Multi-Agent Archive-Based Inverse Reinforcement Learning by Improving Suboptimal Experts

Shunsuke Ueki and Keiki Takadama
Department of Informatics, The University of Electro-Communications, Tokyo, Japan

Keywords: Multi-Agent System, Reinforcement Learning, Inverse Reinforcement Learning, Maze Problem.

Abstract: This paper proposes the novel Multi-Agent Inverse Reinforcement Learning method that can acquire reward functions in continuous state space by improving the "suboptimal" expert behaviors. Specifically, the proposed method archives the superior "individual" behaviors of the agent without taking an account of other agents, selects the "cooperative" behaviors that can cooperate with other agents from the individual behaviors, and improve expert behaviors according to both the individual and cooperative behaviors to obtain the better behaviors of the agents than those of experts. The experiments based on the maze problem in a continuous state space have been revealed the following implications (1) the suboptimal expert trajectories that may collide with the other agents can be improved to the trajectories that can avoid the collision among the agents; and (2) the number of collisions of agents and the expected return in the proposed method is smaller/larger than those in MA-GAIL and MA-AIRL.

1 INTRODUCTION

Reinforcement Learning (RL) (Sutton and Barto, 1998) learns behaviors to maximize the expected rewards through trial and error in a given environment. To obtain the optimal behavior, an appropriate reward function should be designed, but such design is generally difficult as a size of the state-action space and/or a complexity of environment increases. To tackle solve this problem, Inverse Reinforcement Learning (IRL) (Russell, 1998) was proposed to estimate the reward function from optimal behaviors of agents called experts behaviors. Since multiple reward functions can be derived from an expert behavior, the maximum entropy IRL (MaxEntIRL) (Ziebart et al., 2008) was proposed to derive an unique the reward function by applying the maximum entropy principle. However, MaxEntIRL works well in a discreate state space but not in a continuous state space because the state visitation frequency probabilities of "all" states should be calculated which is impossible to cover all states in a continuous space. To address this problem, the sampling-based IRL with adversarial generative networks (Goodfellow et al., 2014) was proposed (Finn et al., 2016) for a single agent environment, and then Multi-Agent Adversarial IRL (MA-AIRL) was proposed (Yu et al., 2019) by extending Finn's method to a multiagent environment. However, MA-AIRL cannot estimates the optimal reward functions of all agents when their expert behaviors are not optimal. In multiagent environment, it goes without saying that design of expert behaviors becomes difficult as the number of agents increases and/or the environment becomes complex because of many combinations of cooperative behaviors. From this fact, it is crucial to estimate the "optimal" reward functions of all agents from their "suboptimal" expert behaviors. For this issue, the reward function is estimated by minimizing the difference between the performance learned from the suboptimal expert behaviors and that learned from Nash equilibrium solutions in multiagent environment (Wang and Klabjan, 2018). However, it is very computationally time-consuming to calculate the Nash equilibrium solutions, meaning that it is not realistic to employ this method.

To overcome this problem, this paper proposes the Archive Multi-Agent Inverse Reinforcement Learning (Archive MA-AIRL) which is extended from MA-AIRL to acquire appropriate reward functions in continuous state space by improving the suboptimal expert behaviors whose computational complexity is independent of the number of agents and the size of the state space. For this purpose, Archive MA-AIRL archives the superior "individual" behaviors of the agent without taking an account of other agents, selects the "cooperative" behaviors that can cooperate

Ueki, S. and Takadama, K.
Multi-Agent Archive-Based Inverse Reinforcement Learning by Improving Suboptimal Experts.
DOI: 10.5220/0012475100003636

In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 3*, pages 1362-1369
ISBN: 978-989-758-680-4; ISSN: 2184-433X

with other agents from the individual behaviors, and improve the suboptimal expert behaviors according to both the individual and cooperative behaviors. To investigate the effectiveness of the Archive MA-AIRL, this paper applies it into the continuous maze problem

This paper is organized as follows. Sections 2 and 3 describe reinforcement learning and inverse reinforcement learning, respectively. Section 4 proposes the archive mechanism and Archive MA-AIRL. The experiment is conducted and its result is discussed in Section 5. Finally, our conclusion is given in Section 6.

2 REINFORCEMENT LEARNING

Reinforcement learning (RL) is a method in which an agent learns a policy to maximize the reward from trajectories by repeatedly observing a state, selecting an action, and acquiring a reward in the environment. RL is modeled as Markov Decision Processes(MDPs). In this paper, we introduce Q-learning (Watkins and Dayan, 1992), a common method of reinforcement learning, in which Q values (state action values) $Q(s,a)$ are updated according to Eq. (1) in the process of repeatedly observing states, selecting actions, and obtaining rewards. We adopt the epsilon-greedy selection method for selecting actions. In this selection method, the agent chooses a random action with ε probability and selects the action with the largest Q-value with $1-\varepsilon$ probability.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in A(s')} Q(s',a') - Q(s,a) \right] \quad (1)$$

where s is the state, a is the action, s' is the next state, a' is the next action, α $(0 \leq \alpha \leq 1)$ is the learning rate, γ $(0 \leq \gamma \leq 1)$ is the discount rate, r is the reward, and $A(s')$ is the set from several reward functions..

3 INVERSE REINFORCEMENT LEARNING

Inverse reinforcement learning (IRL) is a method for estimating the reward function from optimal actions by experts. IRL can be divided into three main categories, Maximum Margin IRL (MaxMarginIRL) (Ng and Russell, 2000), Bayesian IRL (BIRL) (Ramachandran and Amir, 2007), and Maximum Entropy IRL (MaxEntIRL) (Ziebart et al., 2008). In the following section, we will introduce MaxEntIRL, its extension Adversarial IRL (AIRL), and furthermore, the extension of B into a Multi-Agent System, referred to as Multi-Agent Adversarial IRL (MA-AIRL).

3.1 Maximum Entropy IRL (MaxEntIRL)

Maximum Entropy IRL (MaxEntIRL) is a common method of IRL. MaxEntIRL can solve the ambiguity of the reward function (one expert trajectory can be acquired from several reward functions) by the maximum entropy principle. MaxEntIRL estimates the reward function such that the agent trajectory and the expert trajectory are the same. The algorithm of MaxEntIRL is shown in Algorithm 1. The reward function $R(s)$ is defined by Eq. (2)

$$R(s) = \theta^T f_s \quad (2)$$

where θ is the parameter assigned to each state and f_s is the feature of the trajectory. The feature of the trajectory f_ζ is computed by the Eq. (3) using the feature of the state $\phi(s)$ represented by the one-hot vector.

$$f_\zeta = \sum_{s \in \zeta} \phi(s) \quad (3)$$

Maximize the entropy of the probability of executing a certain trajectory under the parameter $P(\zeta|\theta)$ $(\max_\theta \sum_{\zeta \in Z} P(\zeta|\theta) \log P(\zeta|\theta))$. Let the likelihood function be

$$L(\theta) = \sum_{\zeta \in Z} P(\zeta|\theta) = \sum_{\zeta \in Z} \exp(\theta^T f_\zeta) / \sum_{\zeta \in Z} \exp(\theta^T f_\zeta) \quad (4)$$

Z is the set of the agent's trajectory. Then the optimal parameter θ^* and the gradient of the log-likelihood $\nabla L(\theta)$ are as follows:

$$\theta^* = \arg\max_\theta \left\{ \frac{1}{M} \sum_{i=1}^{M} \theta^T f_{\zeta_i} - \log \sum_{\zeta \in Z} \exp(\theta^T f_\zeta) \right\} \quad (5)$$

$$\nabla L(\theta) = \frac{1}{M} \sum_{i=1}^{M} f_{\zeta_i} - \sum_{s \in s_i} P(s_i|\theta) f_{s_i} \quad (6)$$

where M is the number of expert trajectories and $P(s_i|\theta)$ is the expected state visited frequencies (SVF) calculated by Eq. (7) using the policy computed $\pi_\theta(a|s)$ by RL. The learning process of RL in IRL is called inner-loop learning.

Algorithm 1: MaxEntIRL.

1: Set the expert trajectory ζ_{expert}.
2: Initialize the reward functions $R(s)$ and the reward parameters θ.
3: **for** $cycle := 0$ to N_{cycle} **do**
4: $R(s) = (\theta)^T \phi(s)$
5: Compute policies $\pi(s)$ (e.g. $\pi(s)$ is calculated by Q-learning).
6: Update reward parameters: $\nabla L(\theta) = f_{expert} - \sum_j P(s_j|\theta) f_{s_j}$ $\theta \leftarrow \theta - \alpha \nabla L(\theta)$
7: **end for**

$$P(s_i|\theta) = \sum_{i=1}^{T} \mu_t(s_i) \tag{7}$$

$$\mu_t(s_i) = \sum_{s \in S} \sum_{a \in A} \mu_{t-1}(s) \pi_\theta(a|s) P(s|a,s) \tag{8}$$

where T is maximum number of steps, $P(s|a,s)$ is state transition probability.

The parameter θ of the reward function is updated by the product of the gradient $\nabla L(\theta)$ and the learning rate α $(0 \leq \alpha \leq 1)$.

$$\theta \leftarrow \theta - \alpha \nabla L(\theta) \tag{9}$$

3.2 Adversarial IRL

Adversarial Inverse Reinforcement Learning (AIRL) (Finn et al., 2016) is a form of learning and sampling-based inverse reinforcement learning. AIRL reconstructs the reward function based on Generative Adversarial Imitation Learning (GAIL) that employs Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). GAIL involves a generator, which is the agent's policy π_θ, and a discriminator D_ω that identifies pairs of states s and actions a sampled from either the generator or an expert's policy π_E. Minimization and maximization of θ and ω are performed according to Equation (10).

$$\min_\theta \max_\omega \mathbb{E}_{\pi_E}[\log D_\omega(s,a)] + \mathbb{E}_{\pi_\theta}[\log(1 - D_\omega(s,a))] \tag{10}$$

The discriminator D_ω is expressed as $D_\omega(s,a) = \frac{\exp(f_\omega(s,a))}{\exp(f_\omega(s,a)) + q(a|s)}$, where $q(a|s)$ represents the probability computed by the generator. In AIRL, $f_\omega(s,a)$ is defined as $f_{\omega,\phi} = g_\omega(s^t,a^t) + \gamma h_\phi(s^{t+1}) - h_\phi(s^t)$ to reconstruct $g_\omega(s^t,a^t)$ as the reward.

3.3 Multi-Agent Adversarial IRL

The algorithm for Multi-Agent Adversarial Inverse Reinforcement Learning (MA-AIRL) (Yu et al., 2019) is shown in Algorithm 2. MA-AIRL adapts the Logistic Stochastic Best Response Equilibrium (LSBRE), which is well-suited for multi-agent environments, to AIRL.

$$\boldsymbol{\pi}^t(a_1,...,a_n|s^t) = P\left(\bigcap_i z^{t,\infty}(s^t) = a_i\right) \tag{11}$$

where the index i corresponds to each agent, and $z^{t,\infty}$ represents the state of the t-th Markov chain at ∞ steps, calculated as follows:

$$\begin{aligned} z^{t,(k+1)}(s^t) &\sim P_i^t(a_i^t|\boldsymbol{a}_{-i}^t = \boldsymbol{z}_{-i}^{t,(k)}(s^t), s^t) \\ &= \frac{\exp(\lambda Q_i^{\boldsymbol{\pi}^{t+1:T}}(s^t, a_i^t, \boldsymbol{z}_{-i}^{t,(k)}(s^t)))}{\sum_{a_i'} \exp(\lambda Q_i^{\boldsymbol{\pi}^{t+1:T}}(s^t, a_i', \boldsymbol{z}_{-i}^{t,(k)}(s^t)))}. \end{aligned} \tag{12}$$

$\boldsymbol{\pi}^{\varnothing}$ represents a time-dependent policy, and $Q_i^{\boldsymbol{\pi}^{t+1:T}}$ denotes the state-action value function with the policy entropy term added, defined as follows:

$$\begin{aligned} Q_i^{\boldsymbol{\pi}^{t+1:T}}(s^t, a_i^t, \boldsymbol{a}_{-i}^t) &= r_i(s^t, a_i^t, \boldsymbol{a}_{-i}^t) \\ &+ \mathbb{E}_{s^{t+1} \sim P(\cdot|s^t,a^t)} \Big[\mathcal{H} \pi_i^{t+1}(\cdot|s^{t+1}) \\ &+ \mathbb{E}_{a^{t+1} \sim \boldsymbol{\pi}(\cdot|s^{t+1})} \left[Q_i^{\boldsymbol{\pi}^{t+2:T}}(s^{t+1}, \boldsymbol{a}^{t+1}) \right] \Big]. \end{aligned} \tag{13}$$

In the context of LSBRE, MA-AIRL involves minimizing the Kullback-Leibler (KL) divergence between the probability of the expert performing trajectories ζ, denoted as $\hat{p}(\zeta)$, and the probability of the agent obtaining trajectories ζ, denoted as $\hat{p}(\zeta)$.

$$\min_{\hat{\boldsymbol{\pi}}^{1:T}} D_{KL}(\hat{p}(\zeta) || p(\zeta)) \tag{14}$$

$$\begin{aligned} \hat{p}(\zeta) = &\left[\eta(s^1) \prod_{t=1}^{T} P(s^{t+1}|s^t, \boldsymbol{a}^t) \boldsymbol{\pi}_{-i}^t(\boldsymbol{a}_{-i}^t|s^t) \right] \\ &\cdot \prod_{t=1}^{T} \hat{\pi}_i^t(a_i^t|\boldsymbol{a}_{-i}^t, s^t) \end{aligned} \tag{15}$$

$$\begin{aligned} \hat{p}(\zeta) \propto &\left[\eta(s^1) \prod_{t=1}^{T} P(s^{t+1}|s^t, \boldsymbol{a}^t) \boldsymbol{\pi}_{-i}^t(\boldsymbol{a}_{-i}^t|s^t) \right] \\ &\cdot \exp \sum_{t=1}^{T} r_i(s^t, a_i^t, \boldsymbol{a}_{-i}^t) \end{aligned} \tag{16}$$

This optimization problem can be transformed through the entropy maximization in MaxEntIRL as follows:

$$\max_{\omega} \mathbb{E}_{\zeta\ \pi_E}\left[\sum_{t=1}^{T}\log \pi^t(\boldsymbol{a}^t, s^t; \omega)\right]. \tag{17}$$

The loss function becomes as follows:

$$\max_{\omega} \mathbb{E}_{\pi_E}\left[\sum_{i=1}^{N}\sum_{t=1}^{T}\frac{\partial}{\partial \omega} r_i(s^t, \boldsymbol{a}^t; \omega_i)\right] \sum_{i=1}^{N}\frac{\partial}{\partial \omega}\log Z_{w_i}. \tag{18}$$

Similar to AIRL, we use the sampling-based estimation q_θ obtained through Z.

The discriminator learns by maximizing as follows with respect to ω.

$$\max_{\omega} \quad \mathbb{E}_{\pi^E}\left[\sum_{i=1}^{N}\log\frac{exp(f_{\omega_i}(s,\boldsymbol{a}))}{exp(f_{\omega_i}(s,\boldsymbol{a}))+q_{\theta_i}(a_i|s)}\right] + \mathbb{E}_{q^\theta}\left[\sum_{i=1}^{N}\log\frac{q_{\theta_i}(a_i|s))}{exp(f_{\omega_i}(s,\mathbf{a}))+q_{\theta_i}(a_i|s)}\right] \tag{19}$$

The generator learns by maximizing as follows with respect to θ.

$$\max_{\theta} \mathbb{E}_{q_\theta}\left[\sum_{i=1}^{N} f_{\omega_i}(s,\boldsymbol{a}) - \log q_{\theta_i}(a_i|s)\right] \tag{20}$$

In MA-AIRL, $f_{\omega_i,\phi_i} = g_{\omega_i}(s^t, a^t) + \gamma h_{\phi_i}(s^{t+1}) - h_{\phi_i}(s^t)$ is defined to reconstruct $g_{\omega_i}(s^t, a^t)$ as the reward.

4 METHOD

4.1 Archive Multi-Agent Adversarial Inverse Reinforcement Learning

To obtain the optimal reward function from quasi-optimal expert trajectories, we propose Archive Multi-Agent Adversarial Inverse Reinforcement Learning (Archive MA-AIRL).

4.2 Archive Multi-Agent Adversarial IRL

Archive MA-AIRL extends the MA-AIRL approach with an archive mechanism. This mechanism archives trajectories generated by the generator when they exhibit superior performance and treats them as expert trajectories, aiming to acquire the optimal reward function. Figure 1 depicts Archive MA-AIRL.

Algorithm 2: MA-AIRL.

Set the expert trajectory $\mathcal{D}^{expert} = \{\zeta_j^E\}$.
Initialize the parameters of policies $\boldsymbol{q}$, reward estimators $\boldsymbol{g}$ and potential functions $\boldsymbol{h}$ with $\boldsymbol{\omega}, \boldsymbol{\omega}, \boldsymbol{\phi}$.
for $iteration := 0$ to $N_{iteration}$ **do**
  Sample trajectories $\mathcal{D}_\pi = \{\zeta_j\}$ from $\boldsymbol{\pi}$.
  Sample sate-action pairs $\mathcal{X}_\pi$ from $\mathcal{D}_\pi$.
  Sample $\mathcal{X}^{expert}$ from $\mathcal{D}^{expert}$
  for $i := 0$ to N_{agent} **do**
    Update ω_i, ϕ_i to increase the objective in Eq. 19
  end for
  for $i := 0$ to N_{agent} **do**
    Update reward estimates $\hat{r}_i(s, a_i, s')$ with $g_{\omega_i}(s, a_i)$ or $(\log D(s, a_i, s') - log(1 - D(s, a_i, s')))$.
    Update ω_i with respect to $\hat{r}_i(s, a_i, s')$
  end for
end for

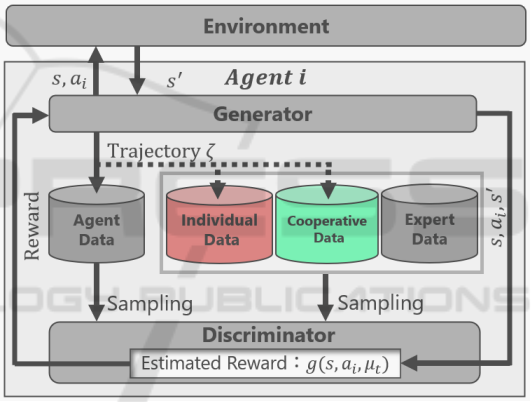


Figure 1: The architecture of Archive Multi-Agent Adversarial IRL.

The proposed method adds three steps to MA-AIRL: trajectories evaluation (I), trajectories archiving (II), and selection of expert state-action sampling for the dataset (III).

(I) Trajectories evaluation involves assessing trajectories ζ generated by the generator. Evaluation encompasses individual evaluation functions, E_{indi}, assessing trajectories based on individual achievement of goals, and a collaborative evaluation function, E_{coop}, rating trajectories based on their collaboration with other agents. These evaluations contribute to an overall evaluation function, $E_{opt} = E_{indi} + E_{coop}$. For example, in a maze problem, E_{indi} might prioritize quicker goal achievement, while E_{coop} might emphasize avoiding collisions with other agents.

(II) Trajectories archiving involves saving sequences with evaluated scores exceeding a certain threshold into either individual archives $\mathcal{D}^{indi}$ or op-

timal archives $\mathcal{D}^{opt}$, based on evaluations from (I). $\mathcal{D}^{indi}$ stores trajectories with individual evaluations E_{indi} surpassing a specific threshold, while $\mathcal{D}^{opt}$ retains trajectories with overall evaluations E_{entire} exceeding a set threshold. Individual and collaborative archives operate on a per-agent basis to archive trajectories. If the number of trajectories within the archive exceeds the predefined limit set by the user, low-rated trajectories are removed from the archive. Consequently, the low-rated trajectories gradually get replaced by higher-rated ones.

(III) Selection of expert state-action sampling for the dataset involves choosing datasets used in the discriminator's training based on the following rules:

For all agents, if trajectories archived in the optimal archive $\mathcal{D}^{opt}$ exist, only $\mathcal{D}^{opt}$ is sampled. If no agent has trajectories archived in $\mathcal{D}^{opt}$ but trajectories exist in the individual archive $\mathcal{D}^{indi}$ for all agents, either $\mathcal{D}^{indi}$ or the initially provided expert dataset $\mathcal{D}^{expert}$ is sampled. If no agent has trajectories in both $\mathcal{D}^{opt}$ and $\mathcal{D}^{indi}$, only the initially provided expert dataset $\mathcal{D}^{expert}$ is sampled. Sampling is performed by randomly selecting state-action pairs from the dataset. If the number of data samples to be collected exceeds the total number of samples in the dataset, sampling is conducted allowing for duplication.

4.3 Algorithm

Algorithm 3 presents the algorithm for Archive MA-AIRL. To start, provide quasi-optimal expert trajectories ζ_{expert} to each of the N agents. Initialize the parameters $\boldsymbol{q}$ for the generator's policy and $\boldsymbol{g}, \boldsymbol{h}$ for the discriminator as $\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\phi}$. Execute a maximum of $N_{iteration}$ iterations as pre-set. Sample trajectories using the policy $\boldsymbol{\pi}$ of the generator. Execute a maximum of $N_{iteration}$ iterations as pre-set. Sample trajectories using the policy $\boldsymbol{\pi}$ of the generator. Evaluate the sampled trajectories. Save trajectories with high individual evaluations into the individual archive $\mathcal{D}^{indi}$ and those with high overall evaluations into the optimal archive $\mathcal{D}^{opt}$. Sample data from the sampled agent's trajectories to match the batch size for learning. For all agents, if trajectories are archived in the optimal archive $\mathcal{D}^{opt}$, only sample from $\mathcal{D}^{opt}$. If trajectories archived in the optimal archive $\mathcal{D}^{opt}$ are not available for any agent, and trajectories archived in the individual archive $\mathcal{D}^{indi}$ exist for all agents, sample from either the individual archive $\mathcal{D}^{indi}$ or the initially provided expert dataset $\mathcal{D}^{expert}$. If trajectories archived in either the optimal archive $\mathcal{D}^{opt}$ for any agent or the individual archive $\mathcal{D}^{indi}$ for any agent are unavailable, sample solely from the initially provided expert dataset $\mathcal{D}^{expert}$. Update ω_i, ϕ_i for each agent. Update reward functions $\hat{r}_i$ for each agent. Also, update ω_i with the updated reward functions $\hat{r}_i$.

Algorithm 3: Archive MA-AIRL.

Set the expert trajectory $\mathcal{D}^{expert} = \{\zeta_j^E\}$.
Initialize the parameters of policies $\boldsymbol{q}$, reward estimators $\boldsymbol{g}$ and potential functions $\boldsymbol{h}$ with $\boldsymbol{\theta}, \boldsymbol{\omega}, \phi$.
for $iteration := 0$ to $N_{iteration}$ **do**
  Sample trajectories $\mathcal{D}_\pi = \{\zeta_j\}$ from $\boldsymbol{\pi}$.
  Evaluate sampled trajectories $\mathcal{D}_\pi$
  Store individual data, optimum data to $\mathcal{D}^{indi}$, $\mathcal{D}^{opt}$
  Sample sate-action pairs $\mathcal{X}_\pi$ from $\mathcal{D}_\pi$.
  if $\bigwedge_{i=0}^{N_{agent}} |\mathcal{D}_i^{opt}| > 0$ **then**
    Sample $\mathcal{X}^{expert}$ from $\mathcal{D}^{opt}$
  else if $\bigwedge_{i=0}^{N_{agent}} |\mathcal{D}_i^{indi}| > 0$ **then**
    Sample $\mathcal{X}^{expert}$ from $\mathcal{D}^{indi}$, $\mathcal{D}^{expert}$
  else
    Sample $\mathcal{X}^{expert}$ from $\mathcal{D}^{expert}$
  end if
  for $i := 0$ to N_{agent} **do**
    Update ω_i, ϕ_i to increase the objective in Eq. (19)
  end for
  for $i := 0$ to N_{agent} **do**
    Update reward estimates $\hat{r}_i(s, a_i, s')$ with $g_{\omega_i}(s, a_i)$ or $(\log D(s, a_i, s') - log(1 - D(s, a_i, s')))$.
    Update ω_i with respect to $\hat{r}_i(s, a_i, s')$
  end for
end for

5 EXPERIMENT

Verifying whether the proposed method can acquire the optimal trajectories of actions when given suboptimal experts in a continuous state space maze problem.

5.1 Problem Settings

Each agent can choose to move up, down, left, right, or take no action at each step, applying a force of 1N in the selected direction. The environment is set with specific start and goal locations, and a game consists of 50 steps. After reaching the goal, agents don't remain in the environment. Agents cannot be outside the environment. The environment involves two agents, Agent 0 and Agent 1, with Agent 0 starting at the top left with the goal at the bottom right, while Agent 1 starts at the top right with the goal at the bottom left.

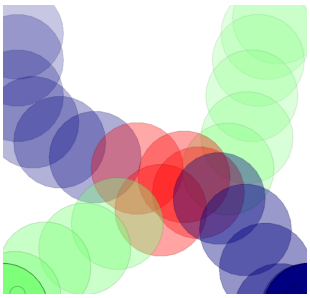
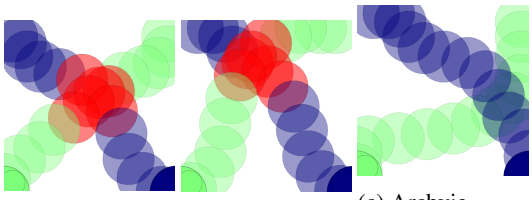


Figure 2: Example of expert trajectory



(a) MA-GAIL (b) MA-AIRL (c) Archvie MA-AIRL

Figure 3: Results of sampled trajectory from learned policy.

The true reward function provides a +100 reward upon reaching the goal, a negative reward based on the distance to the goal, and a -100 reward for collisions. Initially, the expert provided is trained for $10e^5$ steps using ACKTR (Wu et al., 2017) in a single-agent environment based on the true reward function (note that collisions do not occur, so negative rewards due to collisions are not applicable). From the learned policy, 1000 state-action pairs are sampled.

An example of trajectories sampled from the expert's policy is depicted in Fig. 2. Agent 0 is represented in blue, Agent 1 in green, and collisions are depicted in red. Since the agents collide with each other, the expert experts are considered to be quasi-optimal.

5.2 Parameter Settings

The parameters for MA-GAIL, MA-AIRL, and Archive MA-AIRL were set as follows: discount factor of 0.99, batch size of 500 steps (This is the number of state-action pairs sampled from the agent data and expert data), and 550 update iterations (total timesteps is 2.75×10^6).

5.3 Experiment Results

Fig. 3 shows the trajectories sampled from policies learned by each method. Blue represents the trajectory of Agent 0, green represents the trajectory of Agent 1, and red indicates a collision. MA-GAIL and MA-AIRL collide with agents, but Archive MA-AIRL does not.

Table 1 presents the results of sampling 1000 trajectories from the learned policies, showing the average expected returns calculated for each agent by using true reward and the average total number of collisions in 30 seeds. Based on the expected return results in Table 1, Archive MA-AIRL outperformed Expert, MA-GAIL and MA-AIRL. In addition, the number of collisions is the lowest value for Archive MA-AIRL. These indicate that Archive MA-AIRL has acquired a better trajectory than MA-GAIL and MA-AIRL. The difference between Archive MA-AIRL and MA-AIRL is archive mechanism. Therefore, the archive mechanism can contribute to obtaining a better trajectory.

Fig. 4 shows the sum of expected returns in Agent 0 and Agent 1 calculated by using true reward duaring learning. Fig. 5 shows the number of collisions between Agent 0 and Agent 1 during learning. Fig. 6 shows the average number of steps to reach the goal for A and B during learning. These values in figures are averaging 30 seeds. From Fig. 4, it is evident that around 7000 total timesteps, it surpasses MA-GAIL and around 10000 total timesteps, it surpasses MA-AIRL. Until approximately 10000 total timesteps, there isn't a significant difference between MA-AIRL and Archive MA-AIRL. Fig. 5 indicates that around 7000 total timesteps, the collision count of MA-GAIL falls below, and around 10000 total timesteps, it falls below the collision count of MA-AIRL. From the result of Fig. 6, all methods are able to learn shortest trajectories. Archive MA-AIRL and MA-AIRL converge to the shortest step count around 10000 total timesteps, while MA-GAIL converges to the shortest step count at 15000 total timesteps.

5.4 Discussion

From the experiment results, the proposed method was able to outperform both the experts and the conventional methods (MA-GAIL, MA-AIRL). The performance improvement was confirmed as the generated trajectories were appropriately added to the expert's dataset through the archive mechanism. Archive MA-AIRL can acquire more optimal non-collision trajectories without compromising the learning efficiency of MA-AIRL by incorporating the Archive mechanism.

Fig. 7 shows examples of three archived trajectories. These trajectories are stored in the archive and consist of non-collision trajectories.

Fig. 8 shows trajectories sampled from policies for the proposed method. Until agents are learning with individual data, the behavior resembled that of the expert, resulting in collisions. However, by

Table 1: Results of the average expected returns and the average total number of collisions.

Algorithm	Expected returns		Number of Collision
	Agent 0	Agent 1	
Expert	-220.8	-220.0	3.0
MA-GAIL	-119.6	-118.8	2.0
MA-AIRL	-44.0	-42.8	1.3
Archive MA-AIRL	**10.6**	**12.5**	**0.7**

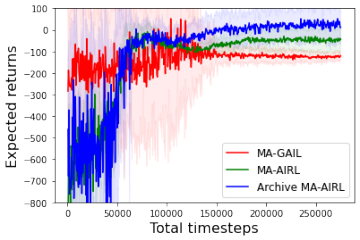


Figure 4: Expected returns for each algorithm

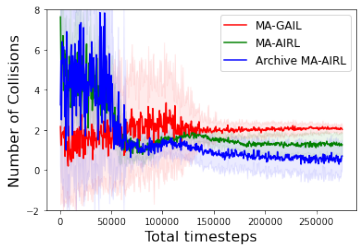


Figure 5: Number of collisions for each algorithm

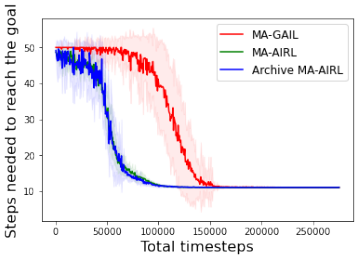
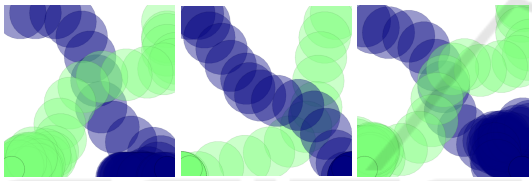


Figure 6: Number of steps to reach goal for each algorithm.

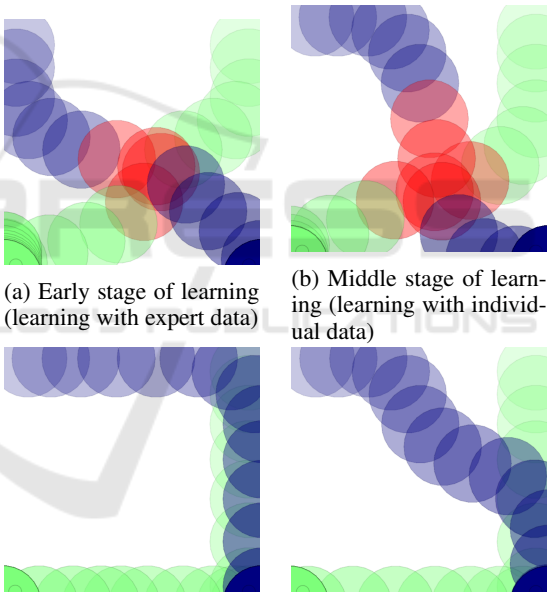


(a) Example 1 (b) Example 2 (c) Example 3

Figure 7: Examples of archived cooperative trajs.

archiving cooperative trajectories and agents learning with them, both Agent 0 and Agent 1 learned to navigate along the edges, avoiding collisions. Additionally, by improving cooperative data, Agent 0 learned shorter non-collision trajectories.

Fig. 9 shows the changes observed during the learning with cooperative trajectories while improving them. In Fig. 9a, agents learned the trajectory that do not collide with agents, but Agent 0 trying to go out of area and slowed down. In Fig. 9b, agents learned a trajectory that do not collide with agents and Agent 0 is stay in the area. Fig. 9b'trajectory is a shorter trajectoryway than Fig. 9a's trajectory, and the archive is appropriately improved. In Fig. 9c, agents learned short collision trajectories. Such collision trajectories are not in the cooperative archive, but in the process of learning short trajectories, they learned trajectories that collide with each other. However, In Fig. 9d, agents learned short non-collision trajectories. As a result, it is obvious that Agent 0 has learned the shortest trajectory from the trajectory along the edge to the goal without collision by improving the archive.



(a) Early stage of learning (learning with expert data)

(b) Middle stage of learning (learning with individual data)

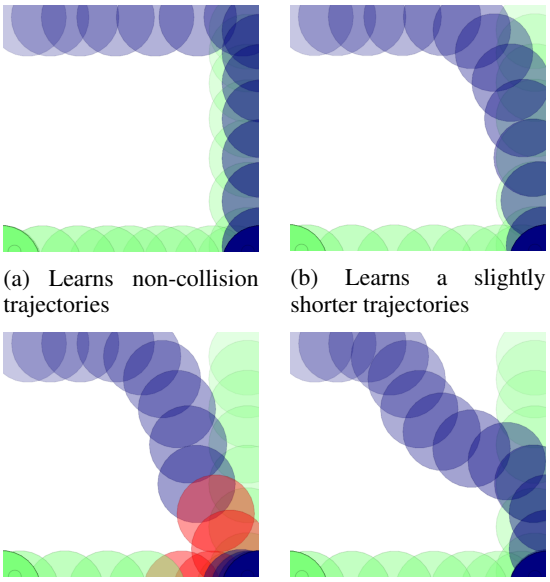
(c) Middle stage of learning (learning with cooperative data)

(d) Late stage of learning (learning with cooperative data)

Figure 8: The overall process of learning (Archive MA-AIRL).

6 CONCLUSIONS

This paper proposed Archive MA-AIRL that can acquire reward functions in continuous state space by improving the "suboptimal" expert behaviors. Specifically, Archive MA-AIRL archives the superior "individual" behaviors of the agent, selects the "cooperative" behaviors from the individual behaviors, and im-



(a) Learns non-collision trajectories

(b) Learns a slightly shorter trajectories

(c) Learns short collision trajectories

(d) Learns shorter non-collision trajectories

Figure 9: Process of improving the archive trajectories (Archive MA-AIRL).

proves the expert behaviors according to both the individual and cooperative behaviors to obtain the better behaviors of the agents than those of experts. For this purpose, the discriminator in Archive MA-AIRL evaluates whether the behaviors generated by the generator are close to the behaviors of experts improved from both individual and collective trajectories. To investigate the effectiveness of Archive MA-AIRL, this paper applied it into the continuous maze problem and the following implications have been revealed: (1) The trajectories that can avoid the collision among the agents can be acquired from the suboptimal expert trajectories that may collide with the other agents (2) Archive MA-AIRL outperforms MA-GAIL and MA-AIRL as the conventional methods in addition to the experts from the viewpoint of the number of collisions of agents and expected return.

What should be noticed here is that these results have only been obtained from the simple testbeds, i.e., the maze problem, therefore further careful qualifications and justifications, such as complex maze problems, are needed to generalized the obtained implications. Such important directions must be pursued in the near future in addition to (1) an exploration of the proper evaluation of trajectories because. the incorrect evaluation of trajectories might deteriorate the archived trajectories and (2) an increase of the number of agents.

REFERENCES

Finn, C., Levine, S., and Abbeel, P. (2016). Guided cost learning: Deep inverse optimal control via policy optimization. In *the 33rd International Conference on Machine Learning*, volume 48, pages 49–58.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Ng, A. Y. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *the 17th International Conference on Machine Learning*, pages 663–670.

Ramachandran, D. and Amir, E. (2007). Bayesian inverse reinforcement learning. In *the 20th international joint conference on Artifical intelligence*, pages 2586–2591.

Russell, S. (1998). Learning agents for uncertain environments. In *the eleventh annual conference on Computational learning theory*, pages 101–F103.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: An introduction. *A Bradford Book*.

Wang, X. and Klabjan, D. (2018). Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations. In *the 35th International Conference on Machine Learning*, volume 80, pages 5143–5151.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Wu, Y., Mansimov, E., Liao, S., Grosse, R., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *the 31st International Conference on Neural Information Processing Systems*, pages 5285–5294.

Yu, L., Song, J., and Ermon, S. (2019). Multi-agent adversarial inverse reinforcement learning. In *the 36th International Conference on Machine Learning*, volume 97, pages 7194–7201.

Ziebart, B. D., Maas, A., Bagnell, J., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438.