DECISION PREFERENCE ALIGNMENT FOR LARGE-SCALE AGENTS BASED ON REWARD MODEL GENERATION

Jiaoling Zheng, Weifeng Xu, Guokang Gao, Yulin Ren, Xingyu Fan
Chengdu University of Information Technology
Chengdu , China

Qian Luo*, Wanli Dang, Long Geng
The Second Research Institute of CAAC
Chengdu , China

ABSTRACT

This paper presents a novel data generation method for large scale agents' decision preference alignment. Despite the recent increasing attention to AI alignment, machine learning approaches for AI alignment are still challenging due to the lack of data. Trajectory data representing agent behavior is essential in various alignment methods such as Reinforcement Learning from Human Feedback(RLHF) or Inverse Reinforcement Learning(IRL). In this paper, we significantly reduces the dependence on trajectory data. Our method uses a generative method to shift the focus from learning about the reward model for alignment to learning how to generate sample data for alignment. Therefore, our method broadens the scope of data needed for alignment to include both microscopic and macroscopic information that can be obtained. By designing detailed macro and micro metrics, it is verified that the simulation results of passenger boarding process based on generated decision preferences match well with those guided by ground truth decision preferences.

1 INTRODUCTION

One obstacle to applying reinforcement learning algorithms to real-world problems is the lack of suitable reward functions. Designing such reward functions is difficult in part because the user only has an implicit understanding of the task objective Ji et al. (2023) Leike et al. (2018). This gives rise to the agent alignment problem: how do we create agents that behave in accordance with the real world agents' intentions?

Consider the following two scenarios:

Scenario 1: the airport is in a normal situation, the check-in area and the security check area are relatively smooth.

Scenario 2: Due to equipment failure, the airport temporarily closes half of the check-in counters and half of the security channels.

The problem is can we predict the flow density of each area in the terminal building in the case of scenario 2?

Passenger flow prediction methods are not suitable for this problem since the scenario has changed and historical data cannot be applied to the new scenario. If using multi-agent reinforcement learning to solve this problem, the most critical issue is to know each passenger's reward function or decision making preferences. For example, passenger A's boarding preference is to avoid waiting in line. Passenger B's boarding preference is to reach the boarding gate as soon as possible. If the decision

*Corresponding author email: caacsri_luoqian@163.com

making preference of each passenger's boarding process can be inferred, the passenger's optimal boarding route can be calculated. Although the passenger's optimal boarding route may be different in various scenarios, his/her decision preference remains the same. After obtaining The passengers' boarding routes, the flow density in each area of the terminal building can be predicted through simulation.

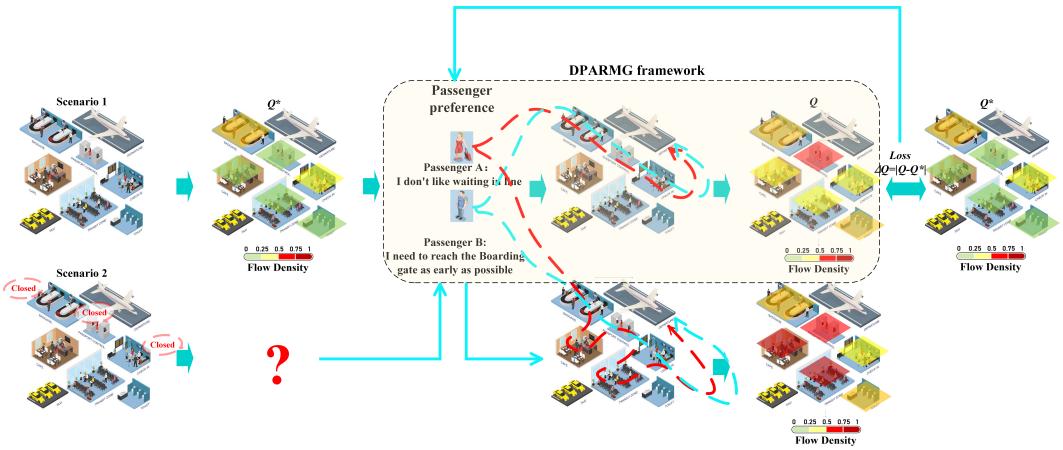

Figure 1: Framework of Decision Preference Alignment Based on Reward Model Generation

How can we obtain each passenger's decision preference?

The DeepMind team in the literature Leike et al. (2018) indicated a research direction for solving the problem of aligning decision preferences of agents by implementing reward modeling. We can use the reward model to represent passenger's decision preference.

So, in this paper, we will construct a reward model that can reflect the passenger's real decision making preference. Assume there are two reward models R^* and R. $R^* = \{r_1^*, r_2^*, \ldots, r_n^*\}$, where r_i^* is the reward model that can represent the true preference of each passenger. $R = \{r_1, r_2, \ldots, r_n\}$, where r_i is the inferred reward model of each passenger. "n" is the number of passengers.

The goal of this paper is to obtain R that aligning with R^* under the conditions that R^* is unknown and no passenger trajectory data is available.

Figure 1 illustrates the framework for solving the problem. In scenario 1, the data we can obtain is the flow density Q^* of each area in the terminal building through cameras. We initialize R. Then we use multi-agent reinforcement learning to find the optimal trajectory for passenger i based on r_i. The flow density Q in each area of the terminal building in the case of scenario 1 can be predicted through simulation. We optimize R by minimizing the difference between Q and Q^*. From a statistical perspective, if Q can approximate Q^*, we believe that R aligns with R^*.

2 Related Work

Three machine learning methods for performing reward function construction are described below.

Behavioral Cloning (BC). This method learns agent actions directly Hussein et al. (2017) Zare et al. (2024), without learning reward functions, but requires a large amount of expert trajectory data. Behavioral cloning directly splits the expert trajectory data into state-policy pairs to express the corresponding expert action under the current state. A supervised learning method is used to train the model by taking states as the input and actions as the output. Behavioral cloning has played a role in many practical applications. Yildirim et al. (2024) demonstrated the low error rates and application potential of behavior cloning-based perception systems for autonomous driving in real-world scenarios. Mani et al. (2024) developed the DiffClone algorithm, which enhances behavior cloning through diffusion policy learning. Scaramuzza & Kaufmann (2023) utilized deep sensor policies to achieve agile simulation-to-reality flight for drones and explored open challenges in improving agility and robustness.

Behavioral cloning simply mimics the expert's behavioral strategy. When the scenario changes, a more effective approach is to learn the reward function in the decision-making process. There are two methods to obtain the reward model, namely Inverse Reinforcement Learning (IRL) and Reinforcement Learning Based on User Feedback (RLHF).

Inverse Reinforcement Learning investigates the problem of learning an agent's preferences from observed behaviors, which are usually represented as reward functions. Unlike just mimicking observed behaviors, learning the reward functions behind them can lead to better robustness, better generalization to different environments, and the ability to combine rewards extracted from multiple behaviors. However, inverse reinforcement learning also requires expert trajectory data to infer the reward function, Wu et al. (2023) proposed an inverse reinforcement learning-based human-like obstacle avoidance trajectory planning strategy, enhancing the adaptability and realism of autonomous driving trajectories. Ye et al. (2023) developed InitLight, an adversarial inverse reinforcement learning method, which improves multi-intersection traffic signal control by pre-trained models with expert trajectories. Qiao et al. (2023) introduced a multimodal inverse-constrained reinforcement learning algorithm, utilizing traffic density estimators to simultaneously identify multiple expert constraints and optimize policy imitation.

RLHF based reward learning. In order to make the output of the model more consistent with human values or expectations, the RLHF method is often used to introduce human feedback to optimize the reward function. First, the method also requires trajectory data, but this trajectory data is generated by the user. Secondly, the loss function is constructed from the labeled preference trajectory data, and the supervised learning method is used to optimize the reward model and finally achieve goal alignment. Casper et al. (2023) explores open challenges and limitations of reinforcement learning from human feedback (RLHF) and recommends strengthening social oversight through auditing and disclosure standards. Wang et al. (2024) leverage RLHF to enhance lane-changing behavior of autonomous vehicles in mixed traffic, making it more human-like. Wang et al. (2023) proposes a hierarchical reinforcement learning framework to identify root causes of anomalies in microservice systems, improving analysis efficiency. Hwang et al. (2023) introduces the SeqRank framework, boosting feedback efficiency through sequence preference ranking.

The three methods need trajectory data. Since it's hard to obtain, this paper focuses on solving the reward function without such data.

3 PROBLEM DEFINITION

The problem of obtaining R that aligning with R^* consists of three subproblems. Firstly, how to accurately define R. Due to the large number of passengers in the terminal, it is impossible to obtain each traveler's reward model. Secondly, how to quantify the degree of R's alignment to R^*. Since it is not possible to obtain the true R^*, we can not directly compare R^* and R. Finally, how to formalize the alignment of R to R^* into an optimization problem. 3.1$\sim$3.3 described the three subproblems.

3.1 REWARD MODEL FOR LARGE-SCALE PASSENGERS

How to accurately define R? The terminal has a large number of passengers, and it is not possible to measure whether each r_i^* approximates each r_i. This paper transfers the reward model of all passengers into one reward model. The decision preferences of different passengers is represented by the probability distributions.

Definition 1: The reward model R for all passengers. $R(s,a) \sim N(\mu(s,a), \sigma^2(s,a))$, is represented by a Gaussian distribution, where $\mu(s,a)$ is the expected reward under the state action pair and $\sigma^2(s,a)$ is the variance of the reward.

3.2 MACRO LEVEL METRIC TO MEASURE THE ALIGNMENT OF R WITH R^*

How to quantify the degree of R's alignment with R^*? Due to the significant difficulty of quantitatively verifying alignment at the micro level, this paper will verify them at the macro level. This section defines the macro level metric for measuring the alignment of R with R^*. From a statistical

perspective, if Q can approximate Q^*, we believe that R aligns with R^*. Furthermore, Q and Q^* can be other easily obtained macro level metrics.

Definition 2: The actual passenger flow Q^* in different areas of the terminal. $Q^* = \begin{pmatrix} Q*_{1,1} & ... & Q*_{1,T} \\ ... & ... & ... \\ Q*_{n,1} & ... & Q*_{n,T} \end{pmatrix}$, $Q^*_{i,t}$ represents the actual passenger flow in area i of the terminal at time t. T is the total time steps, i is the area i of the terminal, and n is the total number of areas.

Definition 3: The simulated passenger flow Q in different areas of the terminal. $Q = \begin{pmatrix} Q_{1,1} & ... & Q_{1,T} \\ ... & ... & ... \\ Q_{n,1} & ... & Q_{n,T} \end{pmatrix}$, $Q_{i,t}$ represents the simulated passenger flow in the ith area of the terminal at time t, T is the total time step, i is the i^{th} area of the terminal, and n is the total number of areas.

This paper uses $\Delta Q = ||Q - Q^*||_F = \sqrt{\sum_{i,j}^{1 \le i \le n, 1 \le j \le T} (Q_{i,j} - Q*_{i,j})^2}$ to measure the degree to which R approaches R^*, with smaller ΔQ indicating that R is closer to R^*.

3.3 Transfer the alignment problem into data generation problem

How to formalize the alignment of R with R^* into an optimization problem?

Definition 4. Sample data for training the reward function. Data is a set of decision-making data during the boarding process of passengers, including states, actions, and the rewards obtained after taking actions. $Data = \{(s_1, a_1, r_1), (s_2, a_2, r_2), \ldots, (s_m, a_m, r_m)\}$. Among them, s_i and a_i are a set of pre-sampled states and actions during the boarding process of passengers.

If the passenger boarding simulation process f is considered a black box process, then the reward model R defined in Definition 1 can be seen as the input to f, and the macro metric ΔQ obtained from the simulation process can be seen as the output of f, $\Delta Q = f(R)$. Under different reward functions R, different ΔQ can be obtained. But black-box optimization can only optimize data, not functions. Since R can be obtained by conducting supervised learning on $Data$. We can use $Data$ as the training data to learn the reward model R. The ultimate goal of this paper shifts from learning the reward model to learning the parameterized model θ that can generate sample data for reward model training, as shown in Equation (1).

$$\theta = \underset{\theta}{\text{Argmin}}(\Delta Q), \Delta Q = f(Data), \tag{1}$$

4 Contributions of this paper

The DPARMG framework has the following advantages:

First, our method significantly reduces the dependence on trajectory data which is necessary within the IRL and the RLHF framework. Trajectory data is not necessary any more in our framework.

Secondly, the framework uses a generative method to shift the focus from learning about the reward model for alignment to learning how to generate sample data for alignment. Our method broadens the scope of data needed for alignment to include both macro-level and micro-level information that can be obtained.

Finally, the framework extends the use of reinforcement learning by creating a connection between micro-level decision preferences and macro-level phenomena. This framework allows reinforcement learning to be applied not only for reward maximization but also to guide and control agent behavior based on macro-level observations.

5 TRAINING THE PASSENGER BOARDING PROCESS DECISION MODEL

This paper adopts distributed training and distributed execution multi-agent reinforcement learning to model the boarding process of a large number of passengers. Each passenger learns his/her decision model independently. For each passenger, we use MAXQ hierarchical reinforcement Dietterich (2000) learning to simplify the learning process. The boarding process is divided into three sub-tasks. Figure 2 shows the MAXQ graph of the passenger's boarding process.

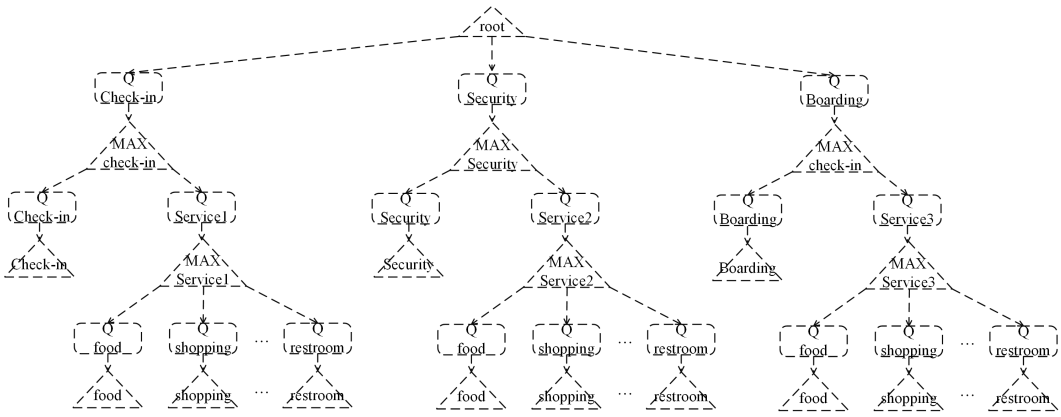

Figure 2: MAXQ diagram of passenger boarding MDP process

Equation (2) is the Bellman equation corresponding to the MAXQ graph. Each basic action and each sub-task (including the root task) has a Max node to store the value function associated with that node. The Q node corresponds to the actions that can be executed for each sub-task. The Q node stores the expected reward after executing the action in a specific state. $C(i, s, a)$ represents the expected discounted cumulative reward of completing sub-task i after performing sub-task a in state s. The greedy action a^* refers to the optimal action chosen in state $s^{'}$. $C(i, s, a)$ is updated using the value of the greedy action a^* in the resulting state $s^{'}$. $V(i, s)$ represents the expected reward of completing sub-task i in state s.

$$
\begin{aligned}
V(i,s) &= \begin{cases} Q(i,s,a) & \text{if } i \text{ is composite,} \\ \alpha \cdot r + (1-\alpha) \cdot V(i,s) & \text{if } i \text{ is primitive.} \end{cases} \\
Q(i,s,a) &= V(a,s) + C(i,s,a) \\
C(i,s,a) &= \alpha \cdot \gamma^N [C(i,s',a^*) + V(a^*,s')] + (1-\alpha) \cdot C(i,s,a)
\end{aligned} \tag{2}
$$

Assume that the passenger has just entered the terminal. Let s_1 represent his/her current state. According to Figure 2, the passenger is in the $QCheck\text{-}in$ sub-task. Equation (3) shows how to calculate the value of $V(MaxCheck\text{-}in, s_1)$

$$
\begin{aligned}
V(MAXCheck\text{-}in, s_1) &= Q(MAXCheck\text{-}in, s_1, MAXService) \\
&= V(QService, s_1) + C(MaxService, s_1, QService) \\
&= Q(MAXService, s_1, s_2, Qfood) + C(MaxService, s_1, QService) \\
&= V(Qfood, s_1, s_2) + C(MAXService, s_1, s_2, Qfood) + \\
&C(MAXService, s_1, QService) \\
&= \alpha \cdot r + (1-\alpha) \cdot V(Qfood, s_1, s_2) + C(MAXService, \\
&s_1, s_2, Qfood) + C(MAXService, s_1, QService)
\end{aligned} \tag{3}
$$

According to Figure 2, the passenger boarding task contains three sub-tasks, i.e., check-in, security check, and boarding. Each sub-task contains primitive and composite states. A detailed description of the state is given in the appendix Table 3.

Each triangle in Figure 2 represents the action taken by the passenger under the current sub-task. The actions for composite tasks is an abstract expression of actions. Those actions can be seen as

the necessary procedural facilities that the passengers can choose to go to under a certain sub-task. The actions for primitive tasks include atomic actions such as check-in, dining, shopping, restroom, security check, and waiting for boarding.

The reward of action for primitive task is calculated according to the reward function, $reward = R(s, a)$, given in definition 1. The reward of action for composite task is recursively calculated based on Equation (2). Chapter 6 describes how to calculate $R(s, a)$.

6 Generating Sample Data for Fitting Reward Model Based on conditional Diffusion Model

Since we can use $Data$ as the training data to learn the reward model $R(s, a)$, the ultimate goal of this section is to shifting from learning to learning the parameterized model θ that can generate $Data$, as shown in Equation (1). To minimize ΔQ in equation (1), we adopt the conditional diffusion model p_θ to generate $Data$, defined in Definition 4, for fitting the reward model $R(s, a)$. $p_\theta : \Re^{\mathrm{d}_{\Delta Q}} \times \Re^{\mathrm{d}_{Data}} \to \Re^{\mathrm{d}_{Data}}$ represents the conditional diffusion model which learns how to transform a vector of white noise $\varepsilon \in \Re^{\mathrm{d}_{Data}}$ into $Data$. The conditional diffusion model is parameterized by trainable parameters θ.

$$Data = p_\theta(\varepsilon, \Delta Q) \tag{4}$$

Figure 3 illustrates the format of $Data$. If the states take numerical values, the data volume can be quite large. To reduce the amount of data in $Data$, this paper discretizes the states. $freetime$ is discretized into three types, sufficient, moderate and limited. Queue length is discretized into three types, long, medium, and short. After discretization, the table length in Figure 3 is the Cartesian products cardinality of states and actions. The last two columns in the table are the mean μ and variance σ^2 of the reward model defined in Definition 1. Since the Cartesian product of state and action data is fixed, the diffusion model only needs to generate μ and σ^2. However, the dimensionality of the data containing only μ and σ^2 is still very high. So, this paper adopt the Latent Diffusion Model(LDM) framework.

The core idea behind LDM is to first map the high-dimensional data into a lower-dimensional latent space using an encoder. Once in the latent space, a diffusion process is applied, and a generative model learns to reverse the diffusion to generate data samples. This approach significantly reduces the computational burden because the diffusion process is performed in the smaller latent space rather than the original high-dimensional pixel space. Figure 3 illustrates the denoising framework of the latent diffusion model.

The encoder maps $Data$ to a latent space representation z:$z = Encoder(Data)$

The forward diffusion process adds noise to the latent representation z over T time steps. At each time step t, the latent representation z_t is computed as: $\mathrm{z}_t = \sqrt{\alpha_t} z_0 + \sqrt{(1 - \alpha_t)}\epsilon$. α_t is a schedule of diffusion coefficients. ϵ is noise sampled from a standard normal distribution.

The reverse diffusion process aims to denoise the latent representations. Given a noisy latent sample z_t, the denoising model D predicts the noise ϵ:$\hat{\epsilon} = D(\mathrm{z}_t, t)$.The predicted noise is then used to update the latent representation to progressively denoise it.

Once the latent representation has been denoised, the decoder maps it back to the original data space: $Data = Decoder(z)$.

The denoising process involves minimizing the loss function that measures the difference between the predicted noise and the actual noise added during the diffusion process:$Loss = E_{Data,\epsilon,t}[||\epsilon - D(z_t, t)||^2]$.

The above describes the unconditional diffusion process. We modify the reverse denoising process by incorporating the ΔQ as embedded conditions. The final loss function of denoising process is as follows:$Loss = E_{Data,\epsilon,t}[||\epsilon - D(concat(z_t, t, \Delta Q))||^2]$.

The overall framework of "Decision Preference Alignment Based on Reward Model Generation(DPARMG)" is illustrated by algorithm 1. $Data_k$ in the initial dataset is a set of random values for μ and σ^2 in Figure 3. $\Delta Q_k = |Q_k^* - Q_k|$ in the initial dataset is obtained by con-

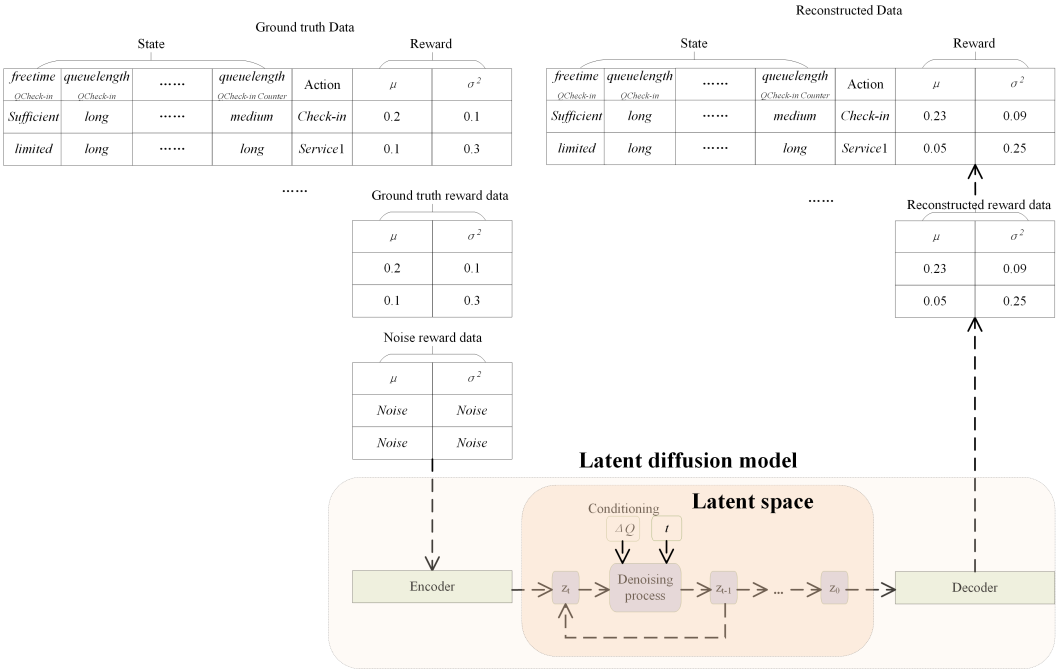

Figure 3: Generating Sample Data for Fitting Reward Model Based on Conditional Diffusion Model

ducting MAXQ with reward model according to $Data_k$. Conditioning on $\{\Delta Q_j^{'}\}_{j=1}^{N}$, we generate N samples $\{Data_j\}_{j=1}^{N}$, $Data_j \sim p_\theta(Data_j|\Delta Q^{'}, D)$. By conducting MAXQ with reward model according to $\{Data_j\}_{j=1}^{N}$, we obtain the $\{\Delta Q_j\}_{j=1}^{N}$. The we append all queried data pairs $\{Data_j, \Delta Q_j\}_{j=1}^{N}$ to the training dataset D. To minimize ΔQ in equation (1), algorithm 1 outputs $Data_k$ that has the minimum ΔQ_k value in the $\{Data_j, \Delta Q_j\}_{j=1}^{N}$ pairs.

Algorithm 1 Framework of Decision Preference Alignment based on Reward Model Generation(DPARMG)

1: Input: Initial dataset $D = \{Data_k, \Delta Q_k\}_{k=1}^{M}$
2: Initialization: Conditional latent diffusion model $p_\theta(Data|\Delta Q)$
3: **for** $k = 1, 2, \ldots, K$ **do**
4: Train the conditional diffusion model with D
5: Randomly sample $\{\Delta Q_j^{'}\}_{j=1}^{N}$, $0 < \Delta Q_j^{'} \leq \alpha$, α is a small positive number
6: Generate $\{Data_j\}_{j=1}^{N}$ where $Data_j \sim p_\theta(Data_j|\Delta Q_j^{'}, D)$
7: Obtain $\{R_j\}_{j=1}^{N}$ based on $\{Data_j\}_{j=1}^{N}$
8: Obtain $\{\Delta Q_j\}_{j=1}^{N}$ based on conducting MAXQ with reward model $\{R_j\}_{j=1}^{N}$
9: $D \leftarrow D \cup \{Data_j, \Delta Q_j\}_{j=1}^{N}$
10: **end for**
11: Output: $Data_k$, s.t. ΔQ_k is the minimal value in D

7 EXPERIMENTS

7.1 EXPERIMENTAL DESIGN

This paper is based on Chengdu Shuangliu airport terminal. The entire terminal is divided into six major areas (Q_1 to Q_6), and each major area is further subdivided into several smaller areas (Q_7 to Q_{21}), totaling 21 areas. The simulation process will obtain the passenger flow density for each

area, i.e., $Q = \{Q_1, Q_2, \ldots, Q_{21}\}$. The map of the airport terminal and the corresponding areas are provided in appendix Figure 5.

We design three different scenarios. Scenario 1 is the baseline scenario. In this scenario, the flow density data Q^* is obtained. In scenario 2, the check-in counter Q_{11} is closed. Passengers who originally checked in at the Q_{11} area will proceed to the Q_{12} area check-in. In scenario 3, security check channels Q_{15} and Q_{16} are closed.

We design the passengers' real decision-making preference by reward model R^* according to Jiang et al. (2023). $R^*(s,a) = w_1 * freetime + w_2 * queuelength + w_3 * action$, where $freetime$, $queuelength$ and $action$ denote the value of the specific state and action. $w_1 \sim N(0.4, 0.1)$, $w_2 \sim N(0.3, 0.1)$, $w_3 \sim N(0.7, 0.1)$. $freetime$ and $queuelength$ are consistent with the definition in section 5.

We designed one baseline algorithm, M, and four comparative algorithms, $M_1 \sim M_4$, as shown in Table 1. Since there is currently no algorithms for learning micro-level reward models based on macro-level metrics, all comparative algorithms are designed based on the DPARMG framework. Each algorithm includes a reward model learning module and a multi-agent reinforcement learning module. Among them, algorithm M is used to generate the benchmark data Q^*, which can be regarded as representing the decision-making process reflecting the true preferences of passengers during boarding. So M is not included in the comparison. M_1 uses maximum entropy inverse reinforcement learning the obtain the reward model. The trajectory data is obtained by conducting M in the simulation environment. M_2 changes the MAXQ method in the DPARMG framework to a tabular Q-learning algorithm, maintaining a Q-table for each passenger. M_3 removes the encoder from the diffusion model. M_4 is the complete DPARMG framework.

Table 1: Comparative methods

	Reward model generation method	**Reinforcement learning algorithm**
M	R^*	MAXQ
M_1	IRL	MAXQ
M_2	LDM	Tabular Q-learning
M_3	LDM without encoder	MAXQ
M_4	LDM	MAXQ

7.2 PURPOSE OF THE EXPERIMENT

This section illustrates the purpose of the experiment:

RQ1: Can the DPARMG framework achieve preference alignment of the passengers' boarding process in scenario 1. The purpose of this experiment is to verify the alignment of R with R^*. We conduct both the reward model learning and multi-agent reinforcement learning according to the DPARMG framework in scenario 1 to obtain the reward model R. We use ΔQ=$|Q - Q^*|$ to measure the alignment of R with R^*. Q^* is obtained according to M. Q is obtained according to $M_1 \sim M_4$.

RQ2: Based on the reward model obtained in scenario 1, can the DPARMG framework be capable of predicting flow density in scenario 2 and 3. The purpose of this experiment is also to verify the alignment of R with R^*. But in this experiment, we only conduct multi-agent reinforcement learning in scenario 2 and 3 with the reward model R obtained in scenario 1.

RQ3: Can the micro level trajectory data obtained through R be consistent with the ground truth trajectory data obtained through R^*, even though we are aligning them based on the macro level metric of flow density ΔQ? The purpose of this experiment is also to verify the alignment of R with R^*. But in this experiment, we use the trajectory data to measure the degree of alignment. The metric is explained in detail in section 7.4.

Table 2: Performance of $M_1 \sim M_4$ for minimizing ΔQ

	$Scenario1$	$Scenario2$	$Scenario3$
M_1	14.39	3.79	32.38
M_2	15.16	5.07	36.49
M_3	25.74	5.89	60.26
M_4	5.29	2.3	32.2

7.3 THE THE ALIGNMENT DEGREE R AND R^* BASED ON THE MACRO LEVEL METRIC ΔQ(RQ1, RQ2)

This subsection evaluates whether the reward function R obtained by the algorithms can minimize ΔQ. ΔQ obtained by the four algorithms for the three scenarios is given in Table 2. M_4 has the best performance. M_3 has the worst performance indicating that the encoder is important for diffusion model when dealing with high dimensional data. Details are provided in Appendix Figure 6∼8.

7.4 THE THE ALIGNMENT DEGREE R AND R^* BASED ON THE MICRO LEVEL METRIC $Accuracy_{Traj}$(RQ3)

We divide the entire simulation process into 10 equal-width time intervals. For any passenger A, in the $i^{th}(1 \leq i \leq k)$ time interval, we use $Q^{*A}_{i,j}$ to denote that A is in the area Q_j under the guidance of R^*. we use $Q^{A}_{i,j}$ to denote that A is in the area Q_j under the guidance of R. We design metric T_i and F_i, T_i = number of passengers, if $j = k$ for $Q^{*A}_{i,j}$ and $Q^{A}_{i,j}$ of all passengers in the $i^{th}(1 \leq i \leq k)$ time interval ; F_i = number of passengers, otherwise. The variable i in equation (5) is set to the values of 1, 2, 5, and 10.

$$Accuracy_{Traj} = \frac{T_i}{T_i + F_i} \tag{5}$$

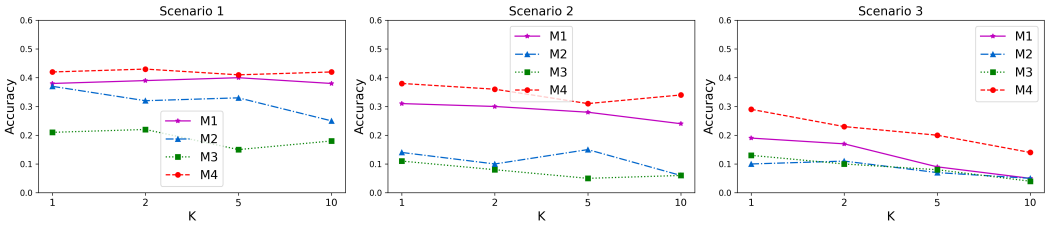

Figure 4: Accuracy of different algorithms in three scenarios

As we observed in Figure 4, M_4 performed the best. M_3 performed the worst in all scenarios, indicating again that diffusion model without encoder has low effectiveness. The performance of M_1 was closest to M_4, but M_4's advantage was still very obvious.

8 CONCLUSION

This paper proposes a framework for aligning decision process preferences based on a reward model generation algorithm. The framework firstly designs macro level quantitative metrics for reward alignment. Secondly, the framework uses a generative method to shift the focus from learning about the reward model to learning from sample reward data, thereby making the learning process easier. Finally, the framework extends the use of reinforcement learning by creating a connection between micro-level decision preferences and macro-level phenomena. This framework allows reinforcement learning to be applied not only for reward maximization but also to guide and control agent behavior based on macro-level observations.

ACKNOWLEDGMENTS

This study was funded by the Natural Science Foundation of Sichuan Province under Grant 2025ZNSFSC0488; NNSFC and CAAC under Grant U2133211.

REFERENCES

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jeremy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro J Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Ségerie, Micah Carroll, Andi Peng, Phillip J. K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco di Langosco, Peter Hase, Erdem Biyik, Anca D. Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback. *ArXiv*, abs/2307.15217, 2023. URL https://api.semanticscholar.org/CorpusID:260316010.

Thomas G Dietterich. An overview of maxq hierarchical reinforcement learning. pp. 26–44, 2000.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL https://doi.org/10.1145/3054912.

Minyoung Hwang, Gunmin Lee, Hogun Kee, Chan Woo Kim, Kyungjae Lee, and Songhwai Oh. Sequential preference ranking for efficient reinforcement learning from human feedback. In *Neural Information Processing Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:268042566.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

Wenjun Jiang, Wayne Xin Zhao, Jingyuan Wang, and Jiawei Jiang. Continuous trajectory generation based on two-stage gan. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4374–4382, 2023.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: A research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Sabariswaran Mani, Abhranil Chandra, Sreyas Venkataraman, Adyan Rizvi, Yash Sirvi, Soumojit Bhattacharya, and Aritra Hazra. Diffclone: Enhanced behaviour cloning in robotics with diffusion-driven policy learning. *ArXiv*, abs/2401.09243, 2024. URL https://api.semanticscholar.org/CorpusID:267028191.

Guanren Qiao, Guiliang Liu, Pascal Poupart, and Zhiqiang Xu. Multi-modal inverse constrained reinforcement learning from a mixture of demonstrations. NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Davide Scaramuzza and Elia Kaufmann. Learning agile, vision-based drone flight: from simulation to reality. In *International Symposium of Robotics Research*, 2023. URL https://api.semanticscholar.org/CorpusID:258048997.

Lu Wang, Chaoyun Zhang, Ruomeng Ding, Yong Xu, Qihang Chen, Wentao Zou, Qingjun Chen, Meng Zhang, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 51165125, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599934. URL https://doi.org/10.1145/3580305.3599934.

Yuting Wang, Lu Liu, Maonan Wang, and Xi Xiong. Reinforcement learning from human feedback for lane changing of autonomous vehicles in mixed traffic. *ArXiv*, abs/2408.04447, 2024. URL https://api.semanticscholar.org/CorpusID:271768813.

Jian Wu, Yang Yan, Yulong Liu, and Yahui Liu. Research on anthropomorphic obstacle avoidance trajectory planning for adaptive driving scenarios based on inverse reinforcement learning theory. *Engineering*, 2023. URL https://api.semanticscholar.org/CorpusID:264952557.

Yutong Ye, Yingbo Zhou, Jiepin Ding, Ting Wang, Mingsong Chen, and Xiang Lian. Initlight: Initial model generation for traffic signal control using adversarial inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2023. URL https://api.semanticscholar.org/CorpusID:260850741.

Mustafa Yildirim, Barkin Dagda, Vinal Asodia, and Saber Fallah. Behavioral cloning models reality check for autonomous driving. *ArXiv*, abs/2409.07218, 2024. URL https://api.semanticscholar.org/CorpusID:272592989.

Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54(12):7173–7186, 2024. doi: 10.1109/TCYB.2024.3395626.

APPENDIX

A A DETAILED DESCRIPTION OF THE STATE

According to Fig.2, the passengers' boarding task contains three subtasks, i.e., check-in, security check, and boarding. The content of each state is shown in Table 3. $Freetime$ means when the passenger just enters the primitive/composite state, the difference between the current time and the passenger's flight departure time. $Queuelength$ means when the passenger just enters the primitive/composite state, the length of the queue in the check-in/security check/boarding area.

B THE MAP OF THE AIRPORT TERMINAL AND THE CORRESPONDING AREAS

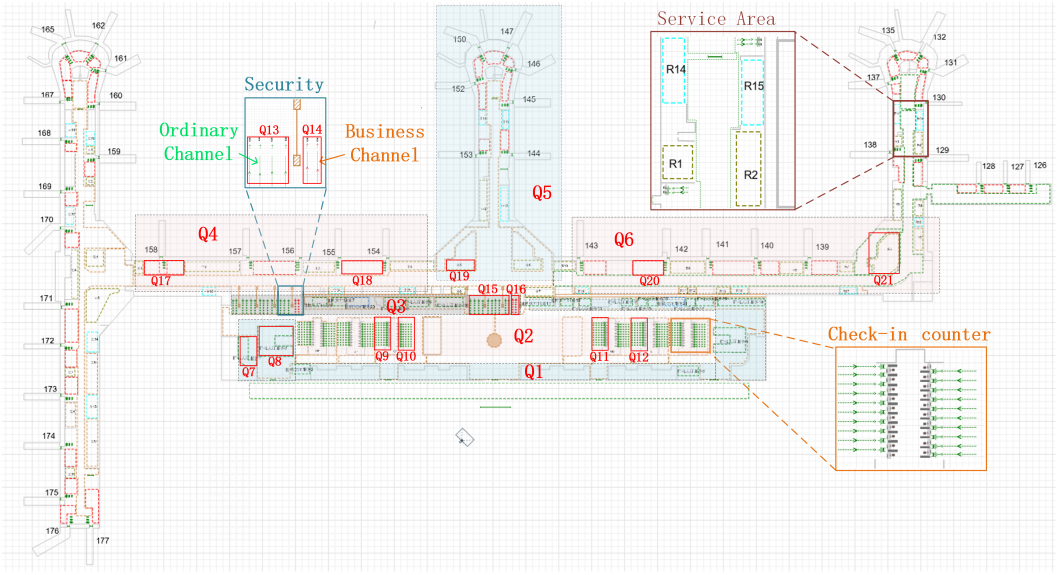

Figure 5: The map of the airport terminal and the corresponding areas

C THE THE ALIGNMENT DEGREE R AND R^* BASED ON THE MACRO LEVEL METRIC ΔQ

Table 3: Content of each state

State	Meaning	State property
$Q_{Check-in}$	$(freetime_{QCheck-in}, queue\_length_{QCheck-in})$	composite
$Q_{Check-inCounter}$	$(freetime_{QCheck-in}, queue\_length_{QCheck-in}, freetime_{QCheck-inCounter}, queue\_length_{QCheck-inCounter})$	primitive
$Q_{Service1}$	$(freetime_{QService1}, queue\_length_{QService1})$	composite
Q_{food}	$(freetime_{QCheck-in}, queue\_length_{QCheck-in}, freetime_{QCheck-inCounter}, queue\_length_{QCheck-inCounter}, freetime_{Qfood}, queue\_length_{Qfood})$	primitive
$Q_{Security}$	$(freetime_{QSercurity}, queue\_length_{QSercurity})$	composite
$Q_{SercurityChannel}$	$(freetime_{QSercurity}, queue\_length_{QSercurity}, freetime_{QSercurityChannel}, queue\_length_{QSercurityChannel})$	primitive
$Q_{Service2}$	$(freetime_{QService2}, queue\_length_{QService2})$	composite
Q_{food}	$(freetime_{QSercurity}, queue\_length_{QSercurity}, freetime_{QSercurityChannel}, queue\_length_{QSercurityChannel}, freetime_{Qfood}, queue\_length_{Qfood})$	primitive
$Q_{Boarding}$	$(freetime_{QBoarding}, queue\_length_{QBoarding})$	composite
$Q_{SercurityChannel}$	$(freetime_{QBoarding}, queue\_length_{QBoarding}, freetime_{QBoardingGate}, queue\_length_{QBoardingGate})$	primitive
$Q_{Service3}$	$(freetime_{QService3}, queue\_length_{QService3})$	composite
Q_{food}	$(freetime_{QBoarding}, queue\_length_{QBoarding}, freetime_{QBoardingGate}, queue\_length_{QBoardingGate}, freetime_{Qfood}, queue\_length_{Qfood})$	primitive

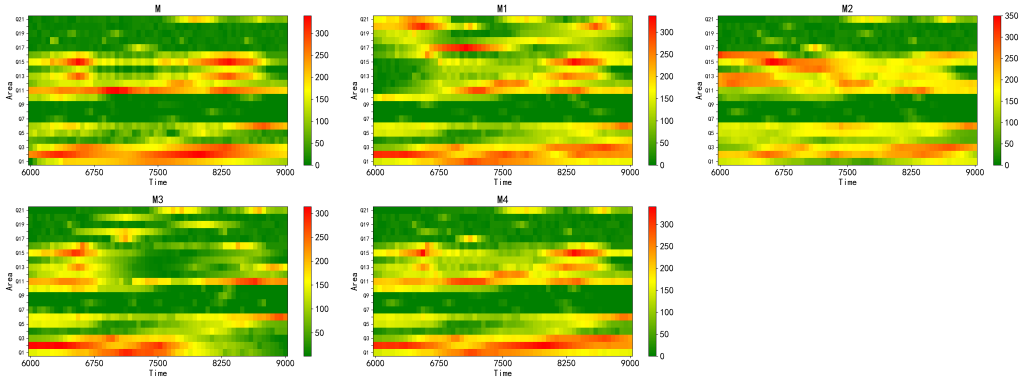

Figure 6: The flow density in original scenario with vertical coordinates for each area (Q_1-Q_{21}) and horizontal coordinates for the simulation time (6000s-9000s)

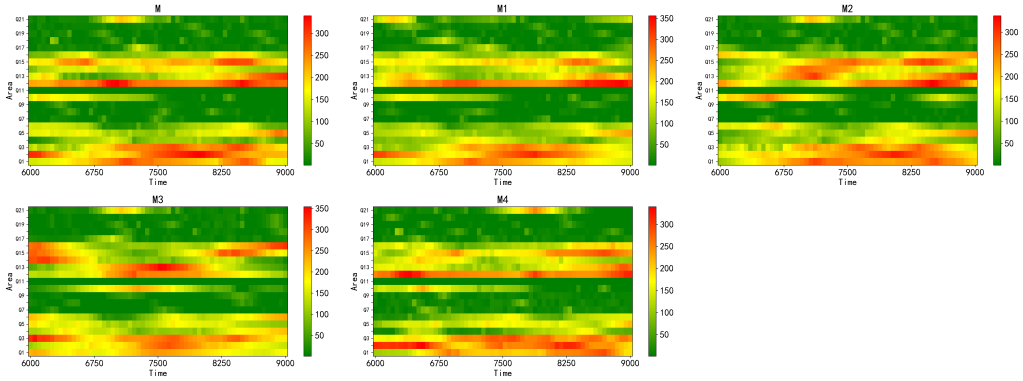

Figure 7: The flow density in scenario 1 (The Q_{11} area is closed so there is no traffic.) with vertical coordinates for each area (Q_1-Q_{21}) and horizontal coordinates for the simulation time (6000s-9000s)

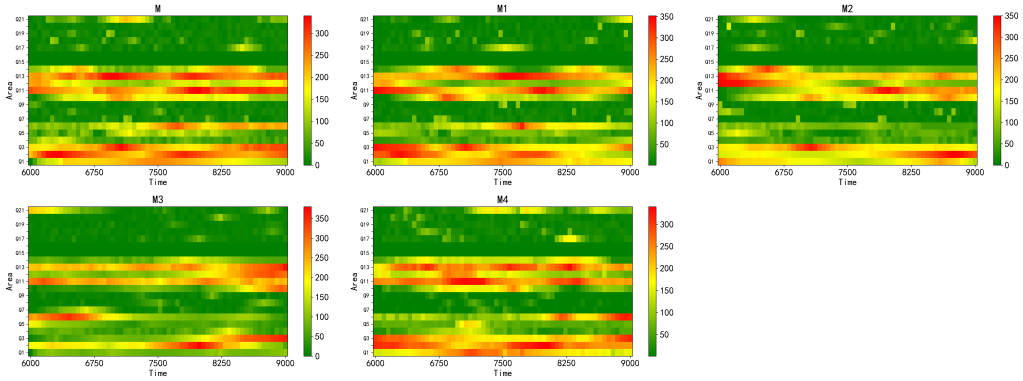

Figure 8: The flow density in scenario 2 (Areas Q_{15} and Q_{16} are closed so there are no traffic.) with vertical coordinates for each area (Q_1-Q_{21}) and horizontal coordinates for the simulation time (6000s-9000s)