Crowd Sourcing Dynamic Pickup & Delivery Problem considering Task Buffering and Drivers' Rejection -Application of Multi-agent Reinforcement Learning-

JUNYI MO, SHUNICHI OHMORI
Department of Business Design and Management
Waseda Universty
3-4-1, Ohkubo, Shinjuku-ku, Tokyo, 169-0072
JAPAN

Abstract: - In the last decade, dynamic and pickup delivery problem with crowd sourcing has been focused on as a means of securing employment opportunities in the field of last mile delivery. However, only a few studies consider both the driver's refusal right and the buffering strategy. This paper aims at improving the performance involving both of the above. We propose a driver-task matching algorithm that complies with the delivery time constraints using multi-agent reinforcement learning. Numerical experiments on the model show that the proposed MARL method could be more effective than the FIFO and the RANK allocation methods.

Keywords: - Last Mile delivery, Crowd Souring, Dynamic Pickup & Delivery Problem, Multi-agent Reinforcement Learning, Task Buffering, Drivers' Rejection

Received: March 1, 2021. Revised: March 26, 2021. Accepted: March 30, 2021. Published: April 2, 2021.

1 Introduction

1.1 Background

Recently, the EC market has been growing year by year, and the importance of Last Mile Delivery (LMD) has been increasing accordingly[1]. On the other hand, it was predicted that the aging rate in Japan will increase every year and reach 39.9% in 2060. The logistics industry will face a serious labor shortage crisis in the future [2]. Therefore, improving the efficiency of LMD will be one of the most important social issues.

LMD can be broadly classified into two types: Delivery problem and Pickup & Delivery problem. The characteristics of each are described in Table 1. These delivery efficiencies have been studied academically as Vehicle Routing Problem (VRP) and Dynamic Pickup & Delivery Problem (DPDP) respectively. In this article, we focus on the Pickup & Delivery Problem, which has drawn attention in the food delivery industry. In the case of Pickup & Delivery Problem, the demand is dynamic and the Time Windows (TW) is short.

Table 1 Classification of LMD

Type	Delivery Problem	Pickup & Delivery Problem
Instance	E-tailing	Food Delivery
Companies for example	Amazon, Rakuten	Uber Eats, DiDi Food
Demand	static	dynamic
TW	long	short
Research Field	VRP	DPDP

In order to solve the problem of the labor crisis of LMD in logistic, Crowd Sourcing (CS) LMD has been is considered as a promising approach. CS is a process of outsourcing work to an unspecified number of people. In this article, CS type DPDP is specifically defined as CSDPDP.

Food delivery operations are described in Figure 1 below. Customers access and search dishes on an operator's (server's) application, such as Uber Eats, and pay for orders. The operator then sends a cooking request to the restaurant. Once the restaurant accepts the order, the operator asks the driver to deliver. After the delivery is completed, final payment is sent the operator and the operator pays the restaurant and the driver for the food and the delivery. It should be paid attention that the delivery fee paid to the driver is calculated on the basis of the delivery distance but not on the pickup distance. Therefore, assigning a task with a long pickup distance to a driver may be rejected.

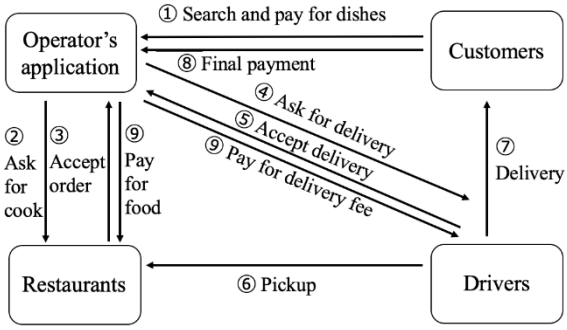

Fig. 1 Food delivery operations

Unlike normal employment, CS allows for shorter periods of engagement, making it easier to secure more labors. However, there are two difficulties with CSDPDP: firstly, unlike normal employment, the number of drivers and their working hours are unknown in advance in the server and are therefore probabilistic. This means that the decisions made at any time may not be optimal. The second difficulty is that drivers have the right to refuse the tasks allocated to them. So, it is necessary to allocate the tasks in such a way that the driver will not reject the task.

There are few previous related works which deal with the above two difficulties. The mentioned waiting for task assignment is known as Buffering Strategy in the field of DPDP, but it is mainly used to verify the effect on the uncertainty of tasks' arrival. The effect on the uncertainty of driver's arrival, which is a characteristic of CS Problem, has not been clarified yet. Matsueda has hypothesized three types of decision criteria for rejection for CSDPDP [3]. That work proposed a task allocation method called RANK, which assigns the task that is least likely to be rejected in order based on the type for each driver. Though, it is difficult to apply this method because it is hard to know the driver's type in advance in practice. In addition, Buffering Strategy is not considered in this paper.

1.2 Purpose

The main objective of this paper is to develop a method for CSDPPD that **takes (1) the Buffering Strategy and (2) the driver's right of rejection into account, which has not been sufficiently considered in previous works.**

We propose a task allocation algorithm built on multi-agent reinforcement learning. It allows for autonomous decentralized decision-making for each agent and thus can cope with situations when the arrivals of tasks and drivers are uncertain. Moreover, by learning how the current actions will affect the future ones, appropriate actions such as waiting or rejecting task allocation could be executed, depending on the state of the system.

2 Related Research

2.1 Dynamic Pickup & Delivery Problem (DPDP)

Dynamic Pickup & Delivery Problem (DPDP) is one of the most popular studies in the field of Pickup & Delivery Problem of LMD..

In recent years, due to the development of the computational resource, the solution of the DPDP can be applied in practice. Basic solution strategies, algorithmic performance assessment and prediction of future task information have been studied as solutions to the DPDP. Further study are worthy such as the optimal Waiting Strategy and the improvement of the objective function using the Rolling-horizon algorithm. The fields of dynamic many-to-many and one-to-many-to-one PDPs have also been highlighted as potential research areas.

Berbeglia et al. surveyed the topic of DPDP and summarized it in the paper [4]. Sayarshad et al. proposed to incorporate non-myopic pricing into the non-myopic dynamic dial a ride problem [5]. Muñoz-Carpintero et al. proposed a solution scheme which is designed to support the dispatcher of a dial-a-ride service [6]. The scheme considers different configurations of particle swarm optimization and genetic algorithms within a proposed ad-hoc methodology to solve in real time the nonlinear mixed-integer optimization problem related with the hybrid predictive control approach. Hyytiä, Penttinen and Sulonen defined a new problem Flexible Two-sided Online task Assignment to guide idle workers based on the prediction of tasks and workers so as to increase the total number of assigned worker-task pairs [12].

2.2 Crowd Sourcing Dynamic Pickup & Delivery Problem (CSDPDP)

In the study of CSDPDP, Arslan et al. proposed the Rolling Horizon method, which is a framework for solving route planning problems with dynamic on-demand deliveries using both occasional and fixed drivers [7]. As represented in Figure 2, the proposed method solves the task and driver matching problem by converting the online problem to an offline problem.

In this paper, an accurate solution method based on the matching formulation is developed to solve various versions of the offline problem repeatedly. It also addresses the lack of control over the supply of drivers by considering the use of normal dedicated vehicles. The proposed technique is a hybrid platform that allows the use of conventional dedicated vehicles to carry out specific tasks.

However, this study does not consider the refusal of the driver and the Buffering Strategy.

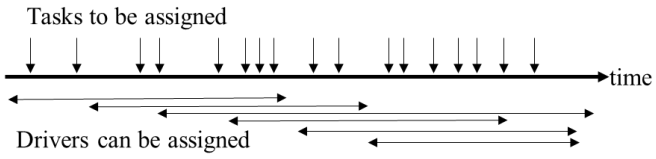

Fig. 2 Rolling Horizon

2.3 CSDPDP with drivers' refusal right

Matsueda studied LMD with drivers' refusal right [3]. When drivers have the refusal right, the solution would

be more complicated because the number of arrival of events increases. She defined 0-1 constant to indicate whether a driver accepts a task or not.

In order to reduce the rate of driver rejection, she proposed to divide the types of drivers who have the right to reject into three categories. The specifics are as follows:

Type 1 The higher reward the better
Type 2 Reward ≥ F: Only do those tasks for which the reward is relatively high (F = average reward of all tasks)
Type 3 Pre-processing time ≤ R: Unwilling to do much that is unrewarding (R = average of all pre-processing time)

In this study, she proposed a model to rank priorities to tasks based on drivers' preferences and compared the RANK algorithm and FIFO algorithm. However, since we could not know which type the driver belongs to before assigning the task, the type of the driver is random and there is not a uniform criterion to rank the task to the drivers, which causes problems to assign in reality. In addition, both RANK and FIFO do not take Buffering Strategy into account.

2.4 Dynamic delivery considering future rewards

Ulmer et al. integrated dynamic requests into delivery routes by taking advantage of preemptive depot returns [8]. And their approximation procedure captures both the current value of a subset selection decision and its impact on future rewards. Voccia et al. introduced a multi-vehicle dynamic pickup and delivery problem with time constraints that incorporates key features associated with same-day delivery logistics [9]. Their solution approach incorporates information about future requests into routing decisions. Klapp et al. introduced the same-day delivery problem that represents a current trend in business practice [10]. They presented a formal Markov decision process model and introduce an analytical result that identifies when it is beneficial for vehicles to wait at the depot. Tong, Wang and Zimu et al. provided a solid motivation for a class of non-myopic policies that take into account the unknown future requests [13]. They mentioned that customers (or items in a parcel service) are flexible in the sense that they tolerate (small) delays in their pick-up and delivery times.

2.5 Contribution in our study

Table 2 shows a comparison between previous studies and the present study. The DPDP investigated by Berbeglia et al. makes the number of drivers and their working hours deterministic, which differs from the others[4]. Ulmer et al. [8], Voccia et al [9]. and Klapp et al. [10] have taken account of **future rewards and buffering strategy** while others are not. And most related works are not applied to **refusal right of drivers** except for Matsueda's paper [3]. **So the contribution in our study is mainly considering the both.**

Table 2 Comparison between related research and this article

Problem	Research	Number and Worktime of Drivers	Buffering Strategy	Refusal Right of Drivers
DPDP	Berbeglia et al.[4]	definite		
	Sayarshad et al.[5]	stochastic		
	Muñoz-Carpintero et al.[6]	stochastic		
	Hyytiä, Penttinen and Sulonen [12]	stochastic		
SDDP(same-day delivery problem)	Ulmer et al[8]	stochastic	√	
	Voccia et al.[9]	stochastic	√	
	Klapp et al.[3]	stochastic	√	
CSDPDP	Arslan et al. [7]	stochastic		
	Matsueda [3]	stochastic		√
	Tong, Wang and Zimu et al. [13]	stochastic		√
	this paper	**stochastic**	√	√

3 Model

3.1 Assumptions

- The driver delivers only one item at a time.
- The speed of the driver is constant.

・The driver's assessment is not taken into account when assigning tasks.

・Routing is not taken into account. However, routing can be taken into account by solving the Traveling Salesman Problem (TSP) as a subroutine for the tasks assigned to each driver in this study.

・The waiting time at the pickup point is included in the pickup time.

・Assume a situation where as many tasks as possible are assigned to drivers. Tasks that are not allocated to any driver are handled by special means at an additional cost.

3.2 Input

$T = \{i = 1, \cdots, n\}$: set of tasks
$D = \{k = 1, \cdots, m\}$: set of drivers
$[e_k, l_k]$: available working time of driver k
$[P_i, D_i]$: pickup location and delivery location of task i
o_k : Initial location of driver k
t_{ij}^p : pickup time for task I followed by task i
t_i^d : delivery time for delivery of task i
$[e_i^d, l_i^d]$: Time Window (TW) early/late arrival time for task i
q_{ij} : binary constant indicating the rejection of task j when it is assigned after task i
$M1, M2, M3$: sufficiently large constants

3.3 Coefficient of determination

x_{ijk} : A binary variable that is 1 if driver k delivers task j after task i, and 0 otherwise
a_i : Delivery start time for task i

3.4 Formulations

Referring to Matsueda's paper[3], we define the offline task allocation problem as in (1).

$$\text{Max.} \sum_{j=1}^{n} \sum_{i=0}^{n} \sum_{k=1}^{m} (1 - q_{ijk}) x_{ijk} \quad (1a)$$

$$\text{s.t.} \quad \sum_{k \in D} \sum_{i \in T_j} x_{ijk} \leq 1 \quad j \in T \quad (1b)$$

$$\sum_{j \in T} x_{0jk} \leq 1 \quad k \in D \quad (1c)$$

$$\sum_{i \in T_j} x_{ijk} - \sum_{i \in T_j} x_{ijk} = 0 \quad k \in D, j \in T \quad (1d)$$

$$a_i + t_i^d + t_{ijk}^p - a_j \leq M_1(1 - x_{ijk}) \; k \in D, i \neq j \quad (1d)$$

$$a_j - t_k^d - t_{0jk}^p \geq M_2(x_{0jk} - 1) \; k \in D, j \in T \quad (1f)$$

$$a_j - l_k^d + t_k^d \leq M_3(1 - x_{i0k}) \; k \in D, j \in T \quad (1g)$$

$$a_j \geq \sum_{k \in D} \sum_{i \in T_j} (e_i^d - t_i^d) x_{ijk} \quad j \in T \quad (1h)$$

$$a_j \leq \sum_{k \in D} \sum_{i \in T_j} (l_i^d - t_i^d) x_{ijk} \quad j \in T \quad (1i)$$

$$x_{ijk} \in \{0,1\} \quad k \in D, j \in T, i \in T_j \quad (1j)$$

$$a_j \in \mathbb{R}_+ \quad j \in T \quad (1k)$$

The objective function (1a) is the maximization of the number of allocated tasks. Equation (1b) is the constraint that each task is delivered at most once. Equation (1c) is the constraint that each driver delivers at most one task a time. Equation (1d) is a flow constraint. Equation (1e) is the compatibility of successive tasks. Equations (1f) and (1g) are the available working time constraints for each driver. Equations (1h) and (1i) are the TW constraints for each task. Equation (1j) is the binary condition of x_{ijk}. Equation (1k) is a restriction on the start time of delivery for each task.

Problem (1) can be solved optimally as an off-line problem if the input information about tasks and drivers is known in advance. However, as this information is changing from time to time, it needs to be solved as an online problem.

4 Math Method

4.1 Application of multi-agent reinforcement learning

Multi-agent reinforcement learning is the learning of optimal policies by means of trial-and-error interactions with the environment by multiple autonomous agents. It is suitable for solving online problems in dynamic situations, because it can learn the policy of what actions to take in uncertain environments and under what conditions.

There are two types of agents in our study: task agents and driver agents. Define T_t as the set of uncompleted tasks and D_t as the set of waiting drivers at each time t of the defined event set E. E is defined by the arrival time c_i of each task i, the start work time e_k of each driver k, the time when each task i must be assigned to driver k at the latest time $l_i^d - t_i^d - t_{j_k i}^p$, which are three types of time sets. However, we assume that j_k is the task that driver k has completed immediately before. As shown in Figure 3(a), each task agent observes the state s_{it} in the environment and decides whether to request or wait for the driver agent according to the policy $\pi_T(s)$. As shown in Figure 3(b), the driver agent receives the request and decides whether to accept or reject it according to the policy $\pi_D(s)$. The algorithm of the proposed technique will be as follows.

Step 1. Task agent observes state s_{it}
Step 2. Task agent determines action a_{it} according to policy $\pi_T(s)$
Step 3. Driver agent observes state s_{kt}
Step 4. Driver agent determines action a_{it} according to policy $\pi_D(s)$
Step 5. The task agent obtains the reward r_{it}
Step 6. Each agent's action results in a state

transition: $s_{it} \rightarrow s_{i(t+1)}$, $s_{kt} \rightarrow s_{k(t+1)}$

Step 7. Time update: $t \rightarrow t+1$

Step 8. If all events are finished, go to Step 9, otherwise go to Step1.

Step 9. The task agent updates the value $Q(s,a)$

Step 10. If t=1, go to Step 1.

Each agent is assumed to be unaware of the internal states and strategies of other agents. As the system cannot direct the drivers, driver agents do not learn this time. We also assume that each driver is proficient so the speed and the capacity of each driver is the same. On this basis, we model a situation in which the task agent learns a policy in different environments. In order to discretize the state and reward of each agent, we classify each parameter into seven classes which are shown in Table 3.

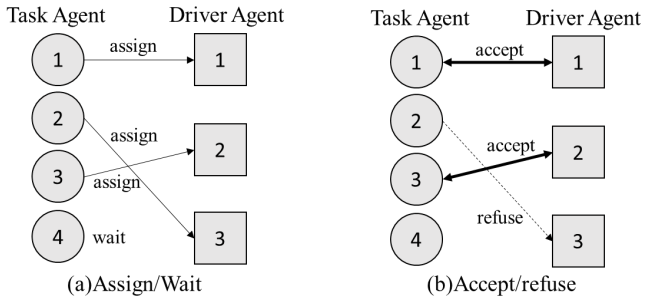

Fig. 3 Examples of agents' determination process

Table 3 The classification of arameters

Class	Range	Class	Range
0	0<	4	[0.6,0.8]
1	[0.0,0.2]	5	[0.8,1.0]
2	[0.2,0.4]	6	1>
3	[0.4,0.6]		

4.2 Task agent's condition, action, reward and value update

The task agent chooses its actions based on the state $S = S^I \times S^E$, which consists of the combination of the agent's internal state S^I and the environmental state S^E. We define the internal state as the class $S^I = \{1, \cdots, 5\}$ of the average urgency of delivery. When task i is assigned to driver k, the distance d_{ik} and the urgency of delivery u_{ik} are defined by the following equations (2) and (3).

$$d_{ik} = t + t^p_{jki} + t^d_i \tag{2}$$

$$u_{ik} = \frac{d_{ik} - a_i}{l^d_i - a_i} \tag{3}$$

The environmental state is defined as the class of driver shortage degree $S^E = \{1, \cdots, 6\}$. The driver shortage degree s^E_i is defined by the following equation (4). $|D_t|$ is the number of active drivers who can be assigned at time t. And $|T_t|$ is the number of incomplete tasks.

$$s^E_t = \frac{|D_t|}{|T_t|} \tag{4}$$

The actions of the task agent are assignment and waiting for a driver. When assigning a task to a driver, if we assume that all drivers can be chosen, the number of actions will be too large to learn. Therefore, we set actions $a_t \in \{0,1,\cdots,5\}$, where $a_t = 0$ means waiting, and $a_t = 1,\cdots,5$ means choosing a driver of reward in class 1, ⋯,5 respectively, as described above. If there is more than one driver of different reward classes, the driver with the highest reward would be chosen.

The reward of the task agent is represented in Equation (5) and Figure 4. Arriving at the early arrival time of task's TW is considered to be the most efficient, and the reward is set to a maximum of 1 at e^d_i. Also, we set the reward to be higher the closer to e^d_i in $[c_i, l^d_i]$. And the absolute value of the gradient of the reward function at $[c_i, e^d_i]$ is set to be lower than $[e^d_i, l^d_i]$ as we believe that to arrive during TW is better than to arrive before the early arrival time e^d_i in order to make the best use of the driver's time.

$$r(d_{ik}) = \begin{cases} \dfrac{d_{ik} - c_i}{e^d_i - c_i}, & d_{ik} < e^d_i \\ 1 - \dfrac{d_{ik} - e^d_i}{l^d_i - e^d_i}, & d_{ik} \geq e^d_i \end{cases} \tag{5}$$

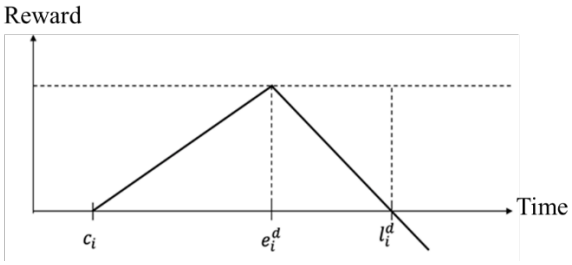

Fig. 4 Task agent's reward function

We choose ε-greedy selection as the task selection policy. This policy takes the action with the largest value $Q(s,a)$ in probability ε and takes random action in probability (1-ε). By setting this ε appropriately, the trade-off between centralization and diversification of search can be considered.

Value update is difficult to apply to bootstrap learning because it is difficult to predict the next state when the task and driver have taken actions. In addition, in the case of sequential updating, the value of waiting cannot be updated correctly because no reward is given when

the driver chooses to wait. Therefore, we choose to use Monte Carlo method which is non-bootstrap with the batch update.

4.3 Driver agent's condition and action

The driver agents' state in class P={1,⋯, 5} is defined by the loaded vehicle ratio of the presented tasks. The loaded vehicle ratio p_i of the task is defined by the following Equation (6).

$$p_i = \frac{t_i^d}{(t_{ij}^p + t_i^d)} \quad (6)$$

The driver agent's behavior is to accept or to reject the presented task. The policy is that the driver agent will accept if the class of the loaded vehicle ratio is above a certain threshold P_{min}, and to rejects if it is below it. Ant the threshold P_{min} is assumed to be the same for all drivers.

5 Experiment Design

We have made numerical experiments to verify the effectiveness of the proposed method.

In order to execute the experiment, the following input information is set up.

- $m = \{10,20,30\}$, $n = \{10,20,30,40\}$.
- $[P_i, D_i]$: made from polar coordinates (R_i, θ_i). R_i is made from uniform random numbers U(0,3000), and θ_i is made from uniform random numbers U $(0,2\pi)$.
- Speed of driver : $v = 250$[meter/min].
- t_{ij}^p: $\|D_j - P_i\|/\, v$[min], t_{ik}^d : $\|D_i - P_i\|/\, v$ [min].
- r_i: 325yen + $t_i^d \times 60/1000$yen.
- a_i: made from uniform random numbers U(540,1370).
- e_i^d: made from uniform random numbers U(a_i + 10, a_i + 30).
- l_i^d: $e_i^d + 30$.
- Driver's acceptance threshold to assigned task: $P_{min} = 3$.

The execution environment is IntelI Core™ i7-8700CPU (2.40GHz, 2.40GHz), 8.00GB Memory. The image diagram generated by the input information of pickup location and delivery location is shown in Figure 5 below. And the time zone from each task arrival time to late arrival time and the working hours of each driver are shown in Figure 6 below.

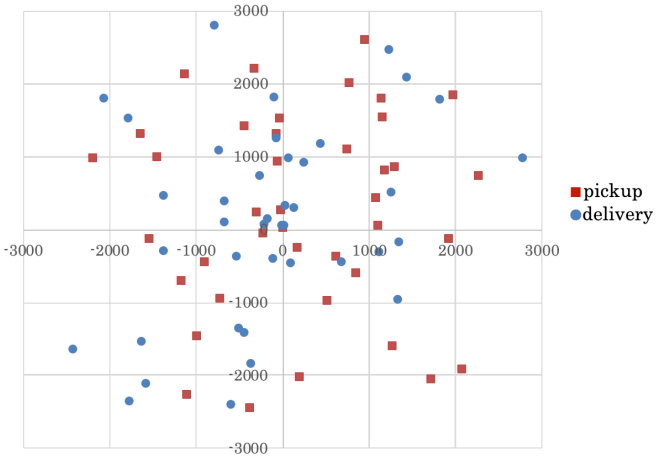

Fig. 5 Pickup location and delivery location

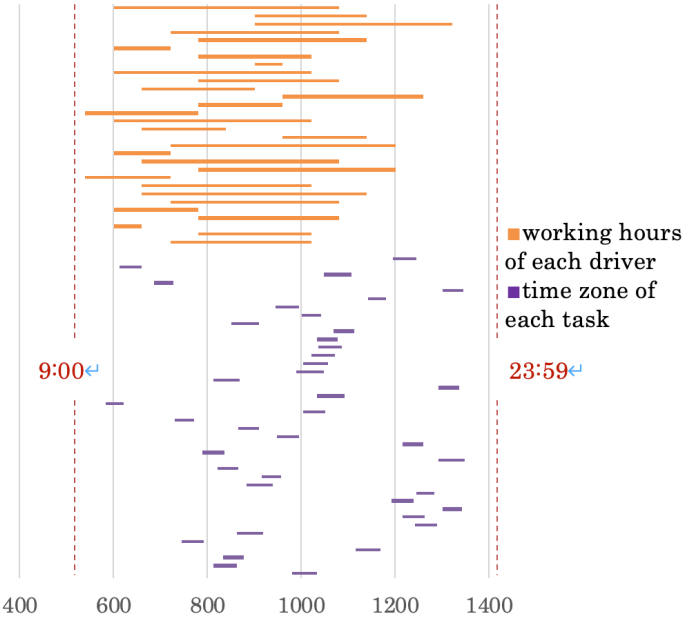

Fig. 6 Time zone of each task and drivers

6 Experiment Results

6.1 Numerical experiments to verify the proposed method

Table 4 shows the experimental results of the number of allocations for acceptances with the loaded vehicle ratio of class 3 or higher. ROTAP is the optimal solution for the offline problem (1), where the drivers' working hours and refusal information are known in advance. FIFO is the method of allocating tasks in the order of arrival time at the server. RANK is the method of allocating tasks to the driver with the shortest pickup distance.

Table 4 The number of allocation for acceptances (the average of 10 times)

m	n	Offline	Online		
		ROTA	FIFO	RANK	MARL
10	10	9.6	7.5	7.9	8.1
	20	19.4	16	16.5	17.7
	30	29.2	23.7	24.8	26.3
20	10	9.2	7.2	7.9	8.5
	20	19.6	16.9	18.2	18.7
	30	29.6	24.8	26.4	26.8
	40	39.6	35	37.2	37.6
30	10	9.9	8.1	9.1	8.9
	20	19.8	16.8	18.4	18.8
	30	29.6	25.2	27.4	28
	40	39.8	36.2	38	38.4

The proposed MARL method gives a better number of allocations than the FIFO and RANK methods. Since the higher the number of allocations, the lower the cost, the proposed MARL method seems to be more effective than the above two methods.

Table 5 The allocation rate of allocation for acceptances (the average of 10 times)

m	n	Offline	Online		
		ROTAP	FIFO	RANK	MARL
10	10	1.0	0.8	0.8	0.8
	20	1.0	0.8	0.8	0.9
	30	1.0	0.8	0.8	0.9
20	10	0.9	0.7	0.8	0.9
	20	1.0	0.8	0.9	0.9
	30	1.0	0.8	0.9	0.9
	40	1.0	0.9	0.9	0.9
30	10	1.0	0.8	0.9	0.9
	20	1.0	0.8	0.9	0.9
	30	1.0	0.8	0.9	0.9
	40	1.0	0.9	1.0	1.0

Table 5 shows the experimental results of the allocation rate for acceptance of class 3 and above actual vehicle rate. We compared the allocation ratio of the proposed MARL method with that of the FIFO and RANK methods. While m = 20, 30 and n = 20, 30, 40, the MARL method's results are close to that of the FIFO and RANK methods, but while m = 10 and n = 20, 30 or m = 20 and n = 10, the allocation ratio of the MARL method is higher than that of the FIFO and RANK methods. These also mean that the proposed MARL method is more effective than the FIFO and RANK methods.

Table 6 The competitive ratio of allocation for acceptances (the average of 10 times)

m	n	Offline	Online		
		ROTAP	FIFO	RANK	MARL
10	10	1.0	0.8	0.8	0.8
	20	1.0	0.8	0.9	0.9
	30	1.0	0.8	0.8	0.9
20	10	1.0	0.8	0.9	0.9
	20	1.0	0.9	0.9	1.0
	30	1.0	0.8	0.9	0.9
	40	1.0	0.9	0.9	0.9
30	10	1.0	0.8	0.9	0.9
	20	1.0	0.8	0.9	0.9
	30	1.0	0.9	0.9	0.9
	40	1.0	0.9	1.0	1.0

Table 6 shows the experimental results of the competitive ratio for acceptances with the loaded vehicle ratio of class 3 or higher. The competitive ratio is a performance evaluation index of the online algorithm for solving the online scheduling problem. The definition of the competitive ratio is based on Matsueda's paper[3] and is given in Equation (7) below.

$$\frac{OPT(L)}{v_a(L)} \leq h \tag{7}$$

$OPT\ (L)$ is the optimal value of the optimal solution to the offline problem (1), where the drivers' working hours and refusal information are known in advance. $v_a(L)$ is the evaluation value of the online algorithm. For a real number $h \in \mathbb{R}$, which is established in Equation (7), is the competitive ratio of the online algorithm.

Therefore, from the experimental results of the competitive ratios of the experiments of the methods in Table 6, it can be seen that the competitive ratio of the proposed method MARL is always less different from the best conflict ratio of 1.0 compared to FIFO and RANK methods. So, the MARL method is considered to perform better than the FIFO and RANK methods.

6.2 Experiment with changing the acceptance rate class

Table 7 shows the experimental results of the number of allocations of the actual car acceptance rate class 1 or more to actual car acceptance rate class 5 or more.

Table 7 The number of allocations with different accepted loaded vehicle ratio

Class	m	n	Offline	Online		
			ROTAP	FIFO	RANK	MARL
≧1	10	10	10	9.6	9.6	9.8
		20	20	19	19.2	19.2
		30	30	28.6	28.6	28.6
	20	10	10	9.4	9.4	9.4

		20	20	19.7	19.7	19.7
		30	30	29	29	29
		40	40	36	36	36
	30	10	10	9.8	9.8	9.8
		20	20	19.2	19.2	19.2
		30	30	29.5	29.5	29.5
		40	40	38	38	38
≧2	10	10	10	9.3	9.6	9.6
		20	20	18.6	19	19.4
		30	28.8	26.8	27	27.2
	20	10	10	9.6	9.6	9.7
		20	20	18.6	19	19
		30	30	28	28.6	28.6
		40	39.4	37.6	37.4	37.8
	30	10	9.8	8.8	9	9.6
		20	20	19.2	19.3	19.3
		30	30	27.5	29.3	29.3
		40	40	38	38	38
≧3	10	10	9.6	7.5	7.9	8.1
		20	19.4	16.0	16.5	17.7
		30	29.2	23.7	24.8	26.3
	20	10	9.2	7.2	7.9	8.5
		20	19.6	16.9	18.2	18.7
		30	29.6	24.8	26.4	26.8
		40	39.6	35.0	37.2	37.6
	30	10	9.9	8.1	9.1	8.9
		20	19.8	16.8	18.4	18.8
		30	29.6	25.2	27.4	28.0
		40	39.8	36.2	38.0	38.4
≧4	10	10	7.8	4	3.6	4.6
		20	15.4	5.4	7	9.4
		30	28	13	16.2	19.2
	20	10	8.4	2.6	4.0	4.8
		20	17.4	6.8	7.0	9.0
		30	27.6	10.6	13.4	15.8
		40	38.4	19.8	24.6	27.0
	30	10	8.4	3.2	3.8	5.0
		20	18.0	7.4	10.0	10.1
		30	26.5	8.5	8.0	12.0
		40	39.0	14.0	20.5	21.0
≧5	10	10	3	0.2	0.2	0.2
		20	9.0	0.6	0.8	1.0
		30	17.4	2	2.2	2.4
	20	10	3.2	0.6	0.6	0.8
		20	8.8	1.2	1.2	1.4
		30	16.6	2	2.4	3
		40	23.8	3.6	3.8	4.6
	30	10	3.3	0.3	0.3	0.3
		20	9.4	1.6	2.0	2.0
		30	13.6	1.2	1.2	1.6
		40	23.5	3.0	3.0	3.0

According to the experimental results of the number of allocations with different accepted loaded vehicle ratio, although the actual loaded vehicle ratio class is 1, 2 and 5, the proposed MARL method is less different from the FIFO method and the RANK method, the MARL method obtains a better number of allocations than the FIFO method and the RANK method. When the accepted loaded vehicle ratio class is 3 and 4, the proposed MARL method is different from the FIFO and RANK methods, and it can obviously obtain a better number of allocations. Therefore, we can say that the MARL method is more effective than the FIFO and RANK methods even when the actual acceptance rate class varies.

6.3 Experiment with changing the demand at different times of the day

We experimented with changing the frequency of task arrivals in response to the high number of customers requesting food deliveries during lunch and dinner hours. According to Uber Eats' research, the peak hours for delivery requests are lunch time (11AM-2PM) and dinner time (6PM-9PM) [6]. Therefore, we set the number as shown in Table 8 below. The probability of a task occurring between 11AM - 2PM is set to 0.4, the probability of a task occurring between 6PM - 9PM is set to 0.4, and the probability of a task occurring at other times is set to 0.2. The time zones for each task and the working hours of each driver, which have been changed are represented in Figure 9. And the results of the experiment are shown in Table 9 below.

Table 8 The setting of changing the demand at different times of the day

Start time	Ending time	Total time	Event probability	Cumulative event probability
11	14	3	0.40	0.40
18	21	3	0.40	0.80
9	11	2	0.04	0.84
14	18	4	0.09	0.93
21	24	3	0.07	1.00

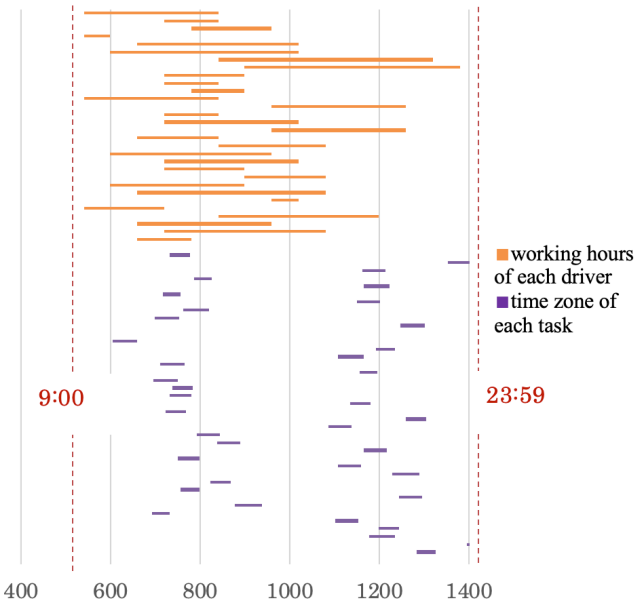

Fig. 7 Time zone of each task and driver in Experiment C

Table 9 The number of allocations with changing the demand at different times of the day

m	n	Offline	Online		
		ROTAP	FIFO	RANK	MARL
10	10	9.4	7.6	8.6	8.8
	20	19.2	16.2	17.0	17.8
	30	29.8	23.2	24.8	26.2
20	10	9.4	7.4	7.8	8.2
	20	19.2	15.6	18.0	18.0
	30	30.0	24.0	27.6	28.0
	40	39.5	32.5	34.5	36.3
30	10	9.8	7.2	8.2	8.4
	20	19.2	16.4	16.6	17.8
	30	30.0	24.7	27.3	27.3
	40	40.0	36.5	36.0	37.0

From the experimental results in Table 9, it can be seen that the number of allocations for the MARL method is higher than that for the FIFO and RANK methods, even when the frequency of task arrivals is changed during lunch and dinner. It is indicating that the MARL method is more effective than the FIFO and RANK methods.

7 Conclusion

Improving the efficiency of LMD has become an important social issue, and CS employment have been focused on as a means of securing employment opportunities. So, this article focuses on CSDPDP in order to solve the problem of efficiency improvement of LMD.

Although there have been studies on CSDPDP already, there are few studies that consider both the driver's refusal right and the buffering strategy. Therefore, **this article aims to improve the algorithm of CSDPDP by considering both driver's refusal right and buffering strategy.** Specifically, we propose a cost-minimizing driver-task matching algorithm that complies with the delivery time constraint using multi-agent reinforcement learning. Numerical experiments on the model show that the proposed MARL method is more effective than the FIFO and the RANK allocation methods. Even if the experiments are carried out in the case of changing the accepted loaded vehicle ratio class or in the case of changing the demand in different times of the day, the MARL method remains the most effective.

As a future prospect, **simultaneous processing of multiple tasks can be considered** in order to further reduce the cost. In addition, since we have trained the task, we can **consider building a model that also takes into account the learning of the driver.**

7.1 Simultaneous processing of multiple tasks

In food delivery, items are comparatively similar and there is little difference between large and small items. So, more than one item can be carried in one delivery. Also, there are cases where items are delivered from shops that are close together, or where items from the same shop are delivered to close destinations. In these situations, if multiple tasks are handled simultaneously, the delivery distance is reduced, and the overall cost is also reduced. Therefore, a future research topic could be the possibility for drivers to accept more than one task each time. Li [14] took a chain supermarket as an example, combined the distance factor and plans the distribution route of vehicle. And the results show heuristic algorithm can effectively solve the optimal solution of vehicle routing problem for single distribution center. In addition, Simultaneous handling of multiple tasks also changes the delivery routing, and it is one of the directions of future research to determine in which situations the costs can be reduced the most.

7.2 Learning of drivers

In this case, the task was learned. However, in reality, ad hoc drivers are hard to be controlled. In order to know each driver better, the future model could take the learning of drivers into account.

References:

[1] FY2019 E-Commerce Market Survey, Ministry of Economy, *Trade and Industry*, Japan, 2020

[2] FY2016 WHITE PAPER Information and Communications in Japan, *Ministry of Internal Affairs and Communications*, Japan, 2017

[3] Matsueda Y, Crowdsourcing Pickup and Delivery Problem Modeling and Economic Analysis, *Master Thesis of The University of Electro-Communications Tokyo*, 2020

[4] Berbeglia G, Cordeau J F, Laporte G. Dynamic pickup and delivery problems. *European journal of operational research*, 2010, 202(1): 8-15.

[5] Sayarshad H R, Chow J Y J. A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological*, 2015, 81: 539-554.

[6] Muñoz-Carpintero D, Sáez D, Cortés C E, et al. A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Science*, 2015, 49(2): 239-253.

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

[8] Ulmer M W, Thomas B W, Mattfeld D C. Preemptive depot returns for dynamic same-day delivery. *EURO journal on Transportation and Logistics*, 2019, 8(4): 327-361.

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

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

[11] Uber Help (Available: https://help.uber.com/ja-JP/driving-and-delivering, Accessed: March 1, 2021)

[12] Hyytiä E, Penttinen A, Sulonen R. Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives[J]. *Computers & Operations Research*, 2012, 39(12): 3021-3030.

[13] Tong Y, Wang L, Zimu Z, et al. Flexible online task assignment in real-time spatial data[J]. *Proceedings of the VLDB Endowment*, 2017, 10(11): 1334-1345.

[14] Xia Li, Optimization of VRP for Single Distribution Center Based on Improved Saving Method, *International Journal of Circuits, Systems and Signal Processing*, pp. 213-221, Volume 13, 2019.