An Optimization Algorithm for Order Delivery Based on Complex Networks

Tongji Yang*, Ruimin Li, Jun Wu

School of business administration, Liaoning Technical University, Huludao, Liaoning, 125105, China

Yhy2112@126.com* 1920528606@qq.com 2762978079@qq.com

Abstract. To address the issue of low order delivery efficiency, this study introduces complex network theory and constructs a complex network model for order delivery. In this model, regions requiring order delivery are represented as nodes, the presence of order delivery between two regions is indicated by edges, and the weights assigned to the edges represent the number of orders to be delivered. Furthermore, each dispatched vehicle is treated as an autonomous agent, and the order dispatch is formulated as a distributed decision problem aimed at maximizing rewards. To address this problem, we propose the delayed algorithm and the fast dispatch algorithm. The results of numerical experiments demonstrate that the proposed algorithms achieve superior quality solutions within a shorter time frame compared to other baseline algorithms, thereby confirming their effectiveness.

Keywords: order dispatch; distributed; agent; complex networks

1 INTRODUCTION

The continuous growth in the number of smartphone users has led to an increasing number of consumers opting for online shopping, resulting in a significant surge in online shopping orders. In 2023, as reported by the State Post Bureau, the national express delivery service industry handled a total of 132.07 billion parcels, indicating the considerable challenges posed by the scale of online shopping orders to the order delivery process. The majority of current order deliveries are assigned to one or more couriers based on regional divisions. However, a challenge arises when a courier is already engaged in delivering an order and a new order task emerges. Determining the appropriate trade-offs in such situations has become a problem that needs to be addressed.

To address the aforementioned issue of inefficient order delivery, we introduce complex network theory. Firstly, we model order delivery as a complex network based on the order volume in each region. In this network model, each region requiring order delivery is represented as a node, while the edges between nodes indicate the existence of an order delivery relationship between two regions. The number of orders dispatched between two regions is represented as the weights assigned to the

V. Vasilev et al. (eds.), *Proceedings of the 2024 5th International Conference on Management Science and Engineering Management (ICMSEM 2024)*, Advances in Economics, Business and Management Research 306,
https://doi.org/10.2991/978-94-6463-570-6_122

edges. Subsequently, each dispatched vehicle is treated as an independent entity, and the order dispatching process requires making distributed decisions based on the relevant attributes of each order. Lastly, we propose the delayed algorithm and the fast delivery algorithm to ensure timely order delivery while maximizing rewards.

Kandula[1] proposed a two-stage framework to enhance the delivery rate of orders and minimize delivery costs. Jiang[2] introduced a hybrid algorithm that combines the Variable Neighborhood Search heuristic and Adaptive Neighborhood Search heuristic to optimize both delivery and return services. Additionally, to minimize order delivery time, Chen[3] and Liu[4] developed three approximation algorithms and Variable Neighborhood Search (VNS) algorithms, respectively, to address this problem. Currently, reinforcement learning and deep learning techniques are widely employed to tackle the order dispatch problem. Zou[5] employed deep reinforcement learning to address the dynamic adjustment challenge of order scheduling and delivery routes in an online food ordering platform. Huang[6] integrated reinforcement learning and deep learning techniques to optimize the order management process in the supply chain. Additionally, Mar[7] presented a linear programming model that incorporates multiple picking locations within a shop. Huang[8] formulated the truck delivery problem as a mixed integer nonlinear programming problem and represented it as a directed graph.

2 PROBLEM DESCRIPTION AND MODEL BUILDING

2.1 Problem Description

Each vehicle responsible for order delivery is treated as an independent agent, and these agents deliver orders to different regions within the city. Connecting the regions traversed by the vehicles during the delivery process using solid lines forms a complex order delivery network. The order delivery network is denoted by $E = \{V, \varepsilon\}$, where $V = (1,2,\cdots,\mathrm{N})$ represents the set of nodes in the network, specifically corresponding to the regions associated with the order destinations. The edges of network is defined as $\varepsilon\left(\varepsilon \in V \times V\right)$. ε_{ij} is represented by a node with respect to $\left(V_i, V_j\right)$ in the network.

In light of the practical scenario, this study treats the order delivery network as a directed network. Therefore, a node pair signifies the dispatch of an order from node i (representing region i) to node j (representing region j).Define the weight adjacency matrix of E as

$$A = (a_{ij})_{N\times N} \begin{cases} a_{ij} > 0 & (i,j) \in \varepsilon \\ a_{ij} = 0 & (i,j) \notin \varepsilon \end{cases} \tag{1}$$

Since order delivery belongs to a directed network, the Laplacian matrix $L=(l_{ij})_{N\times N}$ for E is defined as: $L=D^{in}-A$, where $D^{in}=\mathrm{diag}(d_1^{in},d_2^{in},\cdots,d_N^{in})$, $d_i^{in}=\sum_{I=1}^{N}a_{ij}$ represent the degree of the agent i.

The actual order delivery problem can be approached as a set of multi-agent distributed optimization problems, as outlined below:

$$\max\sum_{i=1}^{N}Q(s_i) \tag{2}$$

$$\text{s.t. } s_i\in\cap_{i=1}^{N}\Omega_i \tag{3}$$

In equation (2) $s_i:R_n\to R$ is expressed as the agent's state value function i, $s_o\in R_n$ represents the decision variable shared by the agent, and the objective function is to find the maximum reward for the agent by sending the order. In equation (3), it refers to the locally closed convex set constraint of the agent.

2.2 Definition of Problem-related Elements

In this paper, the order dispatch problem is modelled as a Markov Decision Process (MDP) with multiple intelligences. Subsequently, we introduce the essential element $K=\langle R,N,S,A,W\rangle$ of the Markov decision process formulation for multi-agent systems. Here, N represents the number of delivery vehicles, S denotes the set of states, A represents the joint action space, and W corresponds to the reward function.

Definition 1 (Order R). Consider the dispatch of an order as a collection of orders consisting of r orders. $r=\langle l_r,e_r,u_r\rangle$, l_r,e_r,u_r represent, respectively, the destination of order r, the delivery deadline, and the reward (utility) received for completing the order.

Definition 2 (Delivery Vehicle). The delivery vehicle can be represented by a three-element vector $n=\langle s_n,d_n,c_n\rangle$, where s_n denotes the initial location, d_n represents the destination to be reached, and c_n indicates the current location of the vehicle. The reward (utility) can only be obtained if the delivery driver successfully completes the order within the specified time window.

Definition 3 (Status S_t). Consider a three-element tuple denoted as S_t, representing the state $S=\langle t,l,1\rangle$. Each of these elements signifies the following: time, the initial position of the vehicle, and the driver responsible for delivering the order.

Definition 4 (Action a_t). This definition signifies the allocation of a particular trip to the driver. It is set to 0when the vehicle is idle and 1 otherwise. $S_0=\langle t_0,l_0,0\rangle$ represents the driver's time t_0, position l_0, and vehicle availability 0 (no delivery of any order has been performed) upon trip assignment. $S_1=\langle t_1,l_1,1\rangle$ represents the time,

location of the order, and the driver delivering the order. The set of all eligible action spaces is denoted by A .

Definition 5 (Reward W). When an order delivery driver successfully delivers an order to the customer's specified address before the deadline, they are rewarded accordingly. Moreover, as the number of orders delivered by the driver increases, the reward also increases. Building upon this premise, a reward function is designed to represent the cumulative rewards per unit time step, as expressed by the following equation:

$$W_t = w_{t+1} + dw_{t+2} + \cdots + d^{k_{t-1}} w_{t+k_t} \tag{4}$$

Definition 6 (Reward). When an order delivery driver successfully delivers an order to the customer's specified address before the deadline, they are rewarded accordingly. Moreover, as the number of orders delivered by the driver increases, the reward also increases. Building upon this premise, a reward function is designed to represent the cumulative rewards per unit time step, as expressed by the following equation:

$$F(s,a) = E\left[\sum_{t=0}^{T} d^t R\left(S_t, A_t\right) \middle| S_0 = 0, A_0 = 0\right] \tag{5}$$

Definition 7 (Strategy $\pi = \langle a|s \rangle$). Strategy $\pi = \langle a|s \rangle$ denotes a distribution strategy that maps states to either an action space or a specific action. It can be expressed in the following manner:

$$\pi(s) = \arg\max V(s,a) \tag{6}$$

Definition 8 (State Value Function $Q(s)$). By adhering to the strategy from the outset, the driver can accumulate the expected cumulative reward until the conclusion of the time step. The state value function is defined as follows:

$$Q(s) = V\left(s, \pi(s)\right) = \max_{a \in A} V(s,a) \tag{7}$$

To achieve the optimal order delivery solution, it is necessary to facilitate distributed collaboration among intelligent agents (delivery vehicles). The subsequent subsections provide a comprehensive description of the visualization network designed for order delivery.

2.3 Visual Network for Orders

Due to the customer-centric nature of online order placement, it is impractical to establish a network solely based on pairwise order relationships. Hence, this paper constructs the order delivery network using inter-city order data, which is illustrated in Figure 1 as the order delivery visualization network. The network incorporates the mobile characteristics of delivery vehicles, treating each region between cities where orders are to be delivered as a network node (represented by letters in Figure 1). This

approach enables clear differentiation between various regions. In the presence of a connection between region i and region j, there exists a flow of orders between the two regions. This connection is referred to as an edge in the network, denoted as (i, j). Furthermore, if there is a requirement to dispatch 30 orders from region i to region j, then the weight assigned to this connection in the network is also 30. In other words, the weight in the network corresponds to the number of orders and is visually represented as a numerical value on the line in Figure 1.

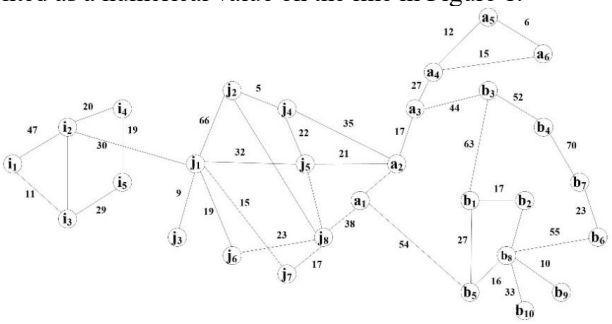


Fig. 1. Visual network for order delivery

3 ALGORITHM DESIGN BASED ON ORDER DISPATCH OPTIMIZATION

Typically, the complete order delivery process comprises three stages: order reception, order delivery, and order completion, as illustrated in Figure 2. Initially, the online platform receives the order, followed by the integration of order information and vehicle data at the distribution center (site). This integration facilitates the optimal allocation of vehicles and drivers to execute the assigned order delivery tasks. Subsequently, the driver ensures the timely delivery of the order to the customer's specified address, thereby concluding the entire order delivery process. The judicious scheduling of orders to delivery vehicles represents a critical decision-making endeavor for distribution centers (stations).

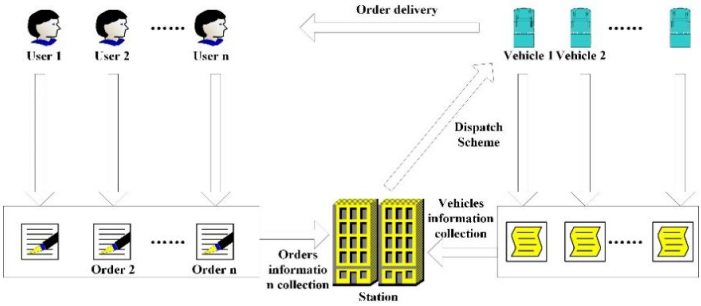


Fig. 2. General framework for order dispatch

3.1 Optimization Strategies for the Problem

This paper incorporates elements of complexity theory and utilizes a grid graph as a visual representation of an actual order dispatch network, as depicted in Figure 3. The nodes in the grid graph correspond to distinct regions within the city, while the edges signify the presence of inter-order delivery connections between these regions. Each edge is annotated with a numerical value representing the quantity of orders to be delivered between the respective regions, thus serving as weights in the network. Both vehicles and orders are randomly positioned within the grid graph.

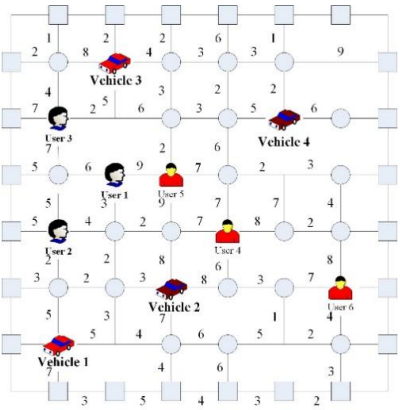


Fig. 3. Roadmap for order dispatch

Delivery vehicle drivers are incentivized through rewards for timely and accurate delivery of orders to customers at the designated addresses. To maximize their rewards, it is crucial for drivers to deliver as many orders as possible within the specified time window. To enhance the efficiency of order delivery, this paper proposes the delayed delivery algorithm.

3.2 Delayed Delivery Algorithm

The delayed delivery algorithm involves temporarily assigning one or more orders to a driver at a distribution center (referred to as a "site") while they are already engaged in an ongoing delivery. However, this assignment is subject to the time window of the ongoing delivery. To accommodate these temporary assignments, an order pool is created at the distribution center to store the dispatched orders. Once a delivery vehicle finish delivering its current order, it is assigned a new order delivery from the order pool using the Delayed Delivery algorithm.

Algorithm1 provides a comprehensive description of the precise steps involved in the deferred dispatch algorithm in Table 1. Delayed Delivery algorithm, outlined below:

Table 1. Delayed Delivery algorithm

Algorithm 1 Delay−Delivery

Input: A Set of vehicles N , a set of orders R

Output: Delivery for $n \in N$

1 orderPool $\leftarrow \varnothing$;
2 freeVehicleSet $\leftarrow \varnothing$;
3 Set Sn an empty order for each $n \in N$
4 **for** each new arrival request **do**
5 if the request is an order r **then**
6 if there exists a vehicle $n' \in$ freeVehicleSet can accomplish r with largest BEN **then**
7 Append r to $S_{n'}$
8 freeVehicleSet $\leftarrow$ freeVehicleSet$-\{n'\}$
9 **else**
10 orderPool$\leftarrow$orderPool $\cup \{r\}$;
11 **else**
12 // Denote the arrival vehicle by n
13 BenefitGreey(n ,orderPool);
14 if there is no order appended to S_n **then**
15 freeVehicleSet $\leftarrow$ freeVehicleSet $\cup \{n\}$

In lines 1-2, the order pool and the set of free vehicles are initialized. Each delivery vehicle, denoted as $n \in N$, is associated with an order sequence S_n in line 3. In line 5, a decision is made to determine whether the demand corresponds to an order. When an order arises in lines 6-8, it is prioritized for assignment to the vehicles in the set of free vehicles. If scheduling an order fails in lines 9-10, the order is temporarily placed in the order pool. When the driver of a delivery vehicle completes their current order delivery, the order previously placed in the order pool is scheduled for that driver on line 13. A delivery vehicle is classified as a "new vehicle" in the delayed delivery algorithm only when it successfully completes its current order delivery, as indicated in lines 12-15.

3.3 Fast Delivery Algorithm

To address the potential issue of the delayed delivery algorithm being unable to complete the order within the specified time frame, leading to a relatively small reward, we propose the fast delivery algorithm. The fast delivery algorithm promptly assigns an order to a delivery vehicle regardless of its time of arrival. To maximize the total reward (utility), the algorithm straightforwardly integrates the new order with the delivery vehicle's ongoing task. Likewise, Algorithm 2 provides a detailed description of the steps involved in the fast delivery algorithm in Table 2. Fast delivery algorithm, outlined below

Table 2. Fast delivery algorithm

Algorithm 2 Fast–Delivery

Input: A set of Vehicles N, a set of orders R

Output: Plans for $n \in N$

```
1  aVehicleSet ← ∅, freeOrderSet ← ∅;
2  for each new arrival request do
3    if the request is a vehicle n then
4      BenefitGreey( n , freeOrderSet);
5      aVehicleSet←aVehivleSet ∪ {n}
6    else
7      // Denote the arrival order by r
8      n_best ← None, bestComPos ← –1, minCost ← ∞
9      for each combination position ComPos in S_{n_a} do
10       tmpCost ← extra distance if combine r to ComPos
11       if tmpCost < minCost then
12         minCost ← tmpCost, bestComPos ← ComPos, n_best ← n_a
13     if minCost < ∞ then
14       Combine r with S_{n_best} according to bestComPos
15   else
16       freeOrderSet ← freeOrderSet ∪ {r}
```

Line 1 initializes the set of available vehicles and the set of unscheduled orders, also known as the free order set. Line 3 determines whether the new demand corresponds to vehicle n. When a vehicle arrives in lines 4-5, it is selected from the free order set, and an order delivery is scheduled for that vehicle n using an order from the free order set. In lines 8-15, if order r becomes available, it is combined with the existing order sequence from the set of available vehicles to minimize travel distance. If no such combination exists in lines 16-17, order r is added to the free order set.

4 EXPERIMENTAL EVALUATION

4.1 Experimental Setup

To assess the effectiveness of the proposed algorithm, this paper adopts an experimental design from the literature [9] to evaluate its performance using both synthetic and real data. Table 3. Relevant parameter settings for the synthetic data. presents the relevant parameter settings for the synthetic data. In this case, the orders and delivery vehicles are randomly selected from a 600×600 metric space, each having distinct R and V values.

Table 3. Relevant parameter settings for the synthetic data.

R	3,6,9,12,15
V	2,4,6,8,10

Besides the two algorithms proposed in this paper, three additional baseline algorithms were selected for comparative analysis. To facilitate experimental design, the Delayed-Delivery Algorithm and Fast-Delivery Algorithm are respectively referred to as Delayed and Fast.

Random: At each corresponding time step, the algorithm randomly assigns idle orders to available vehicles for delivery.

Greedy: The algorithm employs a first-come-first-served delivery strategy, prioritizing orders based on their placement time. Consequently, orders placed earlier are given higher dispatch priority.

QMIX: The algorithm, proposed in the literature[10], considers the global state and utilizes decentralized execution to decompose the joint action value network function.

MATLAB2022a was used to run all the algorithms, while the actual operations of different experiment components were executed on a machine equipped with an 11thGen Intel(R) Core(TM) i7-1165G7@2.80GHz processor and 512GB storage.

4.2 Experimental Results

Algorithm Validity Test. Line plots in Figures 4a-4c depict the rewards, algorithm run duration, and algorithm memory consumption (MB) as a function of order R . The results are presented in Figures 4a-4c. It is evident from the results that both the delayed delivery algorithm and the fast delivery algorithm achieve higher order delivery rewards compared to the three baseline algorithms for the same order quantity. Furthermore, as the number of orders increases, the rewards earned for the orders continue to grow, surpassing the performance of the three baseline algorithms consistently.

Regarding the algorithm's running duration, the delayed delivery algorithm proposed in this paper exhibits the shortest execution time for a small number of orders. However, as the number of orders increases, the fast delivery algorithm becomes the most efficient in terms of time consumption.

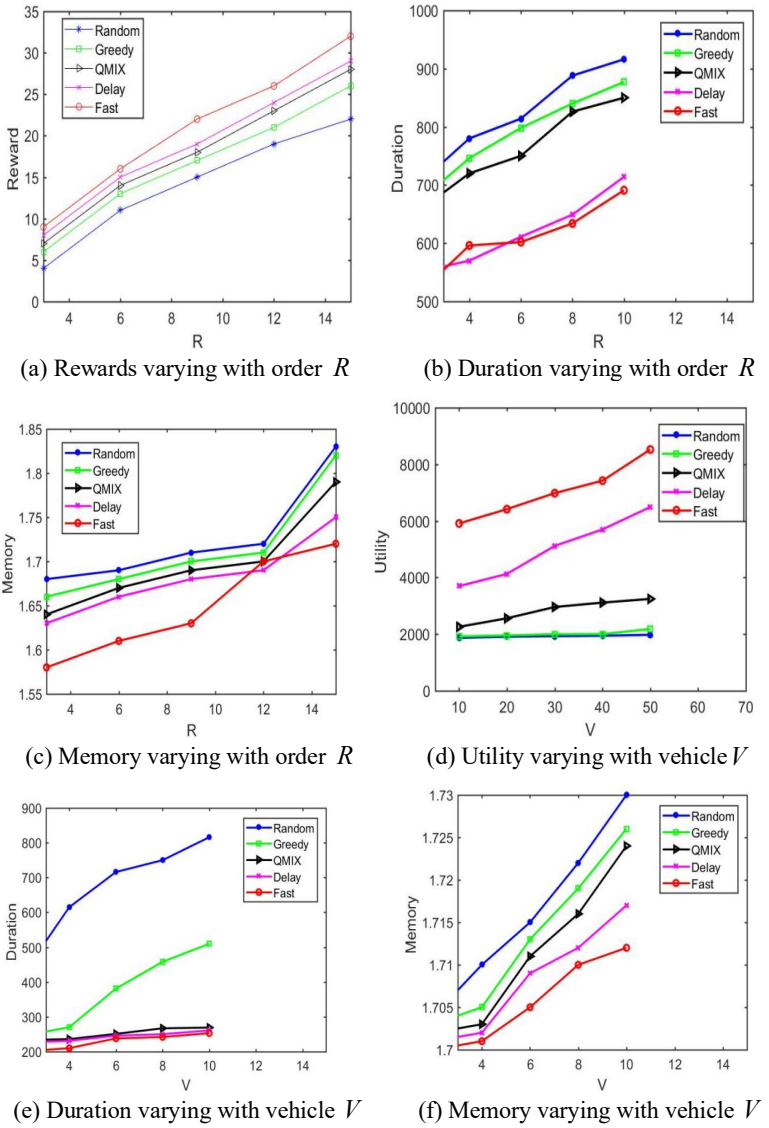
Regarding the memory (MB) consumption of the algorithms, it is apparent that the memory usage increases as the number of orders grows. However, the two algorithms proposed in this paper consume a lower amount of memory compared to the top-performing baseline algorithm.

Figures 4d-4f present line plots depicting the utility, algorithm run duration, and algorithm memory consumption (MB) as a function of vehicle V . The results clearly indicate that the proposed algorithm outperforms the three baseline algorithms significantly.

Regarding vehicle utility, the fast dispatch algorithm yields significantly higher utility compared to the three baseline algorithms, while the utility obtained by the three baseline algorithms remains relatively stable. In terms of algorithm operation duration, the delayed delivery algorithm and fast delivery algorithm proposed in this

paper exhibit improvements over the three baseline algorithms, resulting in time savings.

Regarding algorithmic memory (MB), the memory usage increases as the number of vehicles varies for all algorithms. However, the two algorithms proposed in this paper consume slightly less memory compared to the three baseline algorithms.



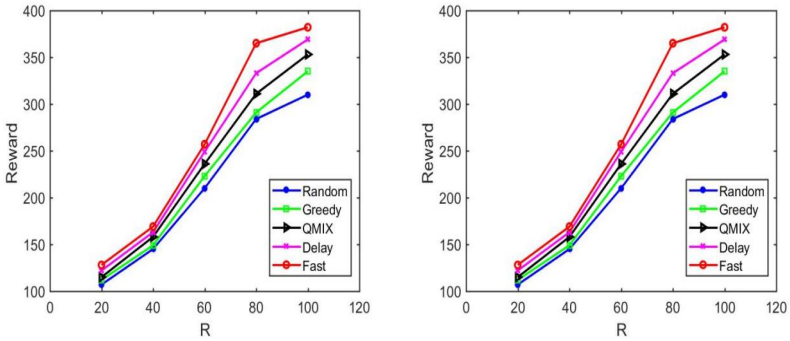
(a) Rewards varying with order R (b) Duration varying with order R

(c) Memory varying with order R (d) Utility varying with vehicle V

(e) Duration varying with vehicle V (f) Memory varying with vehicle V

Fig. 4. (a-c) Line plots of reward, algorithm run duration, and algorithm memory consumption (MB) varying with order R , (d-f) Line plots of reward, algorithm run duration, and algorithm memory consumption (MB) varying with vehicle V .

Experiments based on Real Datasets. Figures 5a-5b display the results of the five algorithms applied to the real dataset, depicting order rewards and vehicle utility. As the number of orders increases, the rewards also increase. Notably, the rewards obtained and the increasing trend observed for the two methods proposed in this paper surpass those of the three baseline algorithms. Likewise, the utility of the vehicles exhibits a linear increase as the number of delivered vehicles rises, with the fast delivery algorithm proposed in this paper achieving the highest utility.



(a)Reward on the real change dataset R (b) Utility on the real change data set

Fig. 5. Real Dataset Results: (a) Rewards on the True-Changing Dataset; (b) utility on real change data set.

5 CONCLUSIONS

This paper addresses the issue of low delivery efficiency in current order delivery systems. To enhance order delivery efficiency, the complex net-work theory is introduced, transforming the order delivery process into a complex network through networked design. To optimize order delivery, this study combines distributed optimization theory and treats each delivery vehicle as an independent intelligent entity. The order delivery problem is formulated as a distributed decision-making problem. Two proposed algorithms, namely the delayed delivery algorithm and fast delivery algorithm, aim to maximize rewards while ensuring timely delivery of orders. Experimental results using synthetic values and real data demonstrate the effectiveness of the proposed algorithms and methods.

REFERENCE

1. KANDULA, Shanthan; KRISHNAMOORTHY, Srikumar; ROY, Debjit. A prescriptive analytics framework for efficient E-commerce order delivery. Decision Support Systems, 2021,147: 113584. DOI:10.1016/j.dss.2021.113584
2. JIANG, Dapei, et al. Integrating order delivery and return operations for order fulfillment in an online retail environment. Computers & Operations Research, 2022, 143: 105749.
3. CHEN, Ren-Xia; LI, Shi-Sheng. Minimizing maximum delivery completion time for order scheduling with rejection. Journal of Combinatorial Optimization, 2020, 40.4: 1044-1064.

4. LIU, Ling, et al. A coordinated production and transportation scheduling problem with minimum sum of order delivery times. Journal of Heuristics, 2020, 26: 33-58.
5. ZOU, Guangyu, et al. Online food ordering delivery strategies based on deep reinforcement learning. Applied Intelligence, 2022, 1-13.
6. TAN, Xin. Application of reinforcement learning algorithm in delivery order system under supply chain environment. Mobile Information Systems, 2021, 2021: 1-11.
7. Vazquez-Noguerol, M., Comesaña-Benavides, J., Poler, R., & Prado-Prado, J. C. (2022). An optimisation approach for the e-grocery order picking and delivery problem. Central European Journal of Operations Research, 30(3), 961-990.
8. Huang, M., Zhang, H., Kuang, H., Yu, Y., Lee, L. H., & Wang, X. (2019). Flexible truckload pickup and delivery problem considering reserved orders and fuel consumption. Computers & Industrial Engineering, 138, 106117.
9. Huang Xiaohui, Zhang Xiong, Yang Kaiming, et al. Car order delivery in reinforcement learning network based on joint Q-value decomposition [J]. Computer Engineering, 2022,48 (12): 296-303+311.
10. Chen, Q. (2021). NQMIX: Non-monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning. arxiv preprint arxiv:2104.01939.