iterations $Max-iter$
Output: Pareto solution set $PF$
1: Initialize: $PF \leftarrow \emptyset$
2: $C_{collab}, C_{DS} \leftarrow$ KDE partition drone service centers($C$)
3: $P_0 \leftarrow$ Create initial paths($m, s, C_{collab}$)
4: $t \leftarrow 1$
5: while $t \leq Max-iter$ do:
6:     $Q_{t-1} \leftarrow GVNS(P_{t-1})$
7:     $Q_{t-1} = TransGRU(Q_{t-1}, C_{DS})$
8:     $Q_{t-1} \leftarrow Repair(Q_{t-1})$
9:     $P_{all} \leftarrow P_{t-1} \cup Q_{t-1}$
10:    $F = Non-dominated-sort(P_{all})$
11:    $F = EliteSelection(F)$
12:    $P_L \leftarrow LocalSearch(F)$
13:    $P_L \leftarrow Repair(P_L)$
14:    $P_{all} \leftarrow F \cup P_L$
15:    $P_{all} \leftarrow RemoveDuplication(P_{all})$
16:    $F = Non-dominated-sort(P_{all})$
17:    $P_t = EliteSelection(P_t)$
18:    $t = t + 1$
19: end
20: $PF \leftarrow F$
```

the customer points in C_{collab} within the population P_0 are optimized through neighborhood transformation operations, generating a new population Q_{t-1}. This step aims to enhance the initial population diversity, providing a better starting point for subsequent optimization searches.

After planning the routes for the C_{collab} customer points, the TransGRU model is used to plan the routes for the drone service centers C_{DS} within the population Q_{t-1} (Line 7). The model takes the coordinate locations and time windows of the customers as input and outputs the optimal drone route planning scheme for each drone service center. The TransGRU model combines the attention mechanism of the Transformer with the sequence modeling capability of the GRU, enabling it to capture both spatial correlations and temporal dependencies, thereby generating route plans that better align with practical requirements.

Upon completing the route planning for both the C_{collab} and C_{DS} customer groups, the algorithm enters the iterative optimization phase. Its core lies in improving solution quality through a multi-strategy optimization mechanism. Specifically, repair operations are first performed on the delivery routes for the C_{collab} customers (Lines 8-11). This step ensures that the generated routes satisfy all constraint conditions.

Subsequently, the newly generated populations P_{t-1} and Q_{t-1} are merged, and a non-dominated sorting strategy is applied to evaluate the combined population. This provides a high-quality starting point for subsequent optimization (Line 12). Building upon this, a local search is performed on the non-dominated solution set F (Lines 13-14), exploring strategies such as neighborhood search to seek potential superior solutions. This step aids in escaping local optima and enhancing the global quality of the solutions. Following this, a repair operation is conducted on the new solution P_L obtained from the local search, and it is merged with the non-dominated solution set to form a new candidate solution set P_{all} (Line 15). To ensure population diversity, duplicate solutions are removed from P_{all}. This strategy helps prevent the population from becoming homogenized and maintains the exploration capability within the search space. Finally, non-dominated sorting and elite selection are performed on the population (Line 16), retaining the best individuals from the current iteration as the input solutions for the next iteration. This elite retention strategy ensures the convergence of the algorithm.

After completing the route planning for both the C_{collab} and C_{DS} customer groups, the algorithm enters the iterative optimization phase, aiming to enhance solution quality through a multi-strategy optimization mechanism. Specifically, repair operations are first performed on the delivery routes for the C_{collab} customers (Lines 8-11) to ensure the generated routes satisfy all constraints, thereby eliminating infeasible solutions. Subsequently, the newly generated populations P_{t-1} and Q_{t-1} are merged, and the combined population is evaluated using a non-dominated sorting strategy (Line 12). A local search is then conducted on the non-dominated solution set F, employing strategies such as neighborhood exploration to seek superior solutions, which helps escape local optima and improves the global quality of the solutions (Line 13). The new solutions P_L obtained from the local search undergo repair operations and are merged with the non-dominated solution set to form a new candidate solution set P_{all} (Lines 14-15). To maintain population diversity, duplicate solutions are removed from P_{all}. This strategy prevents population homogenization and preserves the exploration capability within the search space (Line 16). Finally, non-dominated sorting and elite selection are performed on the population (Lines 17-18), retaining the best individuals from the current iteration as input solutions for the next iteration, thereby ensuring the convergence of the algorithm.

IV. TRANSGU ALGORITHM FRAMEWORK

A. Composition of the TransGRU model

Traditional Transformer models have demonstrated strong performance in time-series prediction tasks, but their effectiveness in combinatorial optimization problems still has certain limitations, primarily manifested in handling complex constraints and prediction accuracy. This paper proposes an improved TransGRU model that emphasizes the temporal correlations among customer satisfaction, time window constraints, and location information. The core innovation of the model lies in integrating the sequential decision-making capability of the Gated Recurrent Unit (GRU) with the attention mechanism of the Transformer. The memory characteristics of the GRU's hidden state effectively handle temporal dependencies between subsequent actions, while the attention mechanism captures long-range dependencies, thereby enhancing the model's performance in complex optimization problems.

In Table I Algorithm 4.1, we first employ Kernel Density Estimation (KDE) to perform spatial clustering analysis based on customer locations, selecting multiple drone service centers. To further optimize the route planning for these drone service centers, we construct a TransGRU model based on an encoder-decoder architecture, as illustrated in Figure 3. This model consists of four core modules: an embedding layer, an encoder, a decoder, and a training strategy. The embedding layer is responsible for mapping customer location information and time window information into a high-dimensional feature space. The encoder uses a multi-head attention mechanism to extract information from the embedded hidden layers. By computing multiple attention heads in parallel, the model can capture the diversity of customer features from different subspaces. The decoder, leveraging the sequential decision-making capability of the GRU, gradually generates the optimal route.

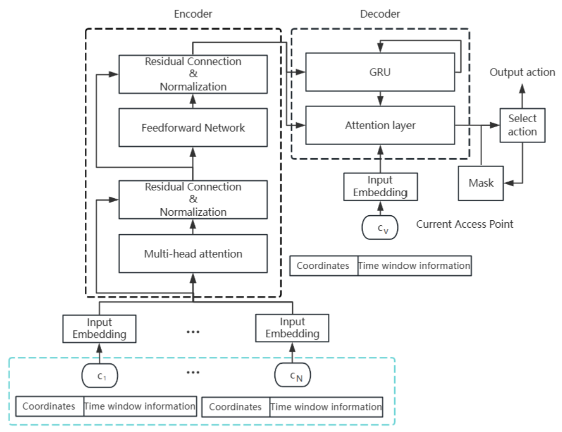

Fig. 3. TransGRU Model.

B. Encoder

The network framework begins with node embedding, whose core function is to transform the input raw information into high-dimensional vector representations, laying the foundation for subsequent feature extraction and computational processing.The location information of each customer can be represented as two-dimensional coordinates (x_i, y_i), while each customer also includes time window information, namely the start time a_i and end time b_i of the time window. To map the node's coordinate information and time window information into a high-dimensional feature space, the model performs an initial embedding operation through a linear transformation formula, expressed mathematically as:

$$h_i^0 = W[x_i; y_i; a_i; b_i] + b \tag{10}$$

Here, W and b are the trainable weight matrix and bias term, respectively, belonging to the customer set C_{DS}. This formula generates the initial embedding vector h_i^0 by combining the node's coordinate information and time window information with learnable parameters. This embedded representation not only preserves the spatial and temporal characteristics of the original data but also provides a foundation for subsequent feature extraction and processing.

The attention mechanism obtains the attention distribution by computing the Query, Key, and Value. The Query typically comes from the feature representation of the currently processed position, while the Key and Value are derived from the features at different positions of the input data. As shown in Figure xx, the attention score for each position is obtained by calculating the similarity between the Query and each Key. These scores are then normalized to produce attention weights, and finally, a weighted sum of the Value is computed based on these weights to yield the attention-based output. Given an input sequence h_i with dimension d_h, the attention mechanism uses learnable weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$ to generate the Query, Key and Value vectors:

$$\begin{cases} q_i = h_i^0 W_Q \\ k_i = h_i^0 W_K \\ v_i = h_i^0 W_V \end{cases} \tag{11}$$

This paper employs the dot product to compute attention scores $u_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}}$. The attention weights are calculated based on the similarity scores between the query vector and the key vectors at each position, which are then normalized via the Softmax function to form the attention weight distribution:

$$\alpha_{ij} = \frac{\exp(u_{ij})}{\sum^n \exp(u_{ij})} \tag{12}$$

This weight indicates the importance of the j position in the input sequence when generating the output at the i position. The final output is the weighted sum of the value vectors at all positions:

$$h_i = \sum_{j=1}^{n} \alpha_{ij} v_j \tag{13}$$

In the TransGRU model, we employ a Multi-Head Attention (MHA) mechanism to enhance the richness of feature representation. This mechanism projects the input features into M distinct subspaces, thereby capturing feature dependencies across different dimensions. For each customer node i, the computation process of the multi-head attention mechanism can be expressed as:

$$h_i' = \mathrm{MHA}(h_i) \tag{14}$$

In this context, h_i' is the feature representation processed by the multi-head attention mechanism, which integrates information from different subspaces, thereby generating a new, comprehensive feature representation for each node. The output of the multi-head attention sublayer, h_i' , undergoes a residual connection with the input features h_i, followed by batch normalization to enhance the model's training stability and feature representation capability:

$$\hat{h}_i = \mathrm{BN}(h_i + h_i') \tag{15}$$

This operation not only mitigates the vanishing gradient problem but also accelerates the model's convergence process. Subsequently, the output of the batch normalization, $\hat{h}_i$, is fed into a fully connected feed-forward network (FFN). This network consists of two linear transformations with a ReLU activation function in between:

$$\mathrm{FFN}(\hat{h}_i) = W_2 \cdot \mathrm{ReLU}(W_1 \cdot \hat{h}_i + b_1) + b_2 \tag{16}$$

Here, W_1 and W_2 are trainable weight matrices, and b_1 and b_2 are bias terms. The output of the feed-forward network again undergoes a residual connection with the input features $\hat{h}_i$, followed by batch normalization, to produce the final node feature representation:

$$\bar{h}_i = \mathrm{BN}(\hat{h}_i + \mathrm{FFN}(\hat{h}_i)) \tag{17}$$

C. Decoder

To fully utilize the node embeddings generated during the encoding phase, we employ a concatenation operation to integrate the embedding vectors of all customers into a global representation. Specifically, for each customer node i, its embedding vector $\bar{h}_i$ is combined into a global embedding vector $\bar{h}_0$ through the concatenation operation. The mathematical expression is as follows:

$$\bar{h}_0 = [\bar{h}_1; ...; \bar{h}_i] \tag{18}$$

where $\bar{h}_0$ represents the global embedding vector of all customer nodes, and $\bar{h}_i$ denotes the embedding vector of the current customer node i where the UAV is located, with $i \in C_{DS}$, preserving the individual characteristics of each node. To further model the temporal dependencies in UAV path planning, we introduce a Gated Recurrent Unit (GRU) network. The GRU integrates the current node embedding vector $\bar{h}_i$ with the hidden state H_{t-1} from the historically visited nodes by the UAV, generating the current hidden state H_t and an updated node embedding vector $\tilde{h}_i$. The computation process can be expressed as:

$$\tilde{h}_i, H_t = GRU(\bar{h}_i, H_{t-1}) \tag{19}$$

The hidden state output by the GRU and the embedding vector of the node where the UAV is currently located are processed through an attention mechanism. Additionally, the satisfaction level and travel distance from the current UAV to other reachable customer points are embedded to obtain h_{ij}, which is also fed into the attention mechanism. This process yields the attention weight α_{ij} of customer point i relative to other customer points.

Subsequently, we integrate the hidden state h_i output by the GRU with the global embedding vector h_g via the attention mechanism. At the same time, we incorporate the satisfaction level and distance information from the current UAV to other reachable customer points, embedding them into a feature vector h_{ij}. These features collectively serve as inputs to the attention mechanism to compute the attention weight $\hat{u_{ij}}$ of customer point i relative to other reachable customer points j. The specific calculation process is as follows:

$$\hat{u}_{ij} = W^a \tanh(W^\theta[\bar{h}_0; \tilde{h}_i; W^j h_{ij}]) \tag{20}$$

Here, W^a, W^θ, and W^j are trainable parameter matrices, $\bar{h}_0$ is the global embedding vector of all customer nodes, $\tilde{h}_i$ is the embedding vector of the node where the current UAV is located, and h_{ij} represents the embedding vector from customer point i to other reachable customer point j. In this context, j belongs to the set of customer points reachable by the UAV in the current state, denoted as $C_{DS} \backslash i$, i.e., the set of candidate customer points excluding the current customer point i.

The computation of attention weights considers not only the state information of the current node but also incorporates the satisfaction and distance features of the target nodes, thereby providing a more comprehensive decision-making basis for path planning. After obtaining the attention weight $\hat{a_{ij}}$, we filter the candidate customer points based on the new action and dynamically update the set of customer points reachable by the UAV at customer point j through a mask mechanism, ensuring the real-time nature of the path planning strategy. In UAV path planning, the probability $P_\theta(a_t = j|s_t)$ of the current UAV selecting the next customer point j from customer point i at step t can be represented by the following conditional probability distribution:

$$\hat{u}_{ij} = W^a \tanh(W^\theta[\bar{h}_0; \tilde{h}_i; W^j h_{ij}]) \tag{21}$$

Here, A_i denotes the set of customer points that are infeasible for customer point i, including points beyond the UAV's remaining battery range and already visited customer points. If $j \in A_i$, then the probability from customer point i to customer point j is 0, ensuring the feasibility of the path planning.

D. Model Training

To train the model more effectively, we adopt the Advantage Actor-Critic (A2C) framework, as shown in Table II Algorithm 4.3. The update of the Actor network parameters θ^a follows the policy gradient theorem:

$$P_\theta(a_t = j|s_t) = \begin{cases} \frac{\exp(\hat{u}_{ij})}{\sum_{j \in C_{DS} \setminus A_i} \exp(\hat{u}_{ij})} & : j \in C_{DS} \setminus A_i \\ 0 & : j \in A_i \end{cases} \tag{22}$$

where T represents the trajectory length, and the advantage function $A(s_t, a_t)$ is estimated by the Critic network:

$$A(s_t, a_t) = \sum_{k=t}^{T} \gamma^{k-t} r_k - V_{\theta^c}(s_t) \tag{23}$$

Within this framework, the actual training employs mini-batch stochastic gradient descent, with the batch size defined as B. The specific loss function and gradient computation methods are as follows:

$$\nabla_{\theta^a} J(\theta^a) = \frac{1}{B} \sum_{b=1}^{B} \sum_{t=0}^{T} \nabla_{\theta^a} \log \pi_\theta(a_t^b|s_t^b)(R^b - V_\theta(s^b)) \tag{24}$$

Here, the state feature s^b encodes customer coordinates and time windows $[x, y, e, a, b, l]$, and $log\pi_{\theta^a}(a_t^b|s_t^b)$ is the policy probability of the actor network selecting action a^b in the current state. R^b represents the reward obtained for the b-th batch, which is evaluated here by the critic network and denoted as V^b. The loss function for the critic can then be expressed as:

$$\text{loss}_\theta = \frac{1}{B} \sum_{b=1}^{B} \sum_{t=0}^{T} \nabla_\theta (R^b - V_\theta(s^b))^2 \tag{25}$$

This loss function is used to optimize the parameters θ^c of the Critic network, with the objective of minimizing the mean squared error between the actual reward R^b and the value $V_{\theta^c}^b(s^b)$ predicted by the Critic network.

TABLE II
ALGORITHM 4.3 MODEL TRAINING ALGORITHM

```
Input: Batch size $B$, maximum trajectory length $T$
1: Initialize: Reward $R$, initialize GRU initial state $H_0$
2: Obtain initial state $s^b$ through environment reset,
   $mask_0^b \leftarrow$ Env.Reset($B$)
3: for $b = 0$ to $B$ do
4:   for $t = 0$ to $T$ do
5:     $a_{t+1}^b, H_{t+1} \leftarrow \pi_{\theta^a}(s_t^b, mask_t^b)$
6:     $s_{t+1}, mask_t^b \leftarrow$ Env.Step($a_{t+1}^s$)
7:     $R^b \leftarrow R^b + R_t^b$
8:   Compute baseline value using Critic network: $V_{\theta^c}^b(s^b)$
9:   Compute policy gradient and update (Actor)
     $d\theta^a \leftarrow \frac{1}{B}\sum_{b=1}^{B}\sum_{t=0}^{T_b} \nabla_{\theta^a} \log \pi_{\theta^a}(a_t^b|s_t^b)(R^b - V_{\theta^c}(s^b))$
10:  Compute critic loss and update (Critic):
     $d\theta^c \leftarrow \frac{1}{B}\sum_{b=1}^{B}\sum_{t=0}^{T} \nabla_{\theta^c}(R^b - V_{\theta^c}(s^b))^2$
```

V. Experimental Design and Analysis

This chapter presents a series of experiments designed to validate the performance of the MONSNSA_RL model and evaluate its performance on the Solomon VRPTW benchmark instances. Furthermore, we establish the key parameter settings for the reinforcement learning module through experimentation. Building upon the task allocation method proposed in Chapter 3, we conduct experiments from multiple perspectives to comprehensively validate both the task allocation method and the model's performance.

A. Reinforcement Learning Module Performance Evaluation

To evaluate the performance of the reinforcement learning module in the path planning task for UAV service centers, we designed problem instances based on random generation. We selected customer locations randomly generated within a 100×100 area as the benchmark set, and generated time windows of 0 20 minutes for them. All time windows were set within the range of 8:00 to 12:00, with no restrictions on the UAV's flight distance.

During testing, we employed two methods to sample solutions from the trained hybrid model.The first method is called the Greedy Algorithm. Its core idea is to select the node with the highest visitation probability at each timestep as the next action. Although this method is computationally efficient, it may converge to local optima. The second method is called the Sampling Method, which independently samples multiple solutions from the trained model and selects the sample with the lowest cost as the final solution. By introducing randomness, this method can explore a larger solution space, thereby having a higher probability of finding the global optimum.

To evaluate the computational efficiency of the model, we measured the inference time of the reinforcement learning model on different benchmark datasets using an NVIDIA RTX 4060 GPU and an AMD 7945HX processor. During the reinforcement learning training process, we performed 10,000 iterations on instances with 25 customer nodes. Among these, random sampling was conducted with 100, 1000, and 4000 samples respectively to optimize the training results. Table III shows the performance of different methods when handling problems of varying scales (25, 50, and 100 nodes) in the UAV service center path planning problem. Experimental results indicate that as the number of samples increases, the random sampling method demonstrates significant improvements in both solution quality and customer satisfaction, but this is also accompanied by a substantial increase in computation time.

TABLE III
RESULT OF RL-GREEDY AND RL-SAMPLING ON RANDOMLY GENERATED TEST SETS

Algorithm	25 Customers		
	cost	sat	Time
RL-greedy	747.46	10.24	0.75s
RL-sampling(100)	755.23	11.13	1.24s
RL-sampling(1000)	786.09	15.38	16.40s
RL-sampling(4000)	771.31	18.10	38.08s
	50 Customers		
	cost	sat	Time
RL-greedy	812.33	36.84	1.14s
RL-sampling(100)	822.54	39.04	1.64s
RL-sampling(1000)	946.74	43.00	36.72s
RL-sampling(4000)	807.41	41.38	95.35s

Figure 4 displays the schematic diagrams of route planning generated using the RL-greedy and RL-sampling (with 1000 samples) algorithms, respectively, under the 25-node scale. In the figures, the red cross symbols represent the base station locations, the red dashed circle indicates the service range of the UAV service center, and the blue lines depict the UAV flight paths. By comparing the two figures, the differences in route planning and customer coverage between the two algorithms can be visually observed.

Although the path cost of the RL-sampling (1000) algorithm is higher than that of the RL-greedy algorithm, its customer satisfaction is significantly improved, being 43.5% higher than the RL-greedy algorithm. This result aligns with the algorithm's optimization objective: to maximize total customer satisfaction while minimizing path cost where possible, rather than solely pursuing the minimization of travel distance. By increasing the number of samples, RL-sampling (1000) explores a broader solution space, thereby achieving superior results in customer satisfaction. In contrast, although the RL-greedy algorithm offers higher computational efficiency, its performance in customer coverage and route optimization may be limited due to the constraints of its greedy strategy.

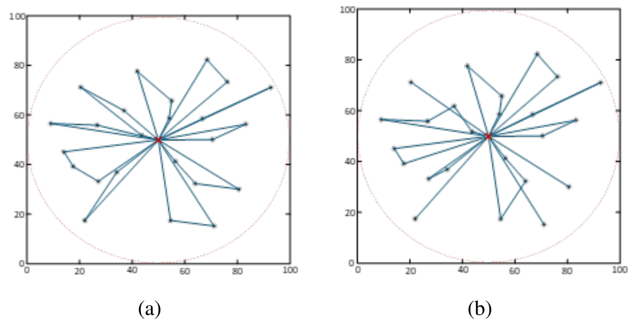

Fig. 4. TransGRU Model.

B. *Comparative Experiments with the MONSNSA Algorithm*

To validate the role of different modules in the MONSNSA_RL algorithm, we designed a series of ablation experiments. These experiments aim to evaluate the contribution of the GRU module in the truck-drone path planning problem by comparing the performance of MONSNSA_RL_noGRU, which is the MONSNSA_RL algorithm with the GRU module removed, against the complete MONSNSA_RL algorithm. The experiments incorporate the Gaussian Kernel Density Estimation (KDE) method proposed in Chapter 4 to scientifically delineate service ranges and further verify the algorithm's effectiveness in larger-scale problems.

The experiments compare the following four methods:

(1) Randomly assigned center points + MONSNSA_RL_no-GRU: This method selects service center points randomly, using the same quantity as determined by KDE, and performs path planning using the MONSNSA_RL algorithm with the GRU module removed.

(2) Randomly assigned center points + MONSNSA_RL: This method also employs randomly selected service center points but uses the complete MONSNSA_RL algorithm (including the GRU module) for path planning.

(3) KDE method + MONSNSA_RL_noGRU: This method delineates service center points based on KDE and performs path planning using the MONSNSA_RL algorithm with the GRU module removed.

(4) KDE method + MONSNSA_RL: This method combines the KDE approach to delineate service center points and employs the complete MONSNSA_RL algorithm for path planning.

In terms of experimental design, this study employs the Hypervolume (HV) as the evaluation metric to comprehensively assess the comprehensive performance of the algorithms concerning path cost and customer satisfaction. Specifically, the reference point is selected based on the maximum cost and the minimum satisfaction observed across all test results. The experimental dataset includes the RC201 instance from the Solomon benchmark set, as well as the extended instances RC2_2_1 and RC2_4_1 from the Homberger benchmark set, corresponding to test scenarios with 100, 200, and 400 customer points, respectively. This dataset encompasses customer point distributions with different characteristics. The map size is uniformly set to 100×100 to maintain consistent experimental conditions. Furthermore, soft time window constraints are adopted, with the soft time window relaxation coefficient w set to 0.25, simulating customer soft time windows in real-world scenarios.

In the experimental setup, the UAV's flight speed is set to twice that of the truck. The configuration utilizes two trucks, each carrying five UAVs. The unit movement cost for the UAV is set to one-quarter of the truck's cost, a design choice aimed at balancing the cost-effectiveness between UAVs and trucks.The Gaussian Kernel Density Estimation (KDE) method is employed for the scientific delineation of UAV service centers. Each minimal UAV service center must satisfy the following condition: the circular area centered on the current customer point, with a radius equal to half the UAV's flight endurance, must cover at least 5 customer points. The UAV's flight endurance is set to 15 unit distances. Each reported result is the average of 10 independent runs.

TABLE IV
RESULT OF DIFFERENT ALGORITHMS IN LARGE-SCALE SCENARIOS

Algorithm	100 Customers	
	HV	Time
Random+ MONSNSA_RL_noGRU	0.42	1131.36s
Random+ MONSNSA_RL	0.53	1204.75s
KDE+ MONSNSA_RL_noGRU	0.69	1242.58s
KDE+ MONSNSA_RL	0.72	1283.69s
MONSNSA	0.74	1783.68s
	200 Customers	
	HV	Time
Random+ MONSNSA_RL_noGRU	0.37	1675.67s
Random+ MONSNSA_RL	0.44	1794.34s
KDE+ MONSNSA_RL_noGRU	0.59	1724.95s
KDE+ MONSNSA_RL	0.62	1879.18s
MONSNSA	0.66	4520.33s
	400 Customers	
	HV	Time
Random+ MONSNSA_RL_noGRU	0.31	3921.61s
Random+ MONSNSA_RL	0.39	4230.49s
KDE+ MONSNSA_RL_noGRU	0.50	4102.21s
KDE+ MONSNSA_RL	0.57	4631.27s
MONSNSA	0.43	-

Table IV shows the performance comparison of different algorithms in large-scale customer scenarios, primarily evaluated from the aspects of the HV metric and computation time. As can be seen from the table, the algorithms using the KDE method to delineate service centers, specifically KDE + MONSNSA_RL_noGRU and KDE + MONSNSA_RL, demonstrate higher HV values across all customer scales. This indicates their ability to maintain good performance in customer populations with clustering characteristics. Particularly, the KDE + MONSNSA_RL algorithm achieves the highest HV value of 0.57 in the 400-customer scenario, proving its superiority in large-scale settings.

In contrast, the algorithms based on randomly assigned center points, namely Random + MONSNSA_RL_noGRU and Random + MONSNSA_RL, perform poorly in terms of the HV metric. For example, the HV value of Random + MONSNSA_RL_noGRU is only 0.31 under the 400-customer scenario. This indicates its ineffectiveness in adequately covering the customer population when handling large-scale problems, resulting in the UAVs' inability to serve customers effectively.

It is worth noting that the MONSNSA_RL algorithm performs better in the HV metric compared to MONSNSA_RL_noGRU. This suggests that the GRU module helps enhance the sequence modeling capability and long-term dependency capture of the RL module, thereby optimizing the path planning strategy more effectively. Furthermore, the computation time of the MONSNSA algorithm is significantly higher than other algorithms, reaching 4520.33 seconds for the 200-customer scenario. This further indicates that the RL module can effectively assist MONSNSA in solving path planning problems for large-scale customer populations.

VI. CONCLUSION

This research addresses the truck-UAV collaborative delivery problem in large-scale urban scenarios by proposing a Multi-Objective Non-Dominated Sorting Simulated Annealing algorithm combined with Deep Reinforcement Learning (MONSNSA_RL). Confronting the logistics timeliness challenges posed by e-commerce development and the limitations of traditional delivery models in high-density urban environments, this study innovatively introduces a hybrid delivery mode. Customers are categorized into dense and sparse types, served by fixed trucks acting as UAV deployment centers and trucks carrying UAVs for delivery, respectively. This strategy effectively enhances UAV utilization and overall delivery efficiency.

To further optimize path planning in large-scale customer scenarios, this study developed the TransGRU model by integrating deep reinforcement learning with an attention mechanism. By combining a multi-head attention mechanism and Gated Recurrent Units (GRU), this model not only captures spatial correlation information between different UAV service centers but also handles temporal dependencies between subsequent actions, thereby significantly improving computational efficiency and generalization capability.

Experimental results validate the effectiveness of the proposed method. When handling large-scale customer points, the MONSNSA_RL algorithm effectively reduces delivery costs and improves customer satisfaction, demonstrating significant advantages particularly in high-density urban environments. Compared to traditional algorithms, MONSNSA_RL shows substantial improvements in both path planning quality and computational efficiency, providing an efficient solution for collaborative delivery problems in large-scale market scenarios. This study not only theoretically enriches the applications of multi-objective optimization and deep reinforcement learning but also offers new ideas and methods for practical logistics distribution. Future research could further explore incorporating more real-world factors (such as traffic congestion, weather conditions, etc.) into the model to enhance the algorithm's practicality and robustness.

REFERENCES

[1] A. Jazairy, E. Persson, M. Brho, R. von Haartman, and P. Hilletofth, "Drones in last-mile delivery: a systematic literature review from a logistics management perspective," The International Journal of Logistics Management, vol. 36, no. 7, pp. 1–62, Feb. 2024, doi: 10.1108/ijlm-04-2023-0149.

[2] S. Liu, W. Zhang, S. Yang, and J. Shi, "RETRACTED: A novel truck-drone collaborative service network for wide-range drone delivery using a modified variable neighborhood search algorithm," Journal of Intelligent & Fuzzy Systems, vol. 43, no. 4, pp. 5165–5184, Aug. 2022, doi: 10.3233/jifs-220378.

[3] L. Di Puglia Pugliese, G. Macrina, and F. Guerriero, "Trucks and drones cooperation in the last-mile delivery process," Networks, vol. 78, no. 4, pp. 371–399, Dec. 2020, doi: 10.1002/net.22015.

[4] L. Zhang, School of Economics and Management, Southeast University, Nanjing 211189, China, Y. Ding, and H. Lin, "Optimizing synchronized truck-drone delivery with priority in disaster relief," Journal of Industrial and Management Optimization, vol. 19, no. 7, pp. 5143–5162, 2023, doi: 10.3934/jimo.2022166.

[5] S. Tian, X. Wen, B. Wei, and G. Wu, "Cooperatively Routing a Truck and Multiple Drones for Target Surveillance," Sensors, vol. 22, no. 8, p. 2909, Apr. 2022, doi: 10.3390/s22082909.

[6] H. Li, F. Wang, and Z. Zhan, "Truck and rotary-wing drone routing problem considering flight-level selection," Journal of the Operational Research Society, vol. 75, no. 2, pp. 205–223, Mar. 2023, doi: 10.1080/01605682.2023.2185548.

[7] A. R. Kuroswiski, H. B. Pires, A. Passaro, L. N. Frutuoso, and E. L. F. Senne, "Hybrid Genetic Algorithm and Mixed Integer Linear Programming for Flying Sidekick TSP," 2023, arXiv. [Online]. Available: https://arxiv.org/abs/2304.13832

[8] O. Bräysy and M. Gendreau, "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms," Transportation Science, vol. 39, no. 1, pp. 104–118, Feb. 2005, doi: 10.1287/trsc.1030.0056.

[9] R. J. Kuo, S.-H. Lu, P.-Y. Lai, and S. T. W. Mara, "Vehicle routing problem with drones considering time windows," Expert Systems with Applications, vol. 191, p. 116264, Apr. 2022, doi: 10.1016/j.eswa.2021.116264.

[10] M. F. N. Maghfiroh, V. F. Yu, A. A. N. P. Redi, and B. N. Abdallah, "A Location Routing Problem with Time Windows Consideration: A Metaheuristics Approach," Applied Sciences, vol. 13, no. 2, p. 843, Jan. 2023, doi: 10.3390/app13020843.

[11] B. Peng, J. Wang, and Z. Zhang, "A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems," 2020, arXiv. [Online]. Available: https://arxiv.org/abs/2002.03282