Computers & Industrial Engineering 182 (2023) 109443

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Reinforcement Learning-based approach for dynamic vehicle routing problem with stochastic demand

Chenhao Zhou [a,*], Jingxin Ma [a], Louis Douge [b], Ek Peng Chew [b], Loo Hay Lee [b]

[a] School of Management, Northwestern Polytechnical University, China
[b] Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore

ARTICLE INFO

Keywords:
Dynamic vehicle routing problem
Stochastic demand
Single courier
Supervised learning
Reinforcement learning

ABSTRACT

This paper studies a dynamic vehicle routing problem under stochastic demands, drawn from a real-world situation. Specifically, a single courier must accomplish two kinds of tasks: deliveries known at the beginning of the operation and pickups that appear throughout the daily operation with specific patterns. The objective is to maximise the rewards obtained from serving both types of customers during a limited period. Our contribution lies in using the neural network and historical couriers' decisions to learn a base policy that captures human experience for better decision making. The reinforcement learning framework is then used to make the base policy explore new scenarios through simulations and further train the base policy with newly generated data. We show that our approach allows the serving of an average of 12% and 8% more customers under some conditions than the nearest-neighbour policy in high density area and low density area, respectively.

1. Introduction

The last-mile delivery industry has experienced tremendous growth for the past few years. Based on an industry report by IMARC Group (2022), the global courier, express and parcel market reached a value of 394 billion USD in 2021 and is expected to reach 519.6 billion USD by 2027. In this context, any improvement of the margins by a few per cent can provide a significant advantage. While large logistics providers like FedEx, DHL, and Alibaba Group increase investments in expanding their logistics networks, seeking to build scale and make profits from a low-margin activity, asset-light companies are adopting advanced technologies like routing algorithms to lower operating costs and improve operational efficiency.

Logistics companies serve both commercial and individual customers. Commercial customers such as e-commerce companies are usually located in commercial areas, while individual customers usually reside in residential areas. In Singapore, densely populated areas make it suitable for delivery companies to serve both commercial and individual customers together. Commercial customer requests may occur near residential areas where the probability of an individual requesting to receive or send a parcel is high. In this study, we examine the case of a large logistics company in Singapore serving both commercial and individual customers concurrently, whereby the couriers receive a set of delivery requests to individual customers beforehand while also picking up parcels from commercial and individual customers. The pickups are not known in advance and will occur during the day, but the requests have to be fulfilled within customers' available time. Pickup requests coming from commercial customers are usually less random than requests coming from individual customers. Some commercial customers tend to make pickup requests at some specific times of the day during the week.

The traditional delivery strategy adopted by the company gives autonomy to the couriers, allowing the couriers to plan their routes based on past experiences. Emphasis is put on deliveries (known customers) in the morning, in order to free time for pickups (unknown customers) in the afternoon. This approach seems reasonable as serving the deliveries first, gives more time for the pickups to appear. However, the traditional way could not fully utilize manpower as the company needs to keep a large number of couriers active in the city waiting for unknown demands. Besides, this traditional way does not make use of potential patterns of pickups appearance through time. If these pickup patterns are exploited, it would be possible to modify the courier's route ahead of time in order to increase the chances of being closer to new pickups while serving deliveries. Therefore, this study intends to improve the delivery and pickup process efficiency at the courier level by providing the sequence of requests to satisfy all delivery requests and serve as

* Corresponding author.
E-mail address: zhou_chenhao@u.nus.edu (C. Zhou).

https://doi.org/10.1016/j.cie.2023.109443
Received 8 July 2022; Received in revised form 23 June 2023; Accepted 10 July 2023
Available online 12 July 2023

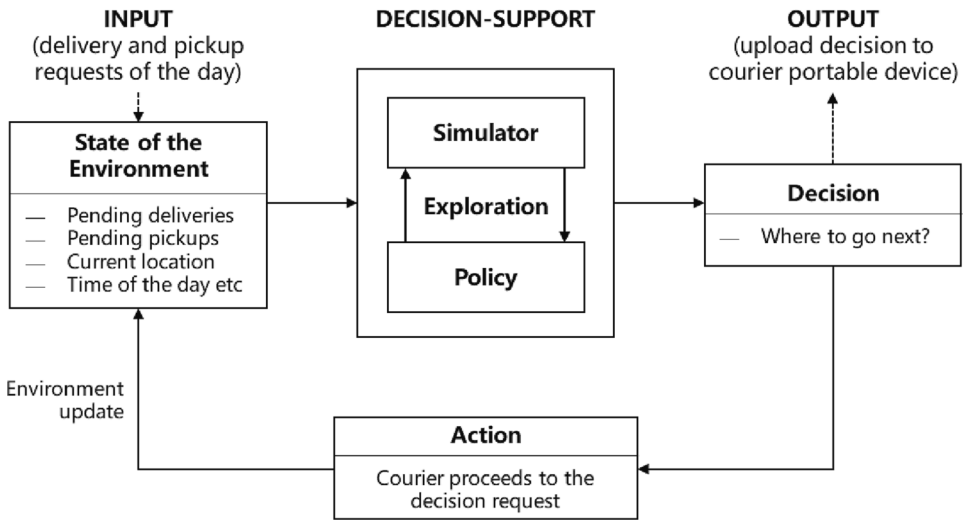


Fig. 1. Problem Overview.

many potential pickup requests as possible. In order to fulfill pickup requests, the routing plan may need to deviate from the shortest path slightly, while the new routing has a higher chance of serving the pickup requests which appear along the way. Given the stochastic feature of the pickup requests and the dynamic feature of making decisions, the problem can be considered as a dynamic vehicle routing problem with stochastic demand (DVRPSD), specifically a stochastic dynamic pickup and delivery problem (SDPDP) according to Soeffker et al. (2022).

With the successful use of Reinforcement Learning (RL) on large-scale sequential decision-making problems like Go and Chess games (Silver et al., 2018), we applied RL to this DVRPSD by considering patterns in the way pickups appear over time, to build a routing policy for efficient solutions. We start by building an environment to simulate realistic service times and demand scenarios. Then, a base policy is learned from historical decisions made by couriers in different settings. The policy is then improved with a reinforcement learning framework to include complex environment patterns represented as distributions learned from real-world data. The contributions of this paper are multifold: 1) an RL-based approach is proposed to solve the DVRPSD that outperforms a heuristic-based policy; 2) a base policy for delivery sequencing is derived from historical data containing the courier's real-life experience, which has been working quite well in real operation; 3) another policy is improved from the base policy by forecasting potential pickup demands along with the operation.

The rest of this paper is organized as follows. Section 2 provides a review of previous studies. A definition of the problem is given in Section 3. Section 4 introduces the environment of the problem and gives an overview of the dataset. Section 5 describes the methodology in detail, including service time estimation, supervised learning methods, pickup pattern generation, and reinforcement learning framework. In Section 6, the quality of our proposed policy is assessed by comparing it with a myopic policy based on the nearest-neighbor strategy. An overall conclusion of the study and ideas for future research is provided in Section 7.

2. Previous studies

Since first introduced by Dantzig and Ramser (1959), vehicle routing problems (VRPs) have been the focus of extensive research over the more than sixty years. Its numerous variants (Bai et al., 2023; Braekers et al., 2016; Vidal et al., 2020) have been extensively studied, including some emerging topics such as green VRP (Moghdani et al., 2021) and VRP with drones (Wang & Sheu, 2019). Given the nature of our problem whereby unknown pickup requests occur throughout the day and the service time is based on historic information, we need to focus on the dynamic version of this thoroughly studied problem.

Dynamic Vehicle Routing Problem (DVRP) is extensively studied in the literature where Psaraftis et al. (2016) and Rios et al. (2021) cover nearly 200 papers published in the literature up to 2021. Pillac et al. (2013) categorized two dimensions in the DVRP: first, information can be available from the beginning to the planner (static information), or it can only be available over time (dynamic information); second, the nature of information is threefold: it can either be completely known (deterministic), partially known (stochastic) or even unknown. Given the stochastic nature, the DVRPSD usually requires an online policy to quickly make decisions, which is generally modeled via a Markov decision process (Ulmer et al., 2020). The solution approaches can found in three categories: optimizing the routing problem given the current state information on a rolling-horizon fashion, identifying characteristics of valuable actions with the consideration of future dynamics (known as *policy function approximation*), and the combination of the above two types of the approaches (known as *cost function approximation*) (Hildebrandt et al., 2022; Soeffker et al., 2022).

Our problem focuses on the decision making for individual carrier and is dynamic and stochastic in the sense that the pickup information is unveiled through time and is partially known (through distributions). Hooshmand Khaligh and MirHassani (2016) studied a single-vehicle routing problem with stochastic demands where the actual demand of a customer becomes known only when the customer is visited. Klapp et al. (2018) investigated an order dispatching problem faced by a distribution center, where orders arise dynamically throughout a day. At each decision epoch, the system chooses whether or not to dispatch a single vehicle in order to minimize vehicle travel costs and penalties for unserved requests. Ulmer and Thomas (2020) examined the capacitated customer acceptance problem with stochastic requests in which a company seeks to maximize expected revenue by accepting or rejecting requests. Early works have widely adopted value function approximation and rollout algorithms as solution approaches. Ulmer et al. (2019) combined offline value function approximation with online rollout algorithms by incorporating spatial and temporal anticipation of requests and routing developments. Yet, Zhang et al. (2023) pointed out that value function approximation approaches are restrained by the high dimensionality of lookup tables, while the rollout algorithms are restrained by the number of simulated scenarios.

The recent renewed interest in RL has shown that, under some assumptions, it is possible to tackle many problems without even considering a model (Sutton & Barto, 2018) and is also possible to achieve impressive results obtained by using deep learning techniques

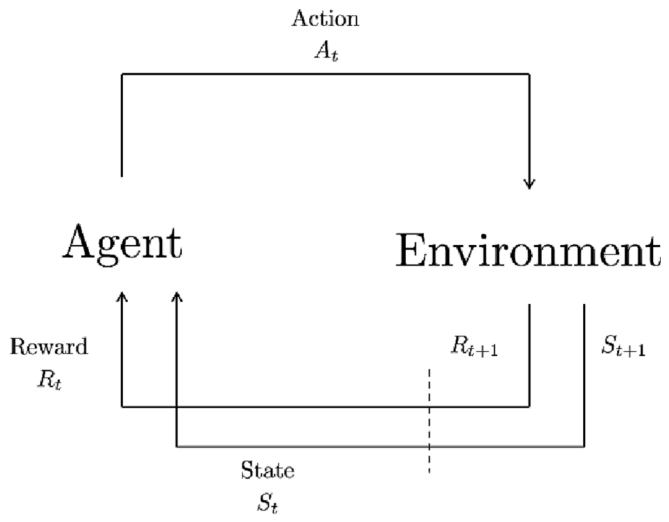


Fig. 2. Reinforcement Learning Framework.

(Goodfellow et al., 2016; Silver et al., 2016). In the context of dynamic and stochastic VRP, (Basso et al., 2022) proposed a safe reinforcement learning method to solve dynamic routing of electric commercial vehicles with the consideration of uncertain energy consumption and random customer requests. Chen et al. (2022) addressed the dynamic delivery problem with fleets of drones and vehicles and presented a deep Q-learning approach. Ma et al. (2021) looked into the dynamic pickup and delivery problem and combined the reinforcement learning-based policies with a hierarchical optimization framework. Given the rare literature above, applying the RL framework to solve DVRPSD, especially its special case SDPDP, remains an interesting combination. With historical data from couriers' daily operations, RL can be utilized to extract insightful information for better decision making, which is much different from the mathematical programming approach.

3. Problem description

We consider one delivery vehicle operating in a given region and serving two types of jobs, namely deliveries, known before the vehicle's departure, and pickups, appearing at random places over time. Our work focuses on morning operations only. This period is critical as most of the deliveries must be satisfied while a large part of the daily pickups' requests also appears. An efficient delivery procedure during this period allows reallocating manpower to other operations in the afternoon. The goal of the courier is to serve as many pickup and delivery requests as possible, while the priority is set on deliveries.

As illustrated in Fig. 1, our work is to assist the courier to make a better decision on which request to serve next in a dynamic environment. Given that the courier is at a certain place, he will receive advice from a portable device on which request to visit next. He then moves to the decided location with the possibility of having new pickups appearing while driving towards the place. Once arrived on site, the courier completes the request before deciding again which customer to visit next. This process continues as long as the departure time towards the next customer happens before the time constraint (for example, 12 pm).

This problem fits well with the paradigm of RL, as shown in Fig. 2, which makes up the decision-support framework in Fig. 1. At its core, the RL framework is based on an agent interacting with an environment and receiving rewards depending on the evolution of this environment, while the state is updated based on the agent's action and the changing environment. We first assume that a reward will be given when completing a delivery or pickup request, and deliveries produce higher rewards than pickups due to the need to complete deliveries first in the morning. The courier is the agent and we create a close-to-reality environment that is able to generate service times and random pickup requests. The agent interacts with the environment by taking action (i.e., the courier's decision on which place to visit the next) that will produce a new state (i.e., the courier's remaining requests, new location and current time) and receives the reward. This process repeats itself over an unknown number of steps but over a finite time horizon (morning operation).

Ideally, the objective is to find a policy that produces the most accumulated rewards by serving successively any of the available customers without violating the time constraint. As the future unfolds in an unknown way, the objective for the agent is in fact to maximize its expected accumulated rewards. The sequential nature of the framework makes it possible to optimize the policy of making decisions given a particular state, i.e., when the courier is at a certain location and time. This is desirable as our agent has to be able to adapt to patterns over time, like traffic or pick up trends.

The above problem can be formulated as below. We consider L locations $\{1, \cdots, L\}$. The number L represents the number of postal codes considered in the experiments. Note that in Singapore and many other countries, each building has a unique postal code. To apply to other countries, the postal code can be replaced by a unique location index. At step k, we define the state $S_k = (l_k, t_k, D_k, P_k)$ from the state space S with:

Current location $l_k \in \{1, \cdots, L\}$ (the last served location);

Current time t_k (continuous value) when leaving to the next customer;

Set of locations of remaining customers of type deliveries: $D_k = \left\{d_k^{(1)}, d_k^{(2)}, \cdots\right\} \subset \{1, \cdots, L\}$;

Set of locations of remaining customers of type pickups: $P_k = \left\{p_k^{(1)}, p_k^{(2)}, \cdots\right\} \subset \{1, \cdots, L\}$.

The next state $S_{k+1} = (x_{k+1}, t_{k+1}, D_{k+1}, P_{k+1})$ is updated based on (S_k, x_k, s), where.

At time t_k, the agent decides which location x_k (x_k represents a location decision, i.e., l_{k+1}) to visit among the action space $x_k \in A_k = D_k \cup P_k$ (waiting customers), a subset of the action space $A = \{1, \cdots, L\}$;

$t_{k+1} - t_k$ is service time (including travel time and interaction time with the customer) which can be derived from a function of distance;

$D_{k+1} = D_k$ if x_k is a pickup, else $D_{k+1} = D_k \backslash \{x_k\}$;

P_{k+1} is updated from P_k by unveiling pickups with a time of appearance comprised between t_k and t_{k+1} according to a certain scenario s;

Scenario s, including requests of deliveries, pickups, as well as their time of appearance, are sampled from distributions and generated as scenarios beforehand. The agent is not aware of the full set of scenarios when starting the operations; he only sees the unveiling of the scenarios provided by the simulator depending on his decisions.

Reward obtained for a single-stage where β_d is for delivery, β_p is for pickup, and $\beta_p < \beta_d$ (as deliveries are more important than pickups):

$$g(S_k, x_k) = \begin{cases} \beta_d, if\, x_k \in D_k \\ \beta_p, if\, x_k \in P_k \end{cases}$$

Objective function: to find the best policy π that maximizes the total expected reward over a specific time horizon, i.e., 12 pm,

$$\max_{\pi \in \Pi} \mathbb{E}^{\pi} \left\{ \sum_{k=0}^{T} g(S_k, \pi(S_k) | S_0) \right\}$$

with

$$T = \max\{k \in \{0, 1, 2 \cdots\} | t_k \leq 12pm\}$$

The optimal way of solving the objective function is to start from what is identified as the final state and find the optimal decision at each step following a backward induction. However, defining what the final state is is impossible in our case, as the number of steps is not known in advance. Therefore, our focus is not to propose an intractable way of

Table 1
Example of Daily Data for a Single Courier.

Stop Order	Time Reached	Postal Code	Longitude	Latitude	Type	Pickup Ready Time
1	9:51am	68,902	103. 84,901	1. 279,100,001	Delivery	NA
2	10:02am	68,807	103.84983	1. 278,710,004	Delivery	NA
3	10:03am	68,807	103.84983	1. 278,710,004	Pickup	9:30am
4	10:06am	68,807	103.84983	1.278710004	Delivery	NA
5	10:08am	68,807	103.84983	1.278710004	Delivery	NA
6	10:12am	68,805	103.84981	1.278680002	Delivery	NA
7	10:14am	68,805	103.84981	1.278680002	Delivery	NA
8	10:19am	68,804	103.85029	1.279470001	Pickup	9:30am
9	10:22am	69,904	103.85029	1.279470001	Delivery	NA
…	…	…	…	…	…	…
58	18:29 pm	688,890	103.84981	1. 278,680,002	Pickup	17:51 pm

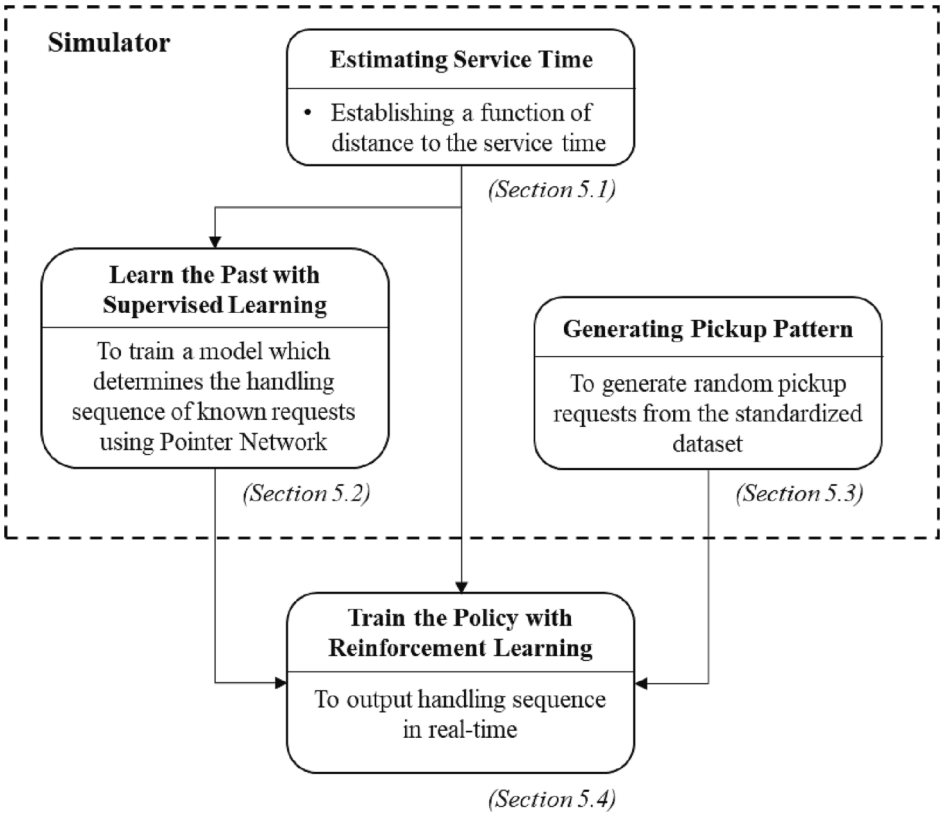


Fig. 3. Reinforcement Learning based Decision-support Framework.

finding the optimal solution but to create a solution approach leading to very good performance.

4. Environment setting

This study is motivated by a real case faced by a large logistics company in Singapore, and adopts a real dataset consisting of about 100,000 samples collected over a month. Each sample represents one-stop to serve a customer (either delivery or pickup). The amount of data is equivalent to about 6,000 parcels delivered or retrieved daily by on average 300 couriers / 130 trucks. Many features are available such as the current location of the courier, the time serving the customer, the time when a pickup becomes known to the courier, the truck ID, remaining deliveries/pickups at each location, etc.

In reality, in order to maximize one's profit, the individual courier will try hard to optimize his choice on request and route. For example, he may learn from his own experience, radio, mobile app, or chit-chat about congested or blocked roads, convenient parking locations, or short-cuts. Despite the fact that a mobile app such as Google Map could provide accurate spot-to-spot information on traffic conditions and estimated arrival time, the courier may not travel as planned. Instead, he may park the vehicle where is within walking distance to several requests, and then visit each customer on foot. Besides, one courier is usually recruited from the target area and assigned to this area, in order to take advantage of his local experience. By learning on his own intelligence, he will achieve decent performance by choosing the most convenient route for him to serve the customers within the target area.

Therefore, in order to learn from the drivers and obtain better results, we filtered the dataset to tours done by one single courier with the following characteristics: (1) one month of daily operations, and (2) the area with higher request density. The dataset has 103 unique postal codes and 2,759 samples, including delivery and pickup requests. Table 1 gives a glance at the dataset.

5. Methodology

Fig. 1 highlights that the fundamental step is to construct a simulator that represents the environment in which a policy can act. After the

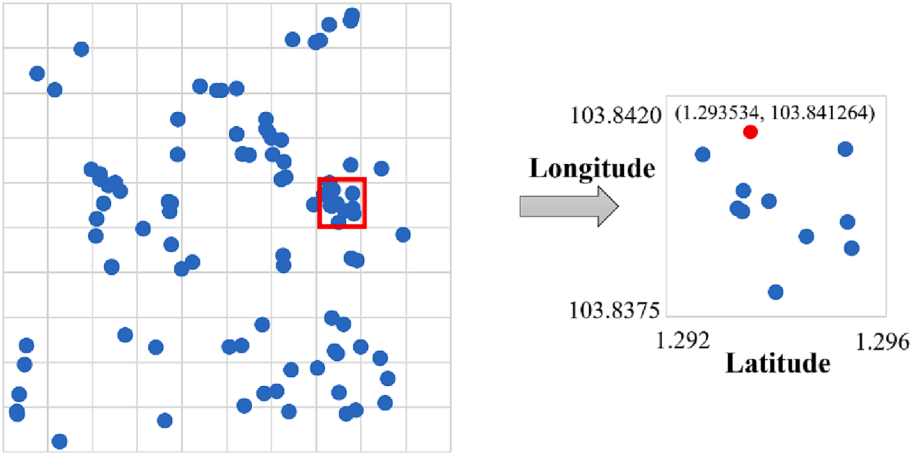


Fig. 4. Single Courier Data Illustration.

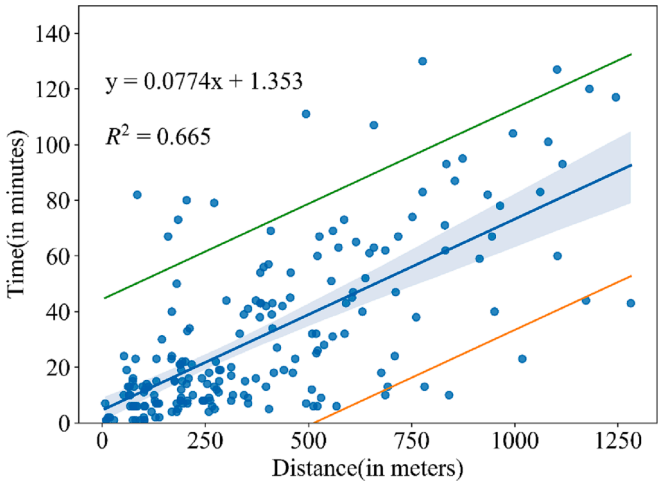


Fig. 5. Service Time Estimation Model.

simulator is built, the agent (a virtual courier) runs against the simulator with a possibility for exploration. For example, instead of following the simulator's decision, the agent chooses another next location which may possibly serve pickup requests appearing along the way. From the newly generated data, we would be able to develop a powerful policy using the RL.

It is well-known that the courier achieves decent performance at his task by anticipating congested roads or future delivery/pickup demands. However, these features are mostly intuition-based. We want to learn these features in order to construct a simulator that represents the environment. Considering the uniqueness of each courier and corresponding service area, the simulator is built for individual courier. The simulator has three major functions, (1) to calculate the service time between requests (Section 5.1), (2) to make a decision on which the next place to visit given its current location (Section 5.2), and (3) to generate pickup requests at a random and patterned time and locations (Section 5.3). Given the dataset that all pickup and delivery requests have been fulfilled, a base policy (denoted as *SL-policy*) which represents how the courier travels based on temporal and spatial context (i.e., service time) can be trained using supervised learning techniques. Although the SL-policy may not perform well on anticipating random pickup requests, it could still make a good decision for the courier to serve as many customers as possible.

Given this simulator, the RL framework can be adopted to derive a look-ahead policy by anticipating pickup trends (Section 5.4). While the agent repeatedly interacts with the simulator, it is possible to optimize the original policy by maximizing the agent's expected accumulated rewards. Given the particular state, i.e., when the courier is at a certain location and time, the improved policy (denoted as *RL-policy*) will help the courier to decide which location to go next. After the courier drives to the decision location, the state of the environment will be changed, and the above process will repeat.

In summary, for ease of understanding, Fig. 3 illustrates the composition of the decision-support framework, including how a simulated environment is built and how the policy using the RL framework is trained to output better solutions.

5.1. Estimating service time

In the given dataset of one courier working for a month, the latitude ranges from 1.264 to 1.304 (equivalent to 5008.17 m), and the longitude ranges from 103.815 to 103.860 (equivalent to 4423.00 m), as illustrated in Fig. 4. Usually, the requests for one courier are concentrated distributed, and it is reliable to use the time reached each customer and the Euclidean distance between customers (by GPS coordinates) to establish a function of distance to the service time (Zhao et al., 2020).

Fig. 5 presents the service time estimation model. By using the liner regression method, the model $t = 0.0774x + 1.353$ (with $R^2 = 0.665$ in the 95% confidence interval) can be derived to calculate the service time between two requests.

5.2. Learn the past with supervised learning

While the agent is interacting with the simulator, the agent will make a decision on the next place to visit. To take advantage of the courier's intelligence which has efficiently and effectively served his customers, a SL-policy can be built based on his past decisions. Thus, deriving the policy becomes a supervised learning task and the neural network is adopted. Past data provides input vector X and the associated output Y for the neural network training. Specifically, the input of the network $X = \{x_1, x_2, \cdots, x_k, \cdots\}$ is the set of customers served from step 1 to k. The element of the input vector x_k includes λ_k^{lat} and λ_k^{long} (current location, denoting by latitude and longitude, at step k) and γ_k (type of request at step k, either delivery or pickup). The output $Y = \{0, y_1, y_2, \cdots, y_k, \cdots\}$ is the actual service sequence. The first element represents the depot and the rest $y_k \in \{1, 2, 3, \cdots, k\}$ is the sequence index. For example, if $Y = \{0, 4, 2, 1, 3\}$, then the service sequence will be the 4th, 2nd, 1st, and 3rd requests. The predicted output obtained from the neural network is denoted by $\widetilde{Y}$.

The first model we considered is a fully connected 4-layer neural network (denoted as FourNN), with three hidden layers as presented in Fig. 6. The input layer has $L+1$ units representing current time and

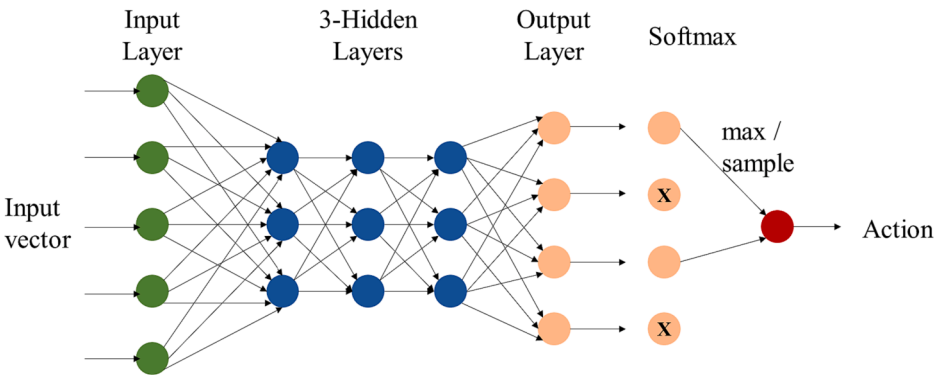


Fig. 6. The 4-Layer Neural Network.

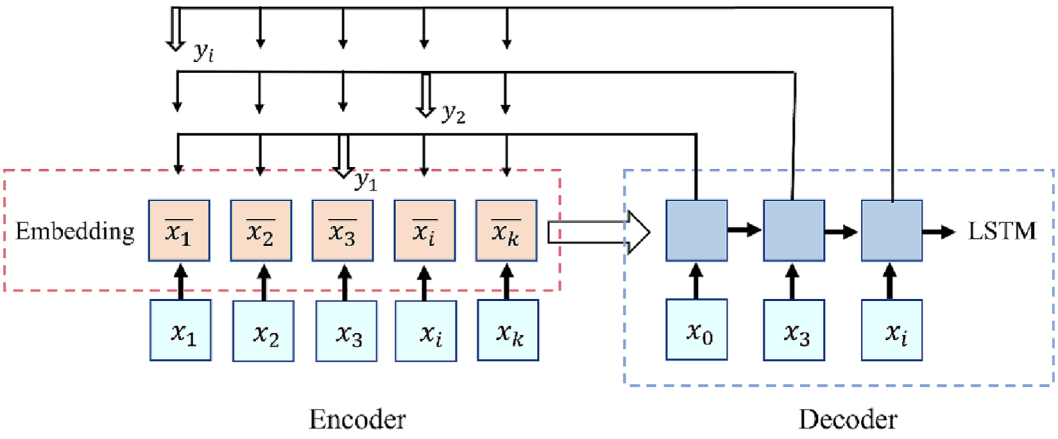


Fig. 7. Illustration of the revised Pointer Network.

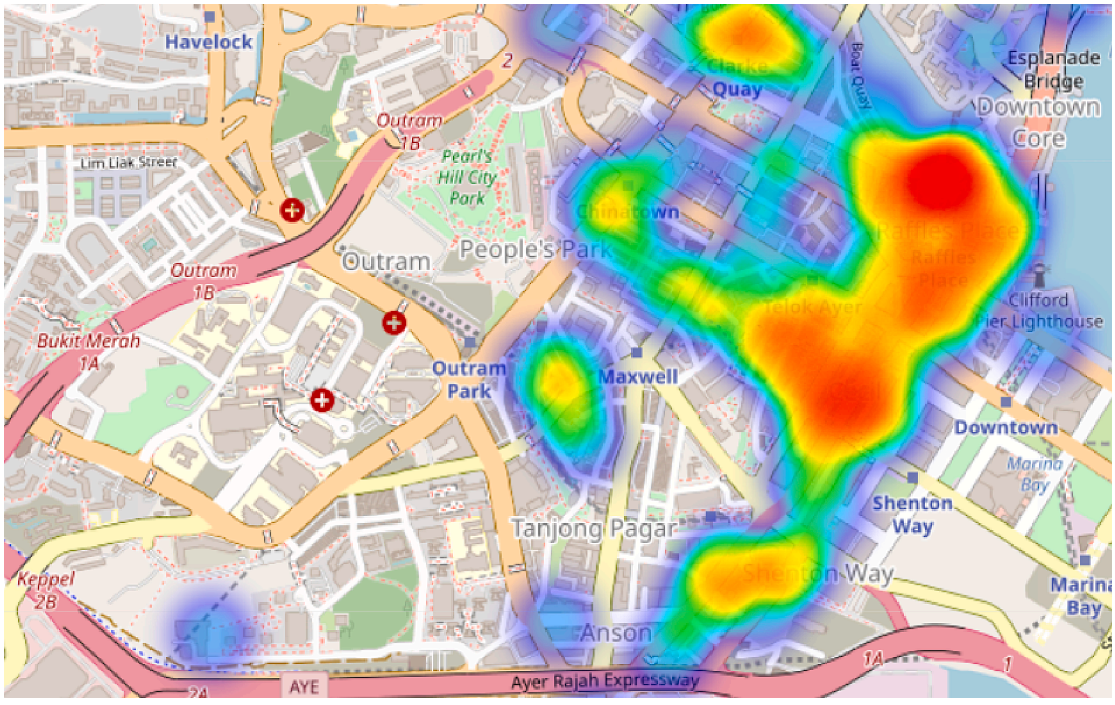


Fig. 8. Pickup Probability Distributions.

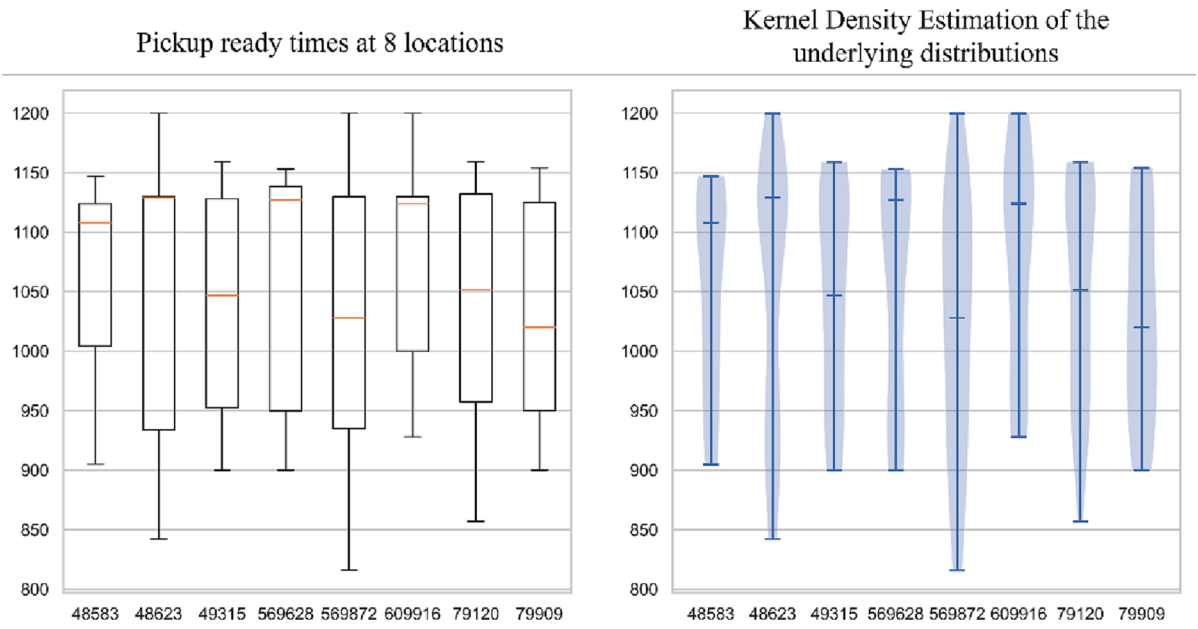


Fig. 9. Boxplot and Violin Plot of Pickup Appearance Times.

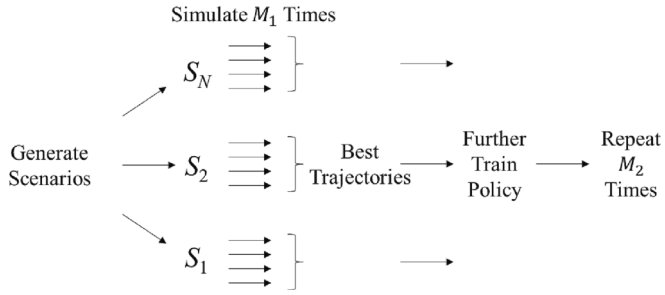


Fig. 10. Procedures of Policy Enhancement.

Table 2
Experiment Results.

Model	Optimizer	Batch Size	Learning Rate	Training Accuracy	Testing Accuracy
Ptr-Net	Adam	64	***0.0005***	***0.978515***	***0.506234***
			0.001	0.980131	0.508675
			0.005	0.513392	0.46875
			0.01	0.553571	0.470703
			0.05	0.455078	0.423828
			0.1	0.242187	0.25
		32	0.001	0.558593	0.453125
	SGD	64	0.0005	0.392857	0.316406
			0.001	0.392857	0.316406
			0.005	0.549107	0.470703
			0.01	0.566964	0.492187
			0.05	0.730468	0.479048
			0.1	***0.857142***	***0.500953***
		32	0.1	0.855468	0.500312
FourNN	Adam	64	***0.001***	***0.670235***	***0.355096***
		32	0.001	0.562579	0.330134
	SGD	***64***	***0.1***	***0.498023***	***0.295062***
		32	0.1	0.507824	0.265973

states of L locations while the output layer has L units. By applying a softmax function to this final layer, we obtain a probability distribution over all the categories (L locations). In other words, from the softmax layer, given a certain input, we have the probability that any of the L locations are to be selected as the decision. By sampling or selecting the unit with the highest probability, a decision is finally returned.

As the standard 4-layer neural network cannot fit well into the decisions with strong sequence correlation, Vinyals et al. (2015) introduced a new neural architecture to learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence, called Pointer Networks (Ptr-Net). The Ptr-Net has been shown to be effective in learning approximate solutions to the planar Travelling Salesman Problem and is suitable for our problem.

In the original Ptr-Net, the Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) is used to model the conditional probabilities for the training set. Note that the LSTM is useful when the elements of the input vector have a sequence (not the input vector has a sequence). For example, the words in a sentence have to follow a sequence according to the gramma. However, the input of our study is a set of customer requests, and the geographic information is disordered. Thus, the LSTM is replaced by the embedding process in our Ptr-Net that allows reducing the model complexity, as illustrated in Fig. 7.

Note that the output decision from the neural network at any step needs to exclude the deliveries/pickups that have been served. Therefore, a pre-process is implemented by removing the non-relevant decisions ($x \notin D_k \cup P_k$) before using softmax function.

To train this fully connected neural network, Stochastic Gradient Descent (SGD) (Bottou, 2010) and Adaptive Moment Estimation (Adam) (Kingma & Ba, 2014) methods are used to minimize the cross-entropy loss function with some regularization terms.

5.3. Generating pickup pattern

We use the Kernel Density method to generate the geographical density distributions of pickup requests. Fig. 8 shows the pickup probability distributions. The areas with dark colors have a higher density of pickup requests than the areas with light colors. Bandwidth h acts as a smoothing parameter of the density estimation, which sets the balance between bias and variance in the obtained distributions. A larger bandwidth will make the distribution over smoothed, while a smaller value will generate under smoothed distributions. Here, the bandwidth has been chosen relatively small, i.e., $h = 0.2$, in order to capture local density trends.

Time at which pickups become known across different locations (represented as postal codes) are used to generate Boxplot and Violin

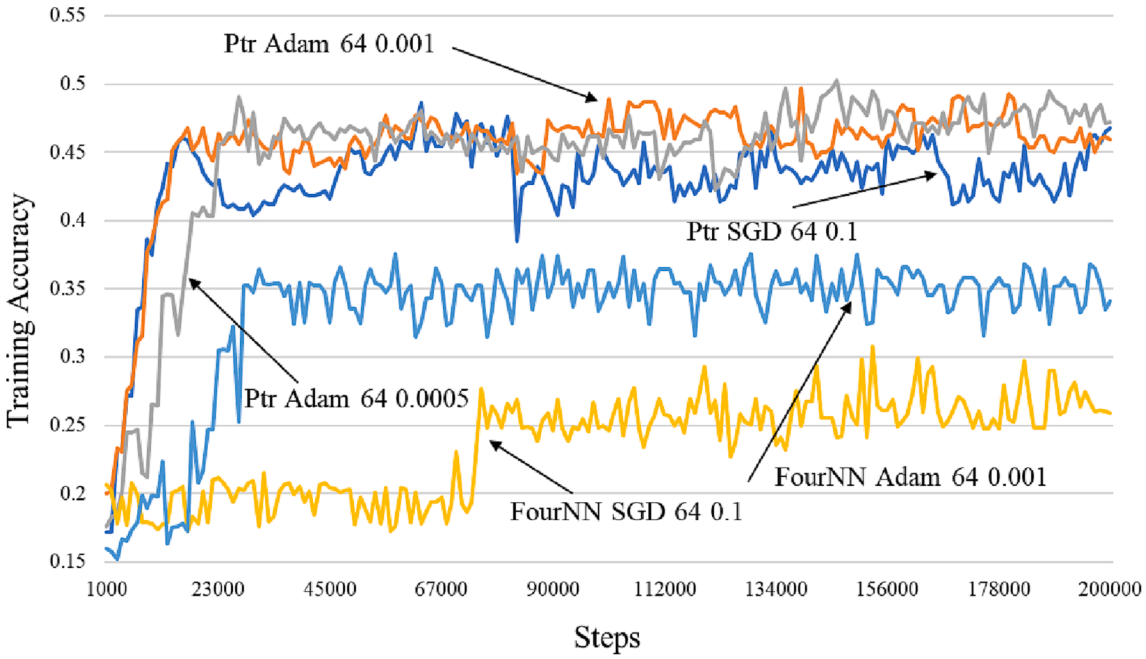


Fig. 11. Training Accuracy – Maximal 200,000 Steps.

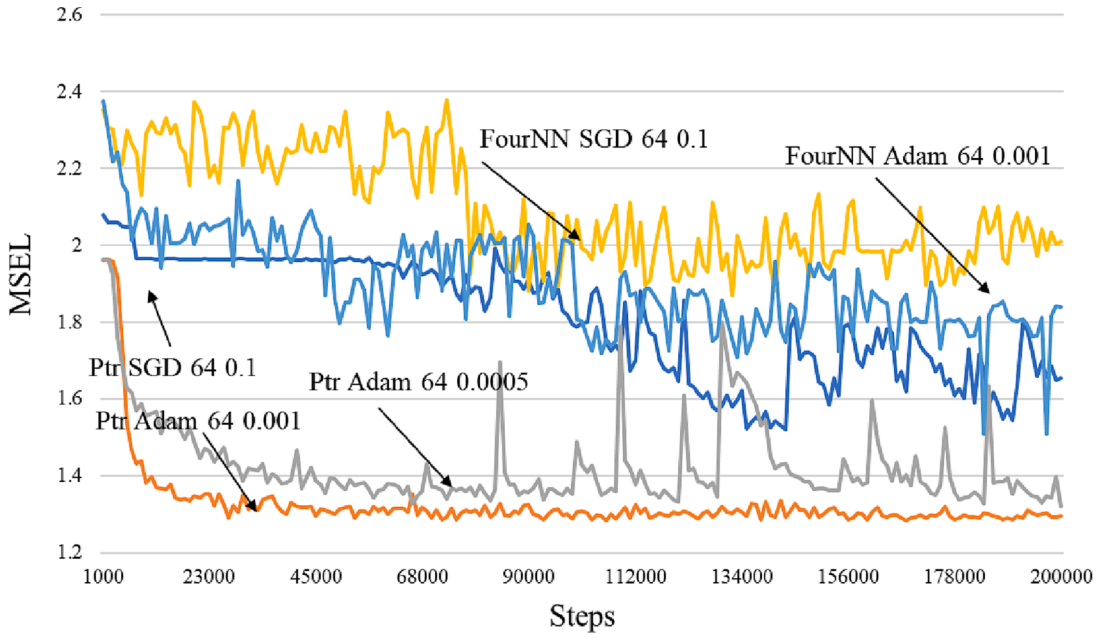


Fig. 12. Mean Squared Error Loss – Maximal 200,000 Steps.

plots (shown in Fig. 9). Kernel Density methods and boxplots give a good idea of how the data is distributed over time and locations. It can be observed that the data has a relatively high variance; however, for some locations, a pattern appears with pickups having a tendency to appear regularly around the same time (e.g., location with postal code 48,623 where pickups tend to appear almost close to the noon). For the needs of our study, simple normal distributions are adopted.

5.4. Train the policy with reinforcement learning

Given a set of new requests, the model obtained from supervised learning is able to generate a service sequence for the courier to follow. In reality, while the courier is working on the delivery requests, he may need to turn to pickup requests popping up along the way at any time, which cannot be captured by the supervised learning model. Thus, an RL framework is proposed as described in Section 3, which allows pickup requests to be generated based on the pickup pattern.

Initially, the framework will generate the service sequence (the SL-policy) using the supervised learning model and then instruct the agent to interact with the environment and accumulate rewards by taking an action. While the state is updated based on the agent's current action and the changing environment (e.g., a new pickup request appears), the agent will need to take another action until no further actions can be taken, or the time constraint is reached. Through policy in exploration allowing random behavior (i.e., random next location), the framework can generate many scenarios and identify the improved

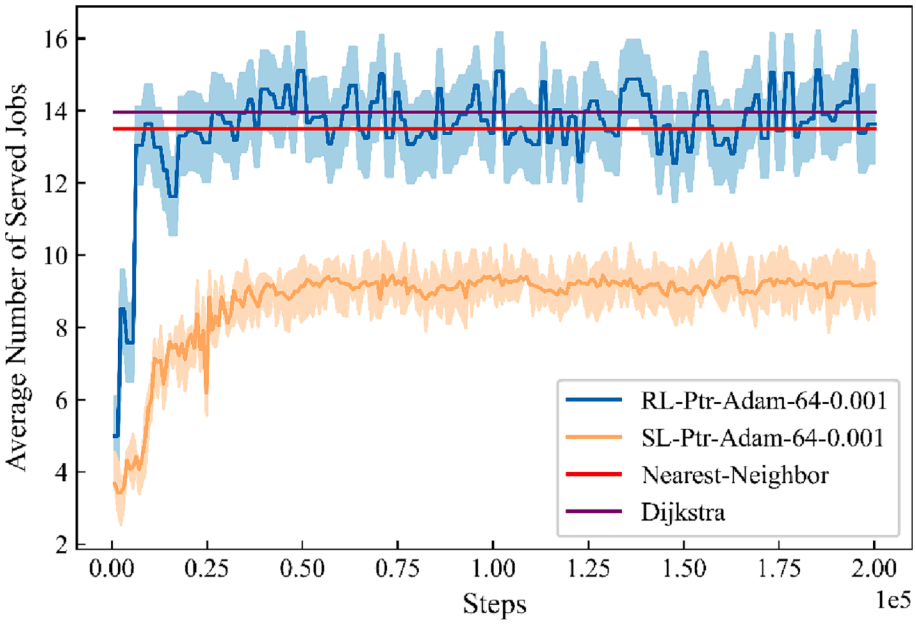


Fig. 13. Policy Comparison.

trajectories in terms of rewards, and eventually enhance the policy. The overall procedure is presented in Fig. 10.

In the real-world application, deliveries are known in advance, so the number of pickups appearing after time t, and corresponding locations can be obtained from the dataset. Therefore, the following notations are defined:

$\mathcal{N}_d(t)$: number of deliveries during the whole operations period.

$\mathcal{G}_d$: geographical location of deliveries.

$N_{\mathrm{d}}\ (i,t)$: number of customers at location.$i(\in\{1,\cdots,L\})$

However, the pickups will have to be estimated and sampled from the distributions above, and relevant notations can be found below:

$N_p(t)$: number of pickups appearing after time.t

G_p: geographical distribution of pickups.

$N_{\mathrm{p}}\ (i,t)$: number of customers at location.$i(\in\{1,\cdots,L\})$

R_i: the appearing-time distribution of a pickup at location.$i(\in\{1,\cdots,L\})$

Hence, scenario generation will follow the procedures presented in Algorithm *generateScenario*(t_k, D_k, P_k), where t_k represents starting time, D_k for existing deliveries, and P_k for existing pickups.

Algorithm ***generateScenario***$(t_k, \boldsymbol{D_k}, \boldsymbol{P_k})$

Input t_k starting time,
D_k existing deliveries,
P_k existing pickups
Output Scenario s
Initialization Scenario s including D_k and P_k
// Sample number of locations
n_d $N_d(t_k)$⊳ number of delivery locations appearing after t_k
n_p $N_p(t_k)$ ⊳number of pickup locations appearing after t_k
// Sample deliveries
for $i = 1$ to n_d **do**
$g_d^{(i)}$ G_d location
$n_d^{(i)}$ $Nd(g_d^{(i)}, t_k)$ number of deliveries at $g_d^{(i)}$
Add delivery $(g_d^{(i)} : n_d^{(i)})$ into scenario
// Sample pickups
for $j = 1$ to n_p **do**
$g_p^{(j)}$ G_p ⊳ location
$n_p^{(j)}$ $Np\left(g_p^{(j)}, t_k\right)$ ⊳ number of pickups at $g_p^{(j)}$
Add pickup $(g_p^{(j)} : n_p^{(j)})$ into scenario
for $l = 1$ to $n_p^{(j)}$ **do**
r_l^j $R_{g_p^{(j)}}$ ⊳ Time of Appearance

(*continued on next column*)

(*continued*)

Algorithm ***generateScenario***$(t_k, \boldsymbol{D_k}, \boldsymbol{P_k})$

Add appearance time r_l^j to $g_p^{(j)}$
return s

Given the generated scenarios, the procedure to perform policy exploration and retrieve the best path among all the explored paths (obtaining "best trajectories" in Fig. 10) is given in Algorithm *bestTrajFromScenario*(S, M_1), where s is the scenario and M_1 is the number of simulations.

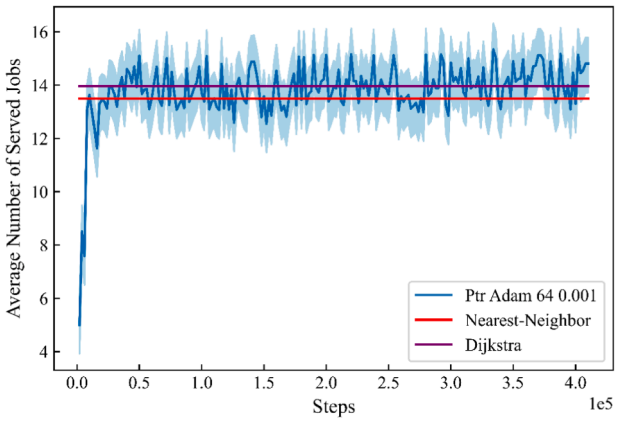
Algorithm ***bestTrajFromScenario***$(s, \boldsymbol{M_1})$

Input s Scenario (typically output of generateScenario()),
M_1 integer, number of Scenario simulations
Output traj*, state-decision samples from the best performing trajectory
Initialization i←1
best_score ←0
best_traj ←[]
S_k← select the initial state from scenario s
while $i \le M_1$ **do**
score ← 0
traj ←[]
// As long as the deadline is not reached
while $S_k.t_k \le 12pm$ **do**
decision ← $\pi_0(S_k)$ // the policy in exploration
S_k← generate next state based on $(\text{decision}, S_k, s)$
if decision ∈ D_k **then**
score ← score $+\beta_d$
else
score ← score $+\beta_p$
traj.append($[S_k, \text{decision}]$)
// update current highest value trajectory
if score > best_score **then**
best_score ← score
best_traj ← traj
i←$i+1$
return traj* = best_traj

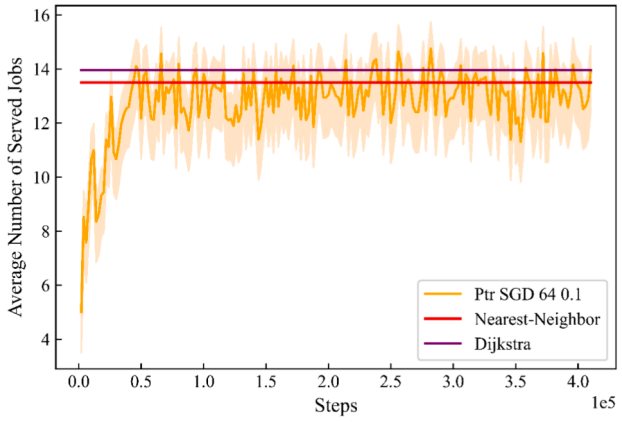
The policy in exploration is denoted as π_0 and defined as follows:

$$\sim \pi_0(S_k) = \begin{cases} \widetilde{x} & \textit{with probability } \int \in [0,\ 1] \\ \pi_0(S_k) & \textit{with probability } 1-\int \end{cases}$$

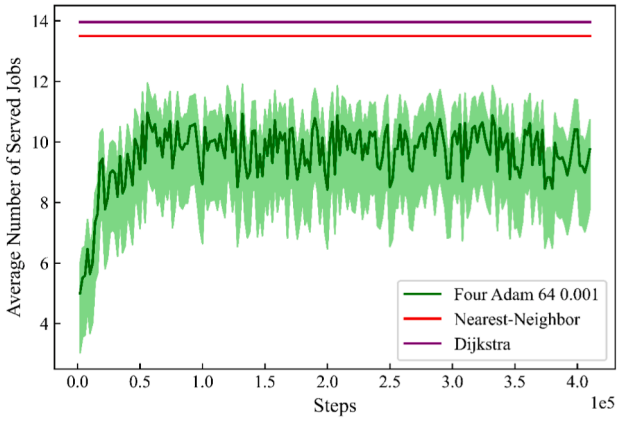
where $\widetilde{x}$ is chosen (uniformly) randomly in $D_k \cup P_k \backslash \pi_0(S_k)$, the set of possible actions minus the one picked by the policy. Through policy



(a) Training initialed from Ptr-Adam-64-0.001



(b) Training initialed from Ptr-SGD-64-0.1



(c) Training initialed from FourNN-Adam-64-0.001

Fig. 14. Policy Comparison using Different Initial Policies.

exploration, the procedure can generate many virtual trajectories of the courier and identify the trajectory with the highest cumulative rewards as the best decision. *State-decision* samples of such trajectories are then collected to re-train the initial policy. This process *policyimprovment*(π_0, N, M_1, M_2) constitutes a cycle that we repeat M_2 times as follows:

Algorithm ***policyimprovment***(π_0, N, M_1, M_2)

Input π_0 the SL-policy
N integer, number of Scenarios
M_1 integer, number of simulations per scenario
M_2 integer, number of learning cycles

(*continued on next column*)

(*continued*)

Algorithm ***policyimprovment***(π_0, N, M_1, M_2)

Output π_0, further trained policy
Initialization $l \leftarrow 1$ ▷ cycle counter
$n \leftarrow 1$ ▷ scenario counter
while $l < M_2$ do
training_data $\leftarrow []$
// Generate improved state-decision samples
while $n < N$ do
$s \leftarrow generateScenario(8am, [], [])$
best_traj $= bestTrajFromScenario(s, M_1)$
training_data.append(best_traj)
$n \leftarrow n + 1$
// Further train the policy with neural network
$\pi_0 \leftarrow furtherTrain(\pi_0, training_{data})$
$l \leftarrow l + 1$
return π_0 the RL-policy

Finally, in practice, the couriers can utilize the RL-policy trained offline to decide which next location to visit suppose they do not have enough time to make decisions. However, when sufficient time is allowed, the policy can be used in a Sample Average Approximation (SAA) algorithm to evaluate the downstream value of taking the best actions. The brief introduction is presented in supplemental online material.Supplementary Material.

6. Numerical study

In this section, the experiments of the SL-policy and the RL-policy are conducted respectively on a workstation with 1 CPU of Intel Core i9-10900 K 3.70 GHz, 64 GB Ram, and OS is Windows 10 Pro.

6.1. SL-Policy

As couriers' previous decisions have good performance by anticipating demands and taking into account the traffic, the dataset in Table 1 is used to establish a basic policy to improve on later.

In this section, the policy is learned with the FourNN, and Ptr-Net proposed in Section 5.2, together with two optimizers Adam and SGD. The experiment is conducted in different settings, i.e., batch size being 32 or 64, and the learning rate varying from 0.0005 to 0.1. The experiment outcomes in terms of training accuracy and testing accuracy can be found in Table 2.

Note that, for both models, the larger the batch size, the result will converge with fewer steps, which also means less training time. The learning rate of Adam and SGD imposes opposite effects on the models: for Adam, training and testing accuracies significantly improve with the learning rate decreasing, while for SGD, both accuracies improve with the learning rate increasing. Therefore, the experiment is mainly conducted using 64 as batch size, and the highlighted results in terms of *Training Accuracy vs. Step* and *Mean Squared Error Loss vs. Step* are illustrated in Fig. 11 and Fig. 12.

From the testing accuracy perspective, the results show that the performance difference between Ptr-Net using Adam and learning rate 0.001, and learning rate 0.0005 is very insignificant, but both experiment settings obviously outperform the other settings. Furthermore, Ptr-Net using Adam and learning rate 0.001 is preferable regarding Mean Squared Error Loss (MSEL): the MSEL of this setting is the smallest compared with the other settings, and it converges and becomes stable in the end.

6.2. RL-Policy

To improve the policy with consideration of both delivery and pickup operations, the simulator is required to evaluate scenarios. The simulator is constructed based on three components, service time estimator, pickup pattern, and the SL-policy, which are all derived from the

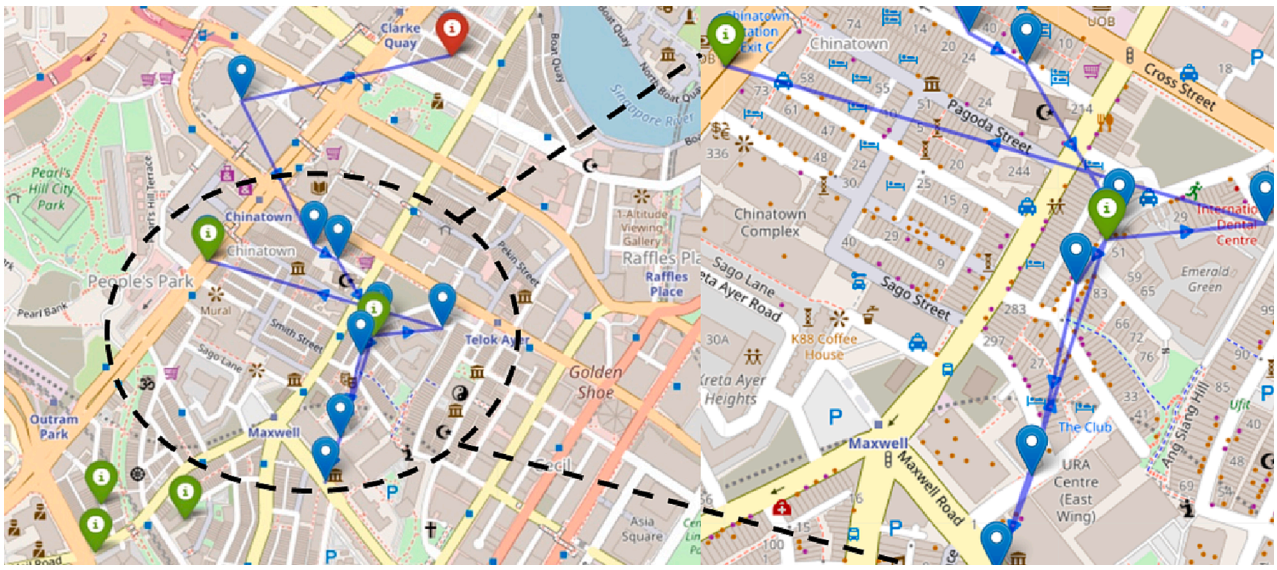


Fig. 15. Vehicle Routing based on the Nearest-Neighbor Policy, Case 1.

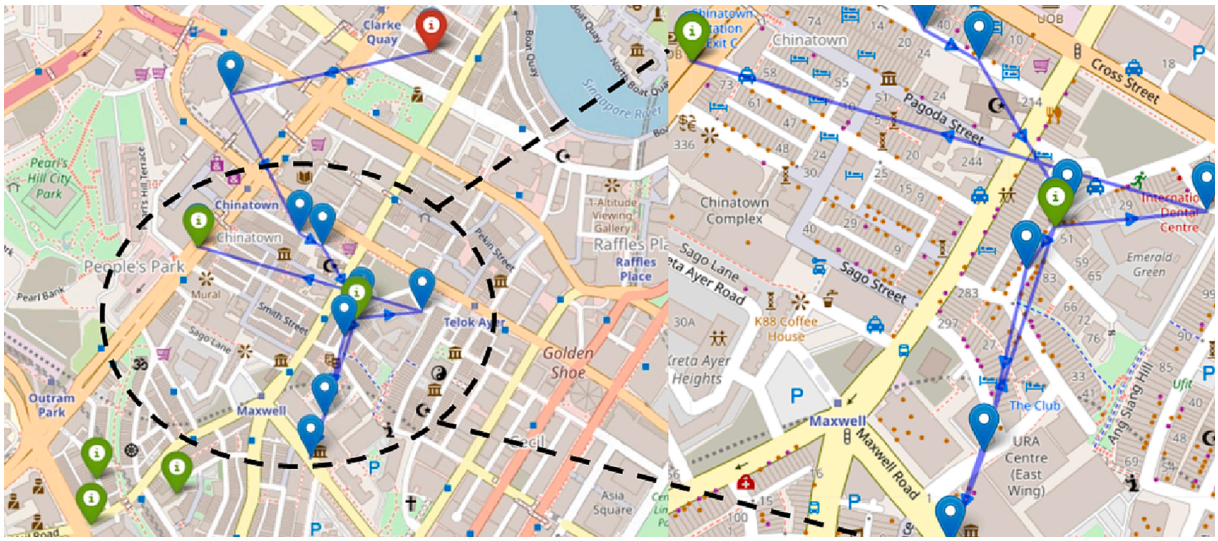


Fig. 16. Vehicle Routing based on the Dijkstra Policy, Case 1.

dataset in Table 1. The parameters of the RL's core function $policyimprovment(\pi_0, N, M_1, M_2)$ is set as follows: π_0 is the model trained based on the SL-policy using Ptr-Adam-64–0.001, $N = 10{,}000$, $M_1 = 25$ and $M_2 = 200{,}000$. In these settings, the ability of the RL-policy is tested to anticipate choices leading to more rewards.

To track the performance of the policy while the model is being trained, a test is conducted every 50 steps of training. In the beginning, 100 test scenarios are initialized with the same number of deliveries and pickups. The reward value is set to 10 for finishing a delivery request, and 5 for a pickup request. More discussions on reward setting can be found in Section 6.4. The locations of the deliveries are fixed, and only the pickups' locations and times of appearance vary. These scenarios are regularly simulated/tested with the policy currently being trained. At the end of each simulation, each test gives the number of customers (deliveries or pickups) the policy succeeded in serving. The average number of served jobs on all the test scenarios is the metric used to assess how well the policy performs.

As the benchmark, three policies are applied:

- **SL-policy:** The setting used for the SL-policy is Ptr-Adam-64–0.001.
- **Nearest-Neighbor policy:** This policy is designed to direct the courier going to the nearest available customer at each step (Cheng et al., 2017).
- **Dijkstra policy:** This policy is designed to calculate the shortest path among existing requests at each step. Initially, the Dijkstra algorithm calculates the shortest route among all delivery requests. while courier moving along the route, the policy will check if there is new request(s), specifically the pickup request(s), appearing. When new request(s) appears, the Dijkstra algorithm will be applied to update the route; otherwise, the courier will continue to serve on its current route.

The results on the average number of jobs served against step by using the RL-policy and three benchmark policies are shown in Fig. 13. Under the setting of Ptr-Adam-64–0.001, the maximal average numbers of served jobs using four policies are 9.43 (SL-policy), 15.13 (RL-policy), 13.50 (Nearest-Neighbor policy) and 13.96 (Dijkstra policy) jobs. If using the Nearest-Neighbor policy as the benchmark, the SL-policy is a 30.15% decrease and the RL-policy is a 12.07% increase. If using the Dijkstra policy as the benchmark, the SL-policy is a 32.45% decrease and

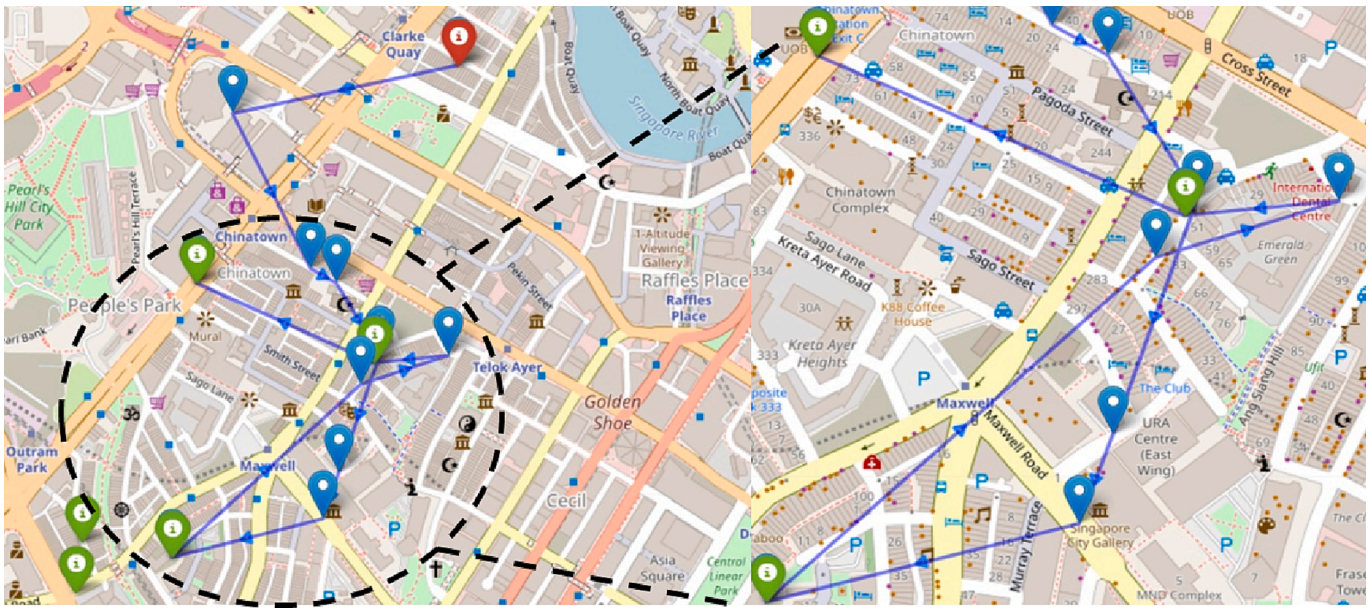


Fig. 17. Vehicle Routing based on the RL-Policy, Case 1.

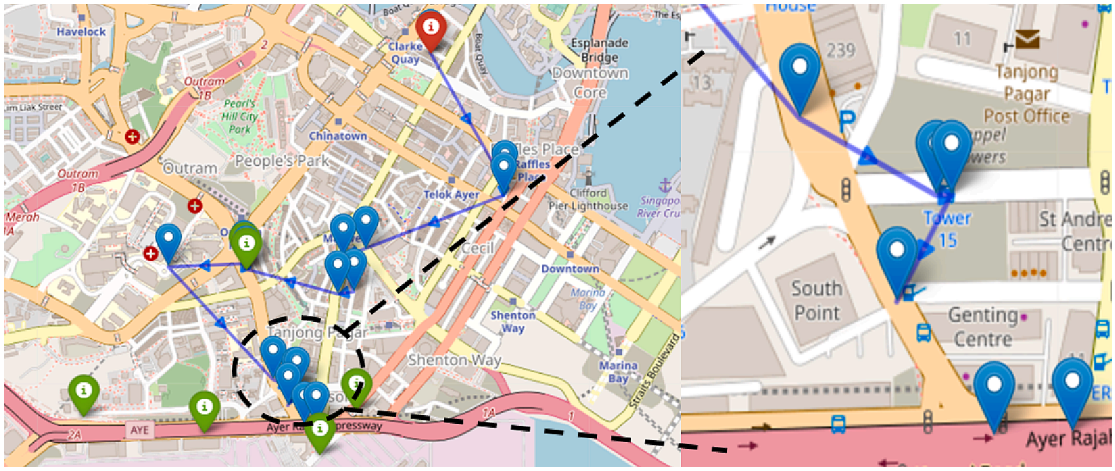


Fig. 18. Vehicle Routing based on the Nearest-Neighbor Policy, Case 2.

the RL-policy is an 8.38% increase. Given that there are more than 6,000 parcels delivered or picked up daily, a 12.07% or 8.38% increase will yield more than 724 jobs or 503 jobs completed daily.

To further examine the impact of the initial policy and training length, π_0 is selected from three settings, i.e., Ptr-Adam-64–0.001, Ptr-SGD-64–0.1 and FourNN-Adam-64–0.001 and $M_2 = 400{,}000$. The other parameters remain the same, i.e., $N = 10{,}000$ and $M_1 = 25$. The following experiments are conducted.

The subfigures of Fig. 14 show that the RL framework performs differently if the training is initialed from different base policies. The maximal average numbers of reserved jobs in three settings, i.e., Ptr-Adam-64–0.001, Ptr-SGD-64–0.1, and FourNN-Adam-64–0.001, are 15.14, 14.75, and 10.96, respectively. Obviously, Ptr-Adam-64–0.001 outperforms the other settings.

Note that, above experiments take approximately ten and half hours to finish. It is mainly due to the large value is set for M_2 for policy comparison purposes. If only Ptr-Adam-64–0.001 is adopted, compared with the maximal average numbers of reserved jobs achieved by $M_2 = 200{,}000$ and $M_2 = 400{,}000$, the values are 15.13 and 15.14, which can barely tell the difference. Hence, M_2 can choose a smaller value if computation time is limited.

6.3. Result analysis

To understand the RL-policy's mechanism, Fig. 15, Fig. 16 and Fig. 17 illustrate how courier travels in the real world. Using the Nearest-Neighbor policy, the courier will keep choosing the task which is closest to his current location. This is a rational choice for the courier: the courier has to finish all deliveries before noon, and he may choose to pick up some parcels along the way. However, there is no incentive for him to visit the places which are away from his delivery tasks. So, in Fig. 15, the courier keeps visiting the nearest task from his current location, and once the last pickup task is served, he goes off work. The Dijkstra policy performs the same route as the Nearest-Neighbor policy. It is mainly due to the limited number of requests can be served per courier given limited time. In many cases, the dynamic shortest route is also the nearest neighbors. The chance to have very difference routing is rare. As the RL-policy could anticipate future possible rewards while ensuring the time constraint is valid, the task which is not the closest one is dispatched in some cases in Fig. 16.

In Case 1, the courier has sufficient time to serve all customers and can serve another pickup request using the RL-policy. Another case is illustrated in Fig. 18, Fig. 19 and Fig. 20 for comparison. The courier is

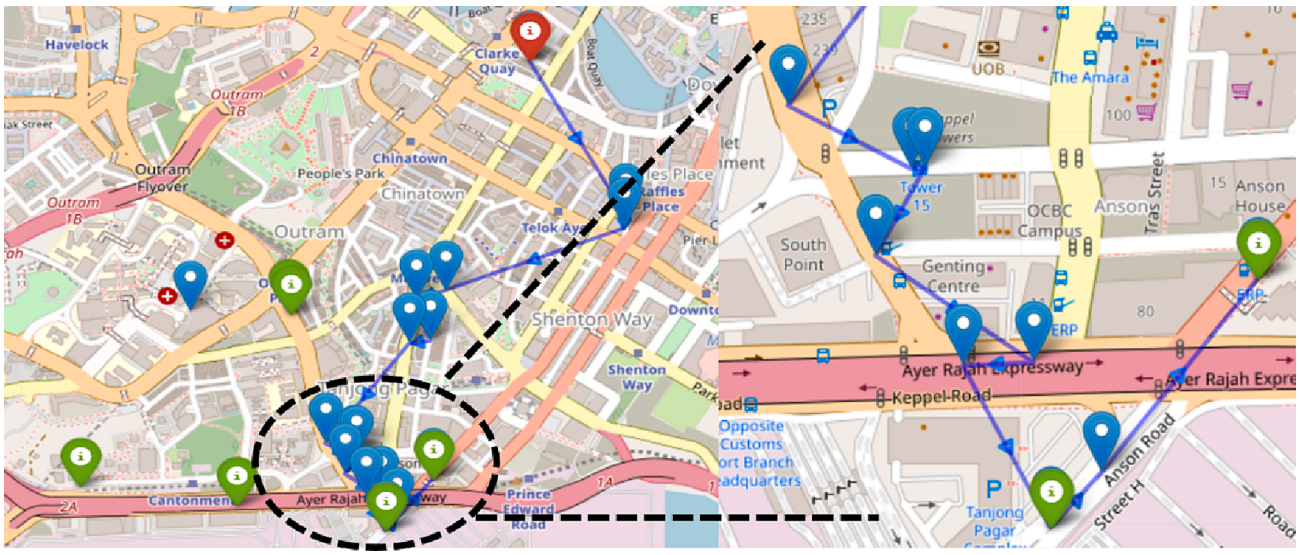


Fig. 19. Vehicle Routing based on the Dijkstra Policy, Case 2.

Fig. 20. Vehicle Routing based on the RL-Policy, Case 2.

facing a lot of requests, and he may not be able to serve all deliveries within the time limit, not to mention the pickup requests. If the courier relies on his instinct, i.e., serving requests based on the Nearest-Neighbor policy, he ends up with the last two deliveries unserved, as shown in Fig. 18. As the RL-policy can forecast future demands and maximize the reward at the same time, it can suggest the courier a better routing by serving more requests (in this case, one more delivery request). Despite slightly different on the route, the Dijkstra policy serves the same requests as the RL-policy.

6.4. Sensitivity analysis on reward values of the RL-Policy

It is well-known that the reward function has an influence on the convergence speed of reinforcement learning. To evaluate the impact, this section conducts a sensitivity analysis on the reward value of the RL-policy.

In this study, the priority of the delivery requests is higher than the pickup requests since the service commitment is important to the company. Thus, the reward values for evaluation are (1,2), (2,4), (3,6), (4,8), (5,10) and (6,12), where the first digit in the tuple is the reward of pickup requests, and the other digit is for delivery requests. Results presented in Fig. 21 show that (5,10) outperforms the other settings on convergence and stability of the outcomes. Thus choosing 5 for pickup and 10 for delivery is reasonable in Section 6.2.

6.5. Impact of request density

According to Section 4, the requests range from 1.264 to 1.304 in latitude, and from 103.815 to 103.860 in longitude, which is equivalent to an area of 5008 m times 4423 m. Given the space, the courier can perform 14.32 requests per day on average, and the average distance between closest neighbors 387.0 m. As an comparison, another set of data with lower density of requests is chosen, where the requests range from 1.283 to 1.348 in latitude, and from 103.753 to 103.812 in longitude. The new data set is spread out over an area of 5792 m times 8042 m. Due to larger area, the courier can perform 11.04 requests per day on average, and the average distance between closest neighbors 987.5 m.

Comparing with Fig. 13, Fig. 22 presents a similar trend, where the maximal average numbers of served jobs using four policies are 7.87 (SL-policy), 11.12 (RL-policy), 10.30 (Nearest-Neighbor policy) and 10.56 (Dijkstra policy). Both cases demonstrate that the RL-policy outperforms the other benchmark algorithms. However, when the request density is low, the improvement of the RL-policy is also limited, which is 8.0% if

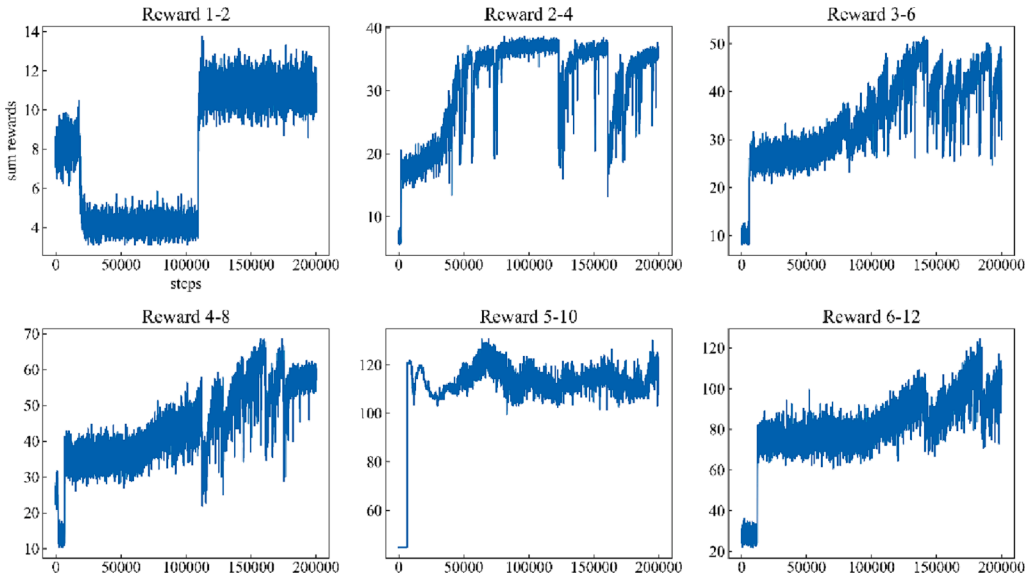


Fig. 21. Sensitivity Analysis on Reward Values.

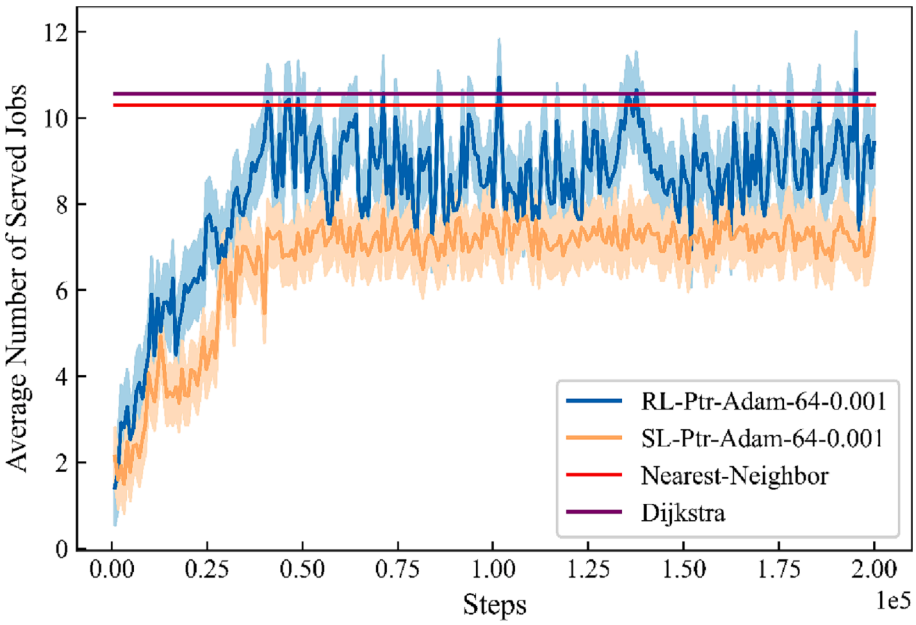


Fig. 22. Policy Comparison with Lower Density of Requests.

using the Nearest-Neighbor policy as the benchmark. The result makes sense as fewer total requests to be served, less improvement can be achieved.

6.6. *Management insight*

Instead of solely using vehicle traveling/navigation data, this study also emphasizes using historical operation data, which includes the situations like the courier making a temporary stop and then walking on foot. Although these data do not have specific remarks on whether the item was delivered on foot or by vehicle, the traveling time may imply the real situation. Adopting the neural network and then the RL framework makes it possible to train policies on deciding operation sequence with the consideration of courier's previous behaviors. Numerical study shows that (1) there exists an operation pattern for individual courier; (2) creating a realistic simulator is vital to the outcome, thus reliable data sources are needed; (3) the method has a good performance compared with the myopic policy which couriers usually adopt; and (4) with more requests to be served, the improvement obtained from the RL framework is higher.

Note that, to our best knowledge, current logistics provider will not advise the courier whether to walk or drive. Our method provides the courier a convenient suggestion on the operation sequence which has captured one's previous experience. In this case, the courier will likely follow the advice while further improving the performance.

7. Conclusion

This paper brought a new perspective to the treatment of last-mile delivery problems with uncertain demands. Specifically, it shows that RL methods from the Artificial Intelligence community can be used to build efficient policies. The profusion of data generated by couriers is an

opportunity to incorporate their knowledge into statistical models. We succeeded in reproducing decisions made by drivers in up to 50% of the test cases, using a recently developed technique, Pointer Network. The policy trained from the RL framework could offer approximately 12% improvement in high density area and 8 % in low density area. This work shows that machine learning is a useful and promising tool to derive empirical models, and offers new research areas for conventional problems, and this work should be seen as an introduction towards a more complex use of RL techniques in DVRPSD. Significant work and improvements remain to be done, and we give some directions hereafter. An emphasis should be made on the practical implementation of the framework. To allow for the use of complex models in an industrial setting, it would be interesting to exploit Graphics Processing Units (GPUs) or distributed computing implementations. Our approach is well suited for highly parallel implementations. This could bring the strength of recent RL techniques to a fast-growing industry that needs to improve its efficiency. Furthermore, since the proposed approaches only make use of a standard reinforcement learning framework, the other advanced techniques, such as Deep Q-Network and Actor-critic, can be applied to resolve the problem.

CRediT authorship contribution statement

Chenhao Zhou: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition. **Jingxin Ma:** Software, Formal analysis, Methodology, Investigation, Validation, Writing – original draft, Writing – review & editing. **Louis Douge:** Software, Methodology, Investigation, Writing – original draft. **Ek Peng Chew:** Methodology, Funding acquisition. **Loo Hay Lee:** Methodology, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China [72101203], Shaanxi Provincial Key R&D Program, China [2022KW-02], and The Youth Innovation Team of Shaanxi Universities, China.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cie.2023.109443.

References

Bai, R., Chen, X., Chen, Z.-L., Cui, T., Gong, S., He, W., … Kendall, G. (2023). Analytics and machine learning in vehicle routing research. *International Journal of Production Research, 61*(1), 4–30.

Basso, R., Kulcsár, B., Sanchez-Diaz, I., & Qu, X. (2022). Dynamic stochastic electric vehicle routing with safe reinforcement learning. *Transportation research part E: logistics and transportation review, 157*, Article 102496.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Springer.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering, 99*, 300–313.

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

Cheng, X., Liao, S., & Hua, Z. (2017). A policy of picking up parcels for express courier service in dynamic environments. *International Journal of Production Research, 55*(9), 2470–2488.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management science, 6*(1), 80–91.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Hildebrandt, F. D., Thomas, B. W., & Ulmer, M. W. (2022). Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Computers & Operations Research, 106071*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735–1780.

Hooshmand Khaligh, F., & MirHassani, S. (2016). A mathematical model for vehicle routing problem under endogenous uncertainty. *International Journal of Production Research, 54*(2), 579–590.

IMARC Group. (2022). *Courier, Express and Parcel (CEP) Market: Global Industry Trends, Share, Size, Growth, Opportunity and Forecast 2022-2027*. https://www.researchandmarkets.com/reports/5547114/courier-express-and-parcel-cep-market-global.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., … Meng, Z. (2021). A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in Neural Information Processing Systems, 34*, 23609–23620.

Moghdani, R., Salimifard, K., Demir, E., & Benyettou, A. (2021). The green vehicle routing problem: A systematic literature review. *Journal of Cleaner Production, 279*, Article 123691.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1–11.

Psaraftis, H. N., Wen, M., & Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks, 67*(1), 3–31.

Rios, B. H. O., Xavier, E. C., Miyazawa, F. K., Amorim, P., Curcio, E., & Santos, M. J. (2021). Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering, 160*, Article 107604.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., … Lanctot, M. (2016). Mastering the game of Go with deep neural networks and tree search. *nature, 529*(7587), 484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Graepel, T. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144.

Soeffker, N., Ulmer, M. W., & Mattfeld, D. C. (2022). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research, 298*(3), 801–820.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., & Hennig, M. (2019). Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science, 53*(1), 185–202.

Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., & Thomas, B. W. (2020). On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics, 9*(2), Article 100008.

Ulmer, M. W., & Thomas, B. W. (2020). Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. *European Journal of Operational Research, 285*(1), 183–195.

Vidal, T., Laporte, G., & Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research, 286*(2), 401–416.

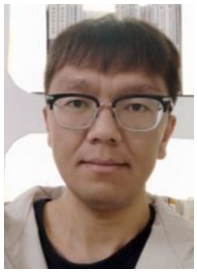
Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *arXiv preprint arXiv: 1506.03134*.

Wang, Z., & Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation research part B: methodological, 122*, 350–364.

Zhang, J., Luo, K., Florio, A. M., & Van Woensel, T. (2023). Solving large-scale dynamic vehicle routing problems with stochastic requests. *European Journal of Operational Research, 306*(2), 596–614.

Zhao, Q., Zhou, C., & Pedrielli, G. (2020). A Decision Support System for Data-Driven Driver-Experience Augmented Vehicle Routing Problem. *Asia-Pacific Journal of Operational Research, 37*(05), 2050018.

Dr Chenhao Zhou, is a Professor from the School of Management at Northwestern Polytechnical University. Prior to this, he was a Research Assistant Professor in the Department of Industrial Systems Engineering and Management, National University of Singapore. His research interests are transportation and logistics systems using simulation and optimization methods.



Mr Jingxin Ma, graduated from the School of Management at Northwestern Polytechnical University with Bachelor degree in 2022. He is currently a software engineer at Industrial and Commercial Bank of China.

Mr Louis Douge, graduated from National University of Singapore with Master of Engineering in 2018. He is currently a Data Scientist with AXA Singapore, Data Innovation Lab.

Dr Ek Peng Chew, is a Professor of the Department of Industrial Systems Engineering and Management in National University of Singapore.

Dr Loo Hay Lee, is a Professor of the Department of Industrial Systems Engineering and Management in National University of Singapore.