IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

MARRGM: Learning Framework for Multi-agent Reinforcement Learning via Reinforcement Recommendation and Group Modification

Peiliang Wu,Liqiang Tian,Qian Zhang,Bingyi Mao, Wenbai Chen

***Abstract*—Sample usage efficiency is an important factor affecting the convergence speed of multi-agent deep reinforcement learning (MADRL) algorithms. Most existing experience replay (ER) methods manually select experience samples to update the agent's policy. It is difficult to give suitable and efficient experience samples for different stages of agent policy learning as well as to effectively mine the potential value of experience samples in the replay buffer. Inspired by the idea of recommendation systems, this paper proposes a MADRL framework based on reinforcement recommendation and group modification to improve sample use efficiency and the ability to find the optimal solution of the multi-agent system in different task scenario categories. First, we use the sampling probability of each experience sample output from the recommendation network to recommend sampling instead of manual sampling; simultaneously, we collect the performance of the multi-agent system after updating the policy with the experience sample of recommendation sampling and construct the reinforcement learning process of the recommendation network. Next, we modify the individual policy of the agent according to the group rewards to improve the agent's ability to learn the optimal solution. We then combine and embed the reinforcement recommendation and group modification modules into the MADRL algorithm MAAC. Finally, we experiment with task scenarios, including cooperative collection, command movement, and target navigation, and extend this framework to the MADDPG algorithm to verify its scalability. The experimental results show that the off-policy MADRL algorithms combined with the proposed framework outperform the baseline algorithm in terms of sample usage efficiency and have better universality for the number of agents and scene categories.**

***Index Terms*—Multi-agent systems, Deep reinforcement learning, Multi-agent reinforcement learning, Reinforcement recommendation system, Group modification.**

Manuscript received: January 6, 2024; Revised: April 1, 2024; Accepted: April 12, 2024.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work is partially supported by the National Natural Science Foundation of China (62276028, U20A20167), Natural Science Foundation of Hebei Province (F202103079), Hebei Province Central Leading Local Science and Technology Development Project (236Z1811G), Innovation Capability Improvement Plan Project of Hebei Province (22567626H). *(Corresponding author: Liqiang Tian.)*

Peiliang Wu, Liqiang Tian and Bingyi Mao are with School of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, China (tlq@stumail.ysu.edu.cn)

Qian Zhang is with College of software, Nankai University, TianJin, 300350, China (1120230334@mail.nankai.edu.cn)

Wenbai Chen is with School of Automation, Beijing Information Science and Technology University, Beijing, 100192, China (chenwb@bistu.edu.cn)

Digital Object Identifier (DOI): see top of this page.

I. Introduction

REINFORCEMENT learning (RL) is a machine learning method in which an agent learns a specific policy to accomplish a certain task using the trial and error method through experience gained by interacting with the environment. With the development of deep neural networks (DNNs), RL has been combined with it to form deep reinforcement learning (DRL), which has achieved great success in many fields; examples include Alpha Go [1], robot skill learning [2]–[4], and autonomous driving [5], [6]. Several real-world applications require multiple agents to collaborate to accomplish specific tasks. Therefore, multi-agent reinforcement learning (MARL), which aims to determine the optimal policy for multiple agents interacting in the same environment, has received extensive attention from researchers in various fields.

Existing MADRL algorithms typically adopt the basic paradigm of centralized training with decentralized execution (CTDE). On this basis, several classical MADRL algorithms have been generated. Lowe et al. [7] have mapped agent-specific observations to agent-specific actions to stabilize the environment. Iqbal et al. [8] successfully combined the attention mechanism using a multi-agent algorithm, and proposed integrating the attention mechanism into the critic. Wang et al. [9] used the double dueling network to transform the IGM principle [10] into a dominance function constraint. Wang et al. [11] used a role selector to decompose the joint action space into a limited action space and proposed a learning method for role-based policy decision-making. Liu et al. [12] proposed an exploration algorithm that reshaped the reward function to train the exploration policy of the agents. Ding et al. [13] design the multi-agent dueling q-learning method based on mean-filed theory and individual value decomposition to tackle the competitive tasks. Current deep RL methods benefit from the experience replay (ER) mechanism, which smooths the training distribution over past behaviors and mitigates the problems with nonstationary distributions. However, it is not straightforward to apply experience replay to multi-agent systems as experience becomes outdated over time and agent policies change [14].

To solve these problems, Foerster et al. [15] proposed a method that attenuates outdated experience samples from replay memory to stabilize the target. Andrychowicz et al. [16] set a 'virtual goal' that can be completed for the experience samples and solve the problem of sparse reward to some extent. Palmer et al. [17] used leniency treatment for each

stored experience sample and associated it with a temperature value to promote the cooperation problem of MARL. Fedus et al. [18] conducted a systematic and extensive analysis of experience replays and observed that increasing replay capacity and shortening the age of the oldest policy can improve performance. Li et al. [19] proposed a clustering experience replay method that divides the entire training process into several stages.

The above studies on the replay of experience samples all use manual settings, which can alleviate the problem of non-stationary distribution of experience data to an extent, but cannot find the best samples suitable for the current policy in time. Inspired by the recommendation system, we propose a framework based on reinforcement recommendation group modification (MARRGM). By using the recommendation network to recommend samples, constructing a reinforcement learning process to learn the recommendation policy, and using the group reward to constrain the individual policy, the framework improves the sample usage efficiency and the ability to find the optimal solution. We combine MARRGM with the popular MADRL algorithm MAAC [8] and conduct experiments in three environments to validate the benefits of MARRGM and the modules within it. We also implemented a recommendation module called NU-RR without a reinforcement framework to verify the improvement of the reinforcement framework. In addition, we combine MARRGM with the MADDPG [7] algorithm to validate the scalability of the framework in the target navigation environment. Experimental results show that the multi-agent system incorporating MARRGM can better use efficient experience samples for learning in all of the above tasks, outperforming the baseline algorithm in terms of algorithm convergence speed and ability to find optimal solutions.

We summarize our main contributions as follows:

- We propose a reinforcement recommendation (RR) module. This module uses a recommendation network instead of random sampling for experience replay and constructs a reinforcement learning process for the recommendation network to improve the efficiency of samples.
- We propose a group modification (GM) module. This module modifies the direction of the individual policy according to the group reward to keep the individual policy consistent with team tasks and to improve the ability of agents to find the optimal solution.
- We combine the RR and GM modules to propose a general multi-agent learning framework, MARRGM. This framework can be combined with most existing off-policy MADRL algorithms, such as MADDPG and MAAC, to improve the sample usage efficiency of multi-agent systems.

The remainder of this paper is organized as follows: Section II introduces the components and algorithm implementation of the MARRGM framework. Section III describes the experimental environment and analyzes the results. Finally, the conclusions of this paper are presented in Section IV.

II. Learning Framework for MARRGM

In this section, we introduce the learning framework, MARRGM. This framework combines the recommendation system with MARL and uses advanced recommendation models instead of the traditional manual extraction to recommend experience samples in the replay buffer. In addition, the group reward is used to modify the objective function of the agent, which alleviates the problem of the agent falling into a suboptimal solution to a certain extent. RR recommends efficient learning experience samples from the replay buffer for the update policy and uses the cumulative return of the multi-agent system in the next stage as the reward feedback of the recommendation system to update the recommendation policy; therefore, a reinforcement learning process is formed, which improves the efficiency of the use of experience samples and accelerates the learning speed of the agent. In addition, to prevent the agent from learning a suboptimal policy owing to the dense individual reward, we use the group reward to modify the individual policy of the agent. The individual and group average rewards are recorded during the training process, and the difference is calculated to impose constraints on the policy network when the agents update policies. The entire framework is structured as shown in Fig. 1.

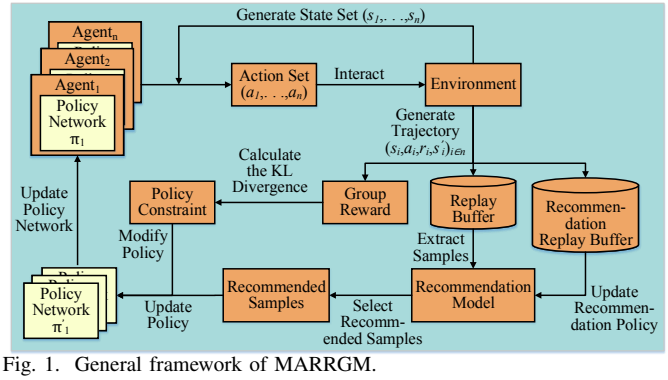


Fig. 1. General framework of MARRGM.

A. Reinforcement recommendation system

Sample usage efficiency is an important factor affecting the policy-learning speed of an agent. Improving the efficiency of sample usage can make more effective use of the existing samples to train the policy and obtain a better policy. A better policy can also guide the selection of interactive actions to obtain more valuable samples and accelerate the learning speed of the agent. Therefore, selecting high-quality experience samples can effectively improve an agent's learning speed. The RR system combines a recommendation system with MARL and uses the recommendation model to find high-quality samples to improve the learning speed of the agent.

To combine the recommendation system with MARL, the experience samples in the replay buffer need to be processed first. In MARL, the form of storing experience samples in the replay buffer can be described as $([n,m,d_s],[n,m,d_a],[n,m,d_r],[n,m,d_{s'}])$, where $d_s, d_a, d_r, d_{s'}$ represent the dimensions of state, action, reward, and next state, respectively. n is the number of agents, and m is the number of experience samples existing in the current replay buffer. First, dimension transformation, feature extraction, and feature fusion are performed on the experience samples in the replay buffer. And transform the dimensions of experience samples in the replay buffer to $[m, n\,(d_s + d_a + d_r + d_{s'})]$. Second, the pre-processed experience samples are input into the existing advanced

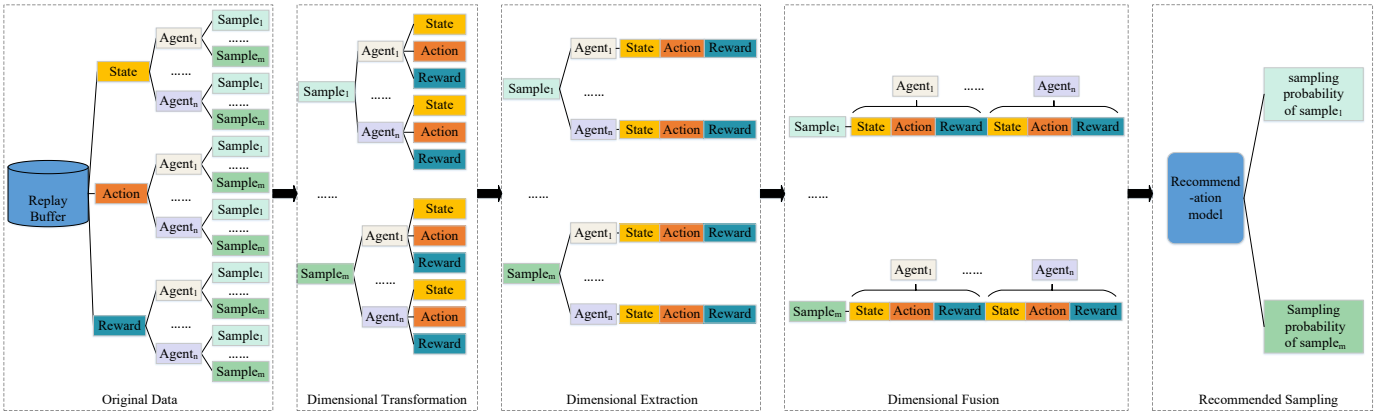


Fig. 2. Experience samples processing.

recommendation model; the model selected in this paper is LightSans, proposed by Microsoft in 2021 [20]. Finally, the recommendation model is used to output the sampling probability of each experience sample. The output form is $[p_1, \ldots, p_m]$, where p_i is the probability that sample$_i$ is recommended. And experience samples of batchsize are selected according to the p_i for updating policy. The processing of the experience samples is shown in Fig. 2.

The experience samples in the replay buffer change dynamically with the training process. To accurately find different efficient experience samples during the entire exploration process and avoid the overfitting phenomenon caused by the recommendation network, which may recommend the same sample multiple times, this paper constructs a reinforcement learning process for the recommendation system based on the original MARL process. In this process, the recommendation model is defined as agent$_{rec}$, whose learning policy is to recommend efficient experience samples. In MARL, the experience samples of batchsize(defined as b_n) recommended by agent$_{rec}$ are used by the multi-agent system to update policy every time step T. Agent$_{rec}$ updates policy with the same time step as T and before recommending experience samples.

The state of agent$_{rec}$ is the feature of the experience sample, and the state space can be represented as $s_{rec} = \left\{cat(s_k, a_k, r_k, s_k^{'})_{k\in n}\right\}$, where cat represents the concatenation of state, action, reward, and next state features for all agents. The action of agent$_{rec}$ is to output the recommendation probability based on the features of the experience sample, and the action space can be represented as $a_{rec} = \left\{(p_i)_{i\in m}\right\}$, where m represents the number of experience samples existing in the replay buffer. The environment of agent$_{rec}$ is the training process of the MARL algorithm. The environment reward of agent$_{rec}$ is the cumulative reward for interacting with the environment after updating the policy network using the experience samples recommended by agent$_{rec}$. Assuming the last time that the multi-agent system updated policy was T_1, the reward of agent$_{rec}$ can be expressed as $r_{rec} = \sum_{t=T_1}^{T_1+T}\sum_{d=1}^{n} r_d^t$.

In the whole training process, the input of agent$_{rec}$ is the feature of each experience sample with dimension $n\,(d_s + d_a + d_r + d_{s'})$, and the output of agent$_{rec}$ is the sampling probability of each experience sample with dimension 1. Therefore, the change of the number m will not affect agent$_{rec}$. The RR process is shown in Fig. 3.

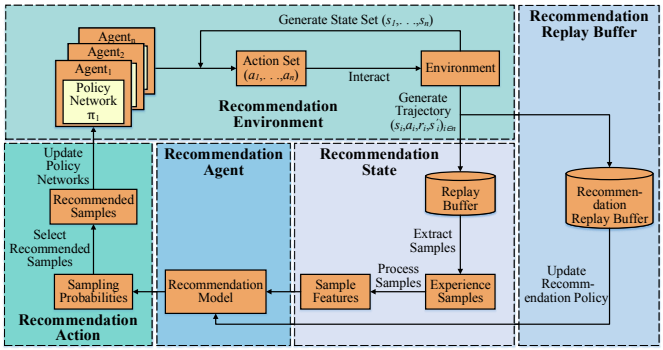


Fig. 3. Reinforcement recommendation process.

The policy π_θ of agent$_{rec}$ needs to learn to recommend the most suitable experience sample for updating the policy at the current stage according to the experience sample existing in the replay buffer to maximize the expected cumulative rewards of the multi-agent system. The objective function of agent$_{rec}$ can be expressed as follows:

$$\nabla_\theta J(\theta) = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\hat{A}_t\right] \tag{1}$$

where $\hat{A}_t$ is the estimator of the dominance function at time step t and $\hat{\mathbb{E}}_t$ is the expected value at time step t. According to the importance sampling formula:

$$\mathbb{E}_{x\sim p}\left[\frac{q(x)}{p(x)}f(x)\right] = \int \frac{q(x)}{p(x)}f(x)p(x)dx = \mathbb{E}_{x\sim q}[f(x)] \tag{2}$$

Using the importance sampling principle, the policy distribution is converted to be estimated at $\pi_{\theta_{old}}$, and the original policy gradient can be rewritten as follows:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[\nabla \log \pi_\theta\left(s_t \mid a_t\right)\hat{A}_t\right] \\ &= \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\nabla \log \pi_\theta\left(s_t \mid a_t\right)\hat{A}_t\right]\end{aligned} \tag{3}$$

To prevent the large deviation of the target value estimation caused by the large difference between the new and old policies, the clip operation is used to suppress the change range of the new policy to reduce the deviation of the target value estimation. The final loss function after introducing the clipping operation can be expressed as follows:

$$L^{clip}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta), \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\right)\hat{A}_t\right] \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the ratio of the old and new policies, which is used to measure the difference between the old and new policies. $\pi_{\theta_{old}}$ represents the old

policy before update, and we use $\pi_{\theta_{old}}$ and its sample distribution to approximately estimate the new policy π_θ according to the principle of importance sampling. The clip function represents that the policy difference is controlled within the range $(1-\epsilon, 1+\epsilon)$ to limit the update range of policy gradient and policy network parameters and ensure that the sample update using the previous policy sampling is effective.

B. Group reward modification

Many real-world tasks make it challenging for multi-agent systems to develop effective policies due to a lack of effective learning information and scarce incentives. A common approach to solving the sparse reward problem is to design dense individual rewards to guide the agent into learning policy. For example, to make the team learn the policy to win in a football game, a series of dense individual rewards are designed for each player so that each player can learn the scoring policy (shooting, catching the ball, etc.) and cooperation policy (passing, etc.). According to individual rewards, the entire team can learn the group policy to win the game.

However, introducing individual rewards to guide policy learning may change the goal or motivation of the agent or even produce unexpected behaviors that deviate from the group policy. Using the above team as an example, the policy that the agents should learn is to score. If the individual reward design is too dense, it is likely to change the learning objective of the agent so that the agent learns a policy that focuses on achieving the above actions rather than scoring. For example, if the reward design for shooting is too dense, the agent will focus on shooting to obtain a high reward, which will affect the agent's original learning goal. Therefore, we propose a method of group modification to guide the agent's policy learning by constraining the individual reward and group contribution to realize the consistency of policy learning.

Suppose that in a multi-agent system, agent$_i$ uses a policy network with parameters θ_i to learn an individual policy. If $R_i(\tau) = \sum_{t=0}^{T} r_i\left(s_i^t, a_i^t\right)$ represents the expectation of the exploration trajectory τ obtained by policy π_i, then the policy gradient of agent$_i$ can be expressed as follows:

$$\nabla_\theta J\left(\pi_\theta^i\right) = \sum_{\tau \sim \pi_\theta^i} R_i^t\left(s_i^t, a_i^t\right) \nabla_\theta \log \pi_\theta\left(a_i^t \mid s_i^t\right) \tag{5}$$

Where $R_i^t\left(s_i^t, a_i^t\right) = \sum_{t'=t}^{T} \gamma^{t'-t} r_i\left(s_i^{t'}, a_i^{t'}\right)$ is the discounted future reward for action at time *t*. The discounting factor γ is applied to reduce variance in the gradient estimate. Group modification adjusts individual learning goals based on the multi-agent system rewards. The policy π_i of each agent$_i$ must adjust its behavior based on the current group reward to produce action with higher group rewards, with the ultimate goal of maximizing the cumulative expected discount group reward $\sum_{i=1}^{n} \sum_{t=1}^{\infty} \gamma^t r_i^t$. Assuming that the ratio of the individual reward of the agent to the average reward of the group is defined as the policy deviation μ, the policy deviation μ_i^t of the exploration trajectory $\left(s_i^t, a_i^t, r_i^t, s_i^{'t}\right)$ of the agent$_i$ at time *t* can be expressed as follows:

$$\mu_i^t\left(r_i\right) = \frac{r_i\left(s_i^t, a_i^t\right) \times n}{\sum_{k=1}^{n} r_k\left(s_k^t, a_k^t\right)} \tag{6}$$

The policy deviation being equal to 1 indicates that the individual and group policies are in the same direction and that the learning of the individual policy should be accelerated. However, when the policy deviation is different from 1, it indicates that there is a conflict between the individual and group policies, and the individual policy should be updated carefully to avoid deviating too far from the group policy. Based on the policy deviation of the agent, using MAAC [8] as an example, the new objective function of the agent regarding team cooperation can be expressed as follows:

$$J^{GM}\left(\pi_\theta^i\right) = \mathbb{E}[min(\mu_i, clip(\mu_i, 1-\epsilon, 1+\epsilon))J(\pi_\theta^i)] \tag{7}$$

where μ_i represents the policy deviation of agent$_i$, π_θ^i represents the individual policy of agent$_i$, ϵ represents the change coefficient that controls the update range of the policy deviation, and $J\left(\pi_\theta^i\right)$ represents the objective function in the MAAC algorithm. The basic idea of GM is to constrain the updating of the individual policy π_θ^i of agent$_i$ to within the range of the policy deviation μ_i. When there is a large difference between the individual reward and the group reward of agent$_i$, i.e., μ_i is not in the range $(1-\epsilon, 1+\epsilon)$, which indicates that the individual policy may focus on learning an action to obtain rewards rather than learning an action to complete the team task, then the policy network parameter of the cumulative expected discounted reward is trimmed so that the individual policy π_θ^i changes slightly.

At the early stage of MARL, individual and group policies may differ greatly. If Eq. 7 is used to guide the agent to perform policy learning, the policy network π_i may not be updated effectively due to the excessive clipping of parameter π_i by the clip function. To ensure the effective update of π_i, the Kullback-Leibler (KL) divergence controlled by the decreasing coefficient α is used to control the distance between individual and group reward, and the agent is constrained to gradually prefer the policy with higher group reward while learning the individual policy. The final objective function of agent$_i$ after introducing the KL divergence is as follows:

$$\begin{aligned} J^{GM}\left(\pi_\theta^i\right) = \mathbb{E}[&min(\mu_i, clip(\mu_i, 1-\epsilon, 1+\epsilon))J(\pi_\theta^i) \\ &-\alpha KL\left(r_i\left(s_i^t, a_i^t\right), \frac{\sum_{k=1}^{n} r_k\left(s_k^t, a_k^t\right)}{n}\right)] \end{aligned} \tag{8}$$

Where $r_i\left(s_i^t, a_i^t\right) = \left[r_i\left(s_i^{T_1}, a_i^{T_1}\right), \ldots, r_i\left(s_i^{T_1+T}, a_i^{T_1+T}\right)\right]$, $\frac{\sum_{k=1}^{n} \mathrm{r_k}\left(s_k^t, \mathrm{a_k^t}\right)}{n} = \left[\frac{\sum_{k=1}^{n} \mathrm{r_k}\left(s_k^{T_1}, \mathrm{a_k^{T_1}}\right)}{n}, \ldots, \frac{\sum_{k=1}^{n} \mathrm{r_k}\left(s_k^{T_1+T}, \mathrm{a_k^{T_1+T}}\right)}{n}\right]$. T_1 represents the time of the last policy update, and T represents the policy update step(how many steps the policy is updated). In this update stage, the KL-divergence can be used to calculate the difference between individual and group rewards, which can then be utilized to constrain the individual policy.

C. Algorithm implementation and complexity analysis

The MARRGM framework uses the recommendation model instead of manual sampling and constructs the RL process to learn the recommendation policy to make efficient use of experience samples in the replay buffer and improve the learning speed of agents. It also uses group rewards to modify individual policies and make them consistent with team policies, which helps improve the ability to learn the optimal solution. The pseudocode for the MARRGM framework is shown in Algorithm 1. To visually represent the change in time complexity of MADRL algorithms after the combination of the

framework proposed in this paper, we assume that the training episodes, steps per episode, and steps per update are i_e, t_e, and T, respectively; the time to initialize the environment per episode is R_e; the time to interact once with the environment is E_i; the sampling time of the multi-agent system is S; and the time to update the policy network of the multi-agent system is U_p. The time complexity of the original algorithm can be expressed as follows:

$$\mathrm{O}\left(i_e \times R_e + i_e \times t_e \times E_i + \frac{i_e \times t_e}{T} \times (U_p + S)\right)$$

Combining the RR system increases the time required to sample a multi-agent system. Assuming that the number of experience samples in the replay buffer is B_n, the time to process an experience sample into sample characteristics is C_s, the time for the recommendation network to work is R_t, and the time to update the recommendation policy network is U_r, the change in sampling time can be denoted as follows:

$$S \longrightarrow B_n \times C_s + R_t + U_r$$

Assuming that the time taken to modify the policy network using policy deviation is S_m, the time complexity of the algorithm when combined with MARRGM can be denoted as follows:

$$\mathrm{O}\left(i_e \times \left(R_e + t_e \times \frac{E_i T + U_p + B_n \times C_s + R_t + U_r + S_m}{T}\right)\right)$$

Algorithm 1 MADRL algorithms combined with the MARRGM framework.

Input: environment E with n agents; train episodes i_e; steps per episode t_e; steps per update T; batch size B; recommendation network RN; replay buffer D; recommendation replay buffer RD; list of policy deviations PDL.

Output: optimal policies of multi-agent systems for specific tasks.

Initialize replay buffer D, recommendation replay buffer RD, and recommendation network RN;
for episodes=1...i_e **do**
  Reset environment and obtain the initial set of observations $\{s_1, s_2, \ldots, s_n\}$ for the multi-agent system;
  for steps=1...t_e **do**
    Select actions for each agent with MADRL algorithms and obtain the action set $\{a_1, a_2, \ldots, a_n\}$;
    Let each agent interact with the environment and gain experience samples (s_i, a_i, si', r_i);
    Store experience samples in D and RD;
    Calculate policy deviations μ_i of each agent using equation 6;
    Update the list of policy deviations PDL;
    if steps=T **then**
      Update recommendation policy of RN based on RD;
      Process experience samples to obtain sample features;
      Input sample features into RN to obtain the sampling probability of each sample;
      Select experience samples of batch size B in D according to sampling probability;
      Update the policy network of each agent with MADRL algorithms;
      for i =1...n **do**
        Modify policy network of agent i according to the policy deviation μ_i using equation 8;
      end for
      Update the list of policy deviations PDL;
    end if
  end for
end for

III. Experiment and Analysis

In this section, we measure the performance of a multi-agent system combined with MARRGM in terms of the average episode reward for the agent. In addition, we implement a recommendation module called NU-RR without a reinforcement framework to verify the improvement of the reinforcement framework for performance. First, we compare the performance of the MAAC algorithm with the combination of NU-RR, RR, GM, and MARRGM in three environments: cooperative collection, command movement, and target navigation. Next, we extend MARRGM to the MADDPG algorithm and verify the scalability of this framework in the target navigation environment. Then, we validate the universality of RR, GM, and MARRGM in the target navigation environment with a different number of agents. In addition, we record the number of collisions by the agent in the target navigation environment to verify whether the agent has better adaptability to the components of the reward while getting a high reward. Finally, we present the time-reward and time-collision results in the target navigation environment to verify the improvement of the learning speed of the agent.

A. Environment description and parameter setting

The cooperative collection environment: There are eight agents, six of which are collectors, and two are collection warehouses. The task of the collector is to collect randomly generated goods of all colors in the environment and deposit them in a collection warehouse of the same color; the task of the collection warehouse is to collect as many goods as possible. Collectors receive global rewards when they successfully collect goods. All agents receive a global reward when the goods are successfully stored.

The command movement environment: There are eight agents, with four speakers and four listeners. At the beginning of each episode, the speaker and listener are randomly paired, and the target position of each listener is randomly generated. The speaker can see the current and target positions of their paired listener and control the movement of the listener by sending a communication message. The listener is unable to see the surroundings or target position and relies on the communication messages of the speaker to move toward the target position. The speaker and listener are rewarded based on the distance between the listener and the target position.

The target navigation environment: There are *N* agents and *N* landmarks. At the beginning of each episode, the positions of the agents and landmarks are randomly generated, and the agents are tasked with moving to the position of the landmarks. The agents are rewarded for minimizing their distance from each landmark and penalized when they collide.

Our comparison experiments are all implemented on Ubuntu 22.04 LTS with a GeForce GTX 3090 GPU. To ensure the

reproducibility and stability of the experimental results, we use the same experimental parameters and network structure (such as the number of internal parameters and linear network layers) as the original algorithm, which can be found at here. Some key parameters are shown in Table I.

TABLE I
PARAMETER SETTING TABLE

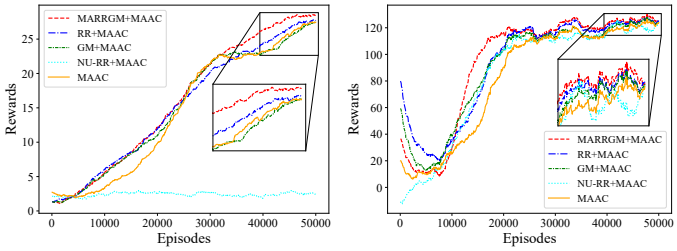
Name	Random seed	Episodes	Episode length	Steps for update	Batch-size	Learning rate
value	0	50000	25	100	1024	0.001

B. Qualitative analysis of experimental results

Figure 4a shows the results in the cooperative collection environment. The combination of MARRGM, RR, and GM has a significant effect on the speed of policy learning during the 0–25K and 35K–50K episodes. The curve combined with RR is lower than the original curve during 25K-35K. This is because RL is a process of continuous trial and error when learning policies. At this stage, the recommendation policy is not beneficial to the learning of the agent due to a lack of exploration, overfitting, and other reasons. However, RR adjusts the policy in 25K–35K episodes and finally learns the policy suitable for different stages after 35K episodes. NU-RR makes the multi-agent system unable to use experience samples for policy learning. We analyze that this is due to the randomness of the initialization of the recommendation model parameters, which makes it impossible to recommend effective experience samples. In addition, since there is no reinforcement framework to update the policy, the recommendation model can only recommend invalid experience samples.

Figure 4b shows the results in the command movement environment. The curves combined with MARRGM and its modules are consistent with the original algorithm in 0–10K episodes. At this time, agent$_{rec}$ cannot find experience samples suitable for the current stage. The curves are greatly improved in the 10K–20K episodes, and the speed of the agent learning policy is greatly accelerated. However, the curve of RR drops and becomes unstable at episodes 20–30K. This is because RR is a process of reinforcement learning, which requires learning the recommendation policy by trial and error. Therefore, the recommended policy is sometimes not suitable for policy learning at the current stage and even hurts policy learning. In the 20K–30K stage, RR continuously updates the recommendation policy to ensure that the recommended experience samples are suitable for the current policy learning. NU-RR can improve the performance of the original algorithm to a certain extent, but the effect is not as good as RR. The results show that a recommendation system with a reinforcement framework can better meet the needs of the current multi-agent system to update policy.

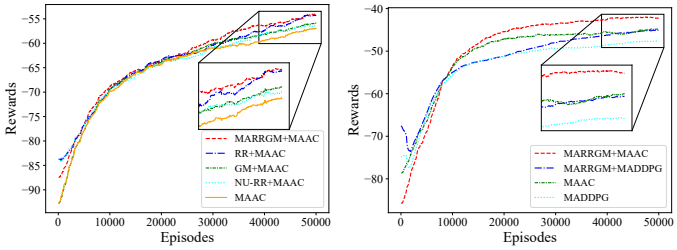
Figure 5a shows the results in the target navigation environment (N=4). The curves combined with MARRGM and its modules are significantly improved after 25K episodes. This shows that after 25K episodes, the recommendation policy can recommend suitable experience samples according to the current stage through the RL process. NU-RR can improve the performance of the original algorithm to a certain extent, but the effect is not as good as RR, indicating that the recommendation model after the introduction of the reinforcement framework is more suitable for the dynamically updated experience samples and the different learning stages. In addition, the use of GM guides the individual policies of agents in group direction, which effectively promotes teamwork.



(a) (b)

Fig. 4. Training results for the cooperative collection environment (**a**) and the command movement environment (**b**).

Figure 5b shows the results after combining different MADRL algorithms with MARRGM in the target navigation environment (N=3). The combination of MARRGM with different MADRL algorithms improves the sample use efficiency to explore the task solution, accelerates the speed of agent learning policies, and enhances the ability to find optimal solutions. The results show that MARRGM is an extensible framework that can be combined with and improves sample use efficiency and the ability to find optimal solutions for most existing off-policy MADRL algorithms.



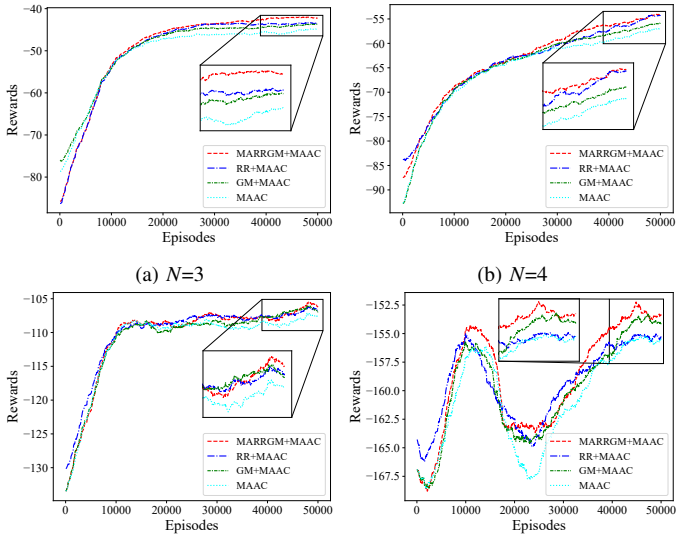
(a) (b)

Fig. 5. Training results for the target navigation environment(N=4) (**a**) and different MADRL algorithms for the target navigation environment(N=3) (**b**).

Figure 6 shows the results in the target navigation environment (N=3/4/10/20). The combination of MARRGM, RR, and GM improves the performance of the MADRL algorithm for any number of agents in the target navigation environment. Since increasing the number of agents may exceed the environmental capacity, landmarks may overlap or be adjacent, which leads to a decrease in the reward curve. We increased the size of the environment proportionally to the number of agents. In all cases, the reward curve and algorithm performance combined with MARRGM and its modules are better than the original algorithm. This shows that MARRGM has a certain degree of robustness and scalability for the number of agents.

The reward obtained by the agent reflects the overall score of the environment. For example, in the target navigation environment, the reward is the combined score of the distance between the agent and the target position and the collision penalty. To verify whether the agent has better adaptability to the components of the reward while getting a high reward, we record the number of collisions with the agent in the target navigation environment. Figure 7 shows the experimental results. It can be seen from the curve that the total number of collisions of the algorithm combined with MARRGM, RR, and GM is lower than that of the original algorithm, and the result

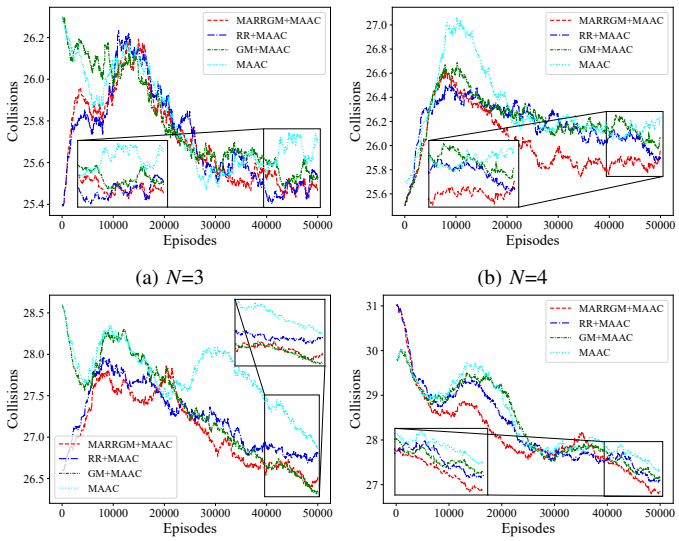
is consistent with the reward definition in this environment. The experimental results show that the proposed method can not only improve the reward obtained but also better adapt to the stealth condition contained in the reward.



(a) N=3 (b) N=4

(c) N=10 (d) N=20

Fig. 6. Training results with different numbers of agents for the target navigation environment.



(a) N=3 (b) N=4

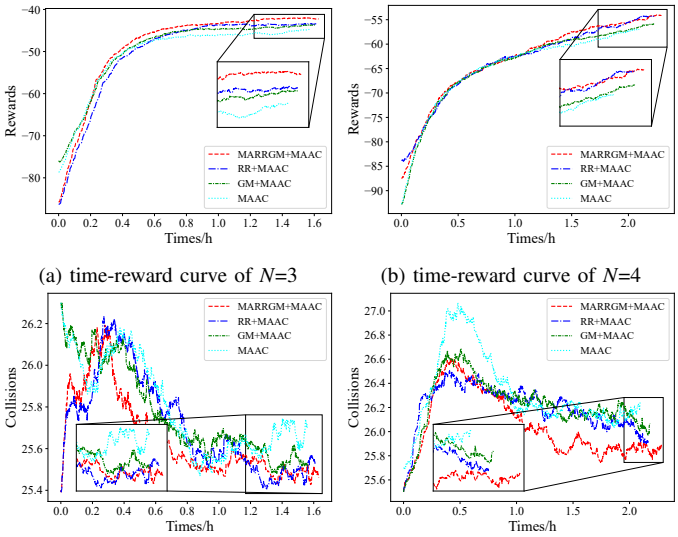
(c) N=10 (d) N=20

Fig. 7. Training results of the number of collisions with different numbers of agents for the target navigation environment.

Figure 8 shows the results of the time curves for the target navigation environment (N=3/4). Although the total time consumption of MARRGM and its modules is higher than that of the original algorithm, the performance of MARRGM and its modules is significantly better than that of the original algorithm at the same time node. The experimental results show that although the combination of MARRGM and its modules will increase the time complexity of the algorithm, it can improve the learning speed of the agent and make the MARL algorithms converge faster.

C. Quantitative analysis of experimental results

First, we combine NU-RR, RR, GM, and MARRGM on the MAAC algorithm in three environments: the cooperative collection, the command movement, and the target navigation environment (N=4) to verify the improvement by each module. The results are listed in Table II. Next, we implement MADDPG combined with MARRGM in the target navigation environment (N=3) to verify the extensibility of this framework. The results are listed in Table III. Finally, to verify the universality of MARRGM for the number of agents and whether the agent has better adaptability to the components of the reward while getting a high reward, we experiment in the target navigation environment with 3, 4, 10, and 20 agents. The results are listed in Table IV.



(a) time-reward curve of N=3 (b) time-reward curve of N=4

(c) time-collision curve of N=3 (d) time-collision curve of N=4

Fig. 8. Training results of time curves for the target navigation environment(N=3/4).

TABLE II

EPISODE REWARDS FOR THE AGENT IN COOPERATIVE COLLECTION, COMMAND MOVEMENT, AND TARGET NAVIGATION ENVIRONMENTS

Environment	Algorithm	Rewards
Cooperative collection	MAAC	15.72
	NU-RR+MAAC	2.43
	RR+MAAC	16.68
	GM+MAAC	16.39
	MARRGM+MAAC	**17.38**
Command movement	MAAC	88.13
	NU-RR+MAAC	91.67
	RR+MAAC	95.51
	GM+MAAC	94.20
	MARRGM+MAAC	**99.01**
Target navigation(N=4)	MAAC	-63.26
	NU-RR+MAAC	-63.07
	RR+MAAC	-62.28
	GM+MAAC	-62.58
	MARRGM+MAAC	**-61.61**

TABLE III

EPISODE REWARDS FOR THE AGENT AND IN THE TARGET NAVIGATION ENVIRONMENT(N=3) FOR DIFFERENT MADRL ALGORITHMS

	MAAC	MARRGM+MAAC	MADDPG	MARRGM+MADDPG
Rewards	-49.56	**-48.86**	-48.94	**-48.25**

D. Algorithm efficiency analysis

To analyze the time complexity of the MADRL algorithm when combined with MARRGM and its modules, we record the time taken by each algorithm to complete the training. The overall complexity of the algorithm is higher than that of the original algorithm when combined with MARRGM, primarily because of the use of a recommendation network to output recommendation samples, the construction of a reinforcement

learning process to learn the recommendation policy, and the use of group modifications to update policies. However, The increase in the time complexity results in an improvement in the algorithm's performance. To exclude the possibility of random exploration interfering with the evaluation results, we use the total number of episodes required to reach the average reward of the original algorithm (Tables II and IV) cumulatively 100 times during training as the evaluation metric. The results are listed in Table V. The results show that although the time complexity of MARRGM is higher than that of the original algorithm, its excellent performance and sample usage efficiency allow the task to be reached in fewer iterations, allowing the MARL algorithm to converge more quickly.

TABLE IV
EPISODE REWARDS AND COLLISIONS FOR THE AGENT IN THE TARGET NAVIGATION ENVIRONMENT WITH DIFFERENT NUMBERS OF AGENTS

The number of agents N	Algorithm	Rewards	Collisions
N=3	MAAC	-48.64	25.77
	RR+MAAC	-47.36	25.72
	GM+MAAC	-47.76	25.74
	MARRGM+MAAC	**-46.70**	**25.71**
N=4	MAAC	-63.26	26.35
	RR+MAAC	-62.28	26.21
	GM+MAAC	-62.58	26.26
	MARRGM+MAAC	**-61.61**	**26.06**
N=10	MAAC	-128.26	27.65
	RR+MAAC	-126.66	27.26
	GM+MAAC	-125.88	27.24
	MARRGM+MAAC	**-125.27**	**27.11**
N=20	MAAC	-210.83	28.33
	RR+MAAC	-203.66	28.10
	GM+MAAC	-202.55	28.24
	MARRGM+MAAC	**-201.42**	**27.87**

TABLE V
THE TRAINING COMPLETION TIMES AND NUMBER OF EPISODES TO REACH THE SPECIFIED REQUIREMENTS FOR EACH ALGORITHM.

Environment	Algorithm	Times	Episodes
Cooperative collection	MAAC	3.96h	27.36K
	RR+MAAC	4.17h	26.47K
	GM+MAAC	4.16h	26.90K
	MARRGM+MAAC	4.46h	**26.26K**
Command movement	MAAC	3.49h	21.65K
	RR+MAAC	3.64h	19.38K
	GM+MAAC	3.58h	19.34K
	MARRGM+MAAC	3.71h	**16.76K**
Target navigation(N=3/4)	MAAC	1.57h / 2.10h	18.17K / 23.03K
	RR+MAAC	1.62h / 2.23h	17.28K / 21.99K
	GM+MAAC	1.61h / 2.23h	16.97K / 21.69K
	MARRGM+MAAC	1.65h / 2.29h	**16.60K / 21.44K**

IV. CONCLUSION

In this paper, we propose a general learning framework called MARRGM, which includes RR and GM modules to improve sample usage efficiency and team policy learning. By comparing it with the baseline algorithm in different environments, it is shown that the proposed framework can improve sample usage efficiency and effectively help MADRL algorithms speed up convergence and find the optimal solution. The results show that this framework is a general learning framework that may benefit other off-policy MADRL algorithms. Our next research direction is to extend this framework to highly complex and dynamically continuous environments to satisfy the requirements for solving real-world problems.

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nat.*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. B. Gjærum, I. Strümke, J. Løver, T. Miller, and A. M. Lekkas, "Model tree methods for explaining deep reinforcement learning agents in real-time robotic applications," *Neurocomputing*, vol. 515, pp. 133–144, 2023.

[3] N. Dorka, T. Welschehold, J. Bödecker, and W. Burgard, "Adaptively calibrated critic estimates for deep reinforcement learning," *IEEE Robotics Autom. Lett.*, vol. 8, no. 2, pp. 624–631, 2023.

[4] Y. Huang, D. Liu, Z. Liu, K. Wang, Q. Wang, and J. Tan, "A novel robotic grasping method for moving objects based on multi-agent deep reinforcement learning," *Robotics Comput. Integr. Manuf.*, vol. 86, p. 102644, 2024.

[5] Z. Zhu, S. Gupta, A. Gupta, and M. Canova, "A deep reinforcement learning framework for eco-driving in connected and automated hybrid electric vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 1713–1725, 2024.

[6] F. Dang, D. Chen, J. Chen, and Z. Li, "Event-triggered model predictive control with deep reinforcement learning for autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 459–468, 2024.

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 6379–6390.

[8] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 2961–2970.

[9] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: duplex dueling multi-agent q-learning," in *ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[10] K. Son, D. Kim, W. J. Kang, D. Hostallero, and Y. Yi, "QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 5887–5896.

[11] T. Wang, T. Gupta, A. Mahajan, B. Peng, S. Whiteson, and C. Zhang, "RODE: learning roles to decompose multi-agent tasks," in *ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[12] I. Liu, U. Jain, R. A. Yeh, and A. G. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 6826–6836.

[13] S. Ding, W. Du, L. Ding, L. Guo, J. Zhang, and B. An, "Multi-agent dueling q-learning with mean field and value decomposition," *Pattern Recognit.*, vol. 139, p. 109436, 2023.

[14] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent deep reinforcement learning with communication," *Auton. Agents Multi Agent Syst.*, vol. 38, no. 1, p. 4, 2024.

[15] J. N. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1146–1155.

[16] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *NIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5048–5058.

[17] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," in *AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 443–451.

[18] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting fundamentals of experience replay," in *ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 3061–3071.

[19] M. Li, T. Huang, and W. Zhu, "Clustering experience replay for the effective exploitation in reinforcement learning," *Pattern Recognit.*, vol. 131, p. 108875, 2022.

[20] X. Fan, Z. Liu, J. Lian, W. X. Zhao, X. Xie, and J. Wen, "Lighter and better: Low-rank decomposed self-attention networks for next-item recommendation," in *SIGIR 21 , Virtual Event, Canada, July 11-15, 2021*. ACM, 2021, pp. 1733–1737.