Value Function Transfer for Deep Multi-Agent Reinforcement Learning Based on N-Step Returns

Yong Liu[1,2] , **Yujing Hu**[2] , **Yang Gao**[1] , **Yingfeng Chen**[2] and **Changjie Fan**[2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Fuxi AI Lab in Netease
lucasliunju@gmail.com, gaoy@nju.edu.cn
{huyujing, chenyingfeng1, fanchangjie}@corp.netease.com

Abstract

Many real-world problems, such as robot control and soccer game, are naturally modeled as sparse-interaction multi-agent systems. Reutilizing single-agent knowledge in multi-agent systems with sparse interactions can greatly accelerate the multi-agent learning process. Previous works rely on bisimulation metric to define Markov decision process (MDP) similarity for controlling knowledge transfer. However, bisimulation metric is costly to compute and is not suitable for high-dimensional state space problems. In this work, we propose more scalable transfer learning methods based on a novel MDP similarity concept. We start by defining the MDP similarity based on the N-step return (NSR) values of an MDP. Then, we propose two knowledge transfer methods based on deep neural networks called *direct value function transfer* and *NSR-based value function transfer*. We conduct experiments in image-based grid world, multi-agent particle environment (MPE) and *Ms. Pac-Man* game. The results indicate that the proposed methods can significantly accelerate multi-agent reinforcement learning and meanwhile get better asymptotic performance.

1 Introduction

Multi-agent reinforcement learning (MARL) is an important learning technique for solving sequential decision-making problems with multiple agents. Recently, with the success of deep reinforcement learning (DRL) [Mnih *et al.*, 2015; Mnih *et al.*, 2016; Schulman *et al.*, 2017], the combination of deep learning models and multi-agent reinforcement learning has also been widely studied.

Current deep multi-agent reinforcement learning algorithms mainly focus on learning in the multi-agent systems where agents are tightly coupled [Sukhbaatar *et al.*, 2016; Foerster *et al.*, 2018; Sunehag *et al.*, 2018; Rashid *et al.*, 2018]. However, many multi-agent systems in the real world have sparse interactions between agents, which means that the agents are independent and don't have to consider the other agents' impact in most situations. Exploiting the sparseness of the interactions between agents can significantly improve the performance of multi-agent reinforcement learning (MARL). Earlier work focuses on constructing coordination graphs (CGs) to represent the interactive relationship between agents and conducting learning according to the specified relationship in each state [Guestrin *et al.*, 2002a; Guestrin *et al.*, 2002b; Kok and Vlassis, 2004]. However, CG-based approaches can only be used in cooperative tasks. Instead of relying on coordination graphs, recent work focuses on automatically identifying interaction areas during the learning process and can be applied to more general learning tasks [De Hauwere *et al.*, 2010; Hauwere *et al.*, 2011; Kok *et al.*, 2005; Melo and Veloso, 2009; Melo and Veloso, 2011; Yu *et al.*, 2015]. In some situations, agents may have already learned some single-agent knowledge (e.g., local value function, local policy) in the same or similar scenarios and appropriately utilizing such knowledge can help to learn better policies faster in multi-agent systems. In this paper, we try to transfer single-agent knowledge to multi-agent environment.

The key to efficiently reutilizing the learned knowledge is Markov decision process (MDP) similarity, which defines the difference between the environmental dynamics of different MDPs and determines whether and how the knowledge can be transferred. A commonly used technique for defining MDP similarity is bisimulation metric [Hu *et al.*, 2015; Song *et al.*, 2016; Ferns *et al.*, 2004], which exactly captures the property of rewards and state transitions in an MDP. However, the fundamental drawback of bisimulation metric is the corresponding high computational cost and inapplicable for high- dimensional state space problem.

In this paper, we propose a novel concept of MDP similarity for identifying the interaction areas of a multi-agent system and propose scalable knowledge transfer methods for deep multi-agent reinforcement learning. Firstly, instead of relying on bisimulation metric, we propose to quantify the environmental dynamics of an MDP by the N-step return (NSR) values, based on which we define a novel concept of MDP similarity. Secondly, based on the novel concept of MDP similarity, we propose two novel knowledge transfer methods called *direct value function transfer* and *NSR-based value function transfer*. Thirdly, experiments are conducted in grid-world, multi-agent particle environment (MPE) and a well-known video game called *Ms. Pac-Man*. The results show that our methods can significantly improve learning efficiency and meanwhile get better asymptotic performance.

2 Background

In this section, we review key concepts in multi-agent reinforcement learning and briefly introduce related work.

2.1 MDP and Sparse Interactions

We start by reviewing the concept of Markov decision process (MDP), which is the fundamental model of reinforcement learning (RL).

Definition 1 *A Markov Decision Process is a tuple $\langle S, A, R, T\rangle$, where S is the state space, A is the action space of the agent, $R : S \times A \to \Re$ is the reward function, $T : S \times A \times S \to [0, 1]$ is the transition function.*

One classic reinforcement learning algorithm for solving an MDP is Q-learning [Watkins, 1989], which iteratively approximates the optimal state-action value function of the MDP with a tabular function Q by the following rule:

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

where α is the learning rate, (s, a) is the current state-action pair, r is the immediate reward, and s' is the next state.

Current deep multi-agent reinforcement learning algorithms assume strong coupling between agents, which means that each agent can exchange information (e.g., actions and reward signals) with other agents at each time step. However, in many multi-agent systems, the interactions between agents is sparse. Melo and Veloso present examples of the so-called multi-agent systems with sparse interactions [Melo and Veloso, 2009]. In Figure 1, the agents can walk freely in most areas. However, for each of the narrow doorways, only one robot can pass through it at one time, which means the agents should coordinate around the doorway. In such systems, the interactions between agents do not occur in all states.

2.2 MDP Similarity and Knowledge Transfer

In a multi-agent system with sparse interactions, sometimes agents may have already learned some single-agent knowledge (e.g., local value functions) in the same or similar scenarios before the multi-agent learning process. With such knowledge, there is no need for agents to learn from scratch. For example, Dota game [1] has multi-battle modes. We can learn a policy in 1v1 scenario and try to transfer the policy to 5v5 scenario. The key to solving it is MDP similarity. On one hand, related MDPs may have some common knowledge that can help the target MDP to be solved better. On the other hand, transferring knowledge from MDPs which differ too much from the target MDP may cause negative transfer.

2.3 Related Work

For transfer in multi-agent systems with sparse interactions, previous work mainly concentrated on the tabular domain. Several works [De Hauwere *et al.*, 2010; Hauwere *et al.*, 2011; Hu *et al.*, 2015] investigate how to reutilize available knowledge to improve learning algorithms. The key to the knowledge transfer process is to evaluate the difference between the agents' local environmental dynamics in the previous and current tasks. Knowledge can only be reused in states where the difference is small. For example, De Hauwere *et al.* propose to detect the difference in environmental dynamics from statistics of immediate rewards [De Hauwere *et al.*, 2010] and extend the detecting method by using future rewards [Hauwere *et al.*, 2011]. Hu *et al.* formally defines the difference as MDP similarity based on bisimulation metric [Ferns *et al.*, 2004] and propose two knowledge transfer methods for game theory-based MARL algorithms.

[1] http://www.dota2.com/

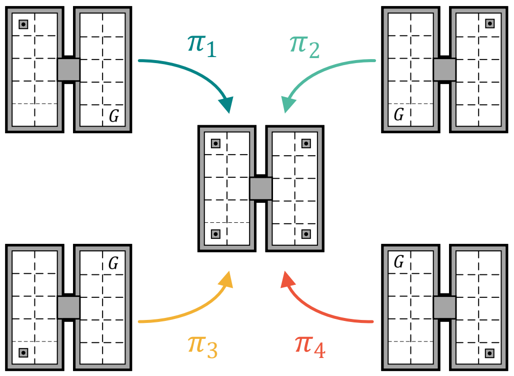


Figure 1: Transfer single-agent knowledge to multi-agent system

Recent work about multi-agent reinforcement learning has started moving from tabular-based method to deep learning-based method. CommNet [Sukhbaatar *et al.*, 2016] uses a centralized network to achieve communication between agents. BiCNet [Peng *et al.*, 2017] uses bidirectional RNNs to communicate with others. These methods need an individual reward for each agent. VDN [Sunehag *et al.*, 2018] learns a joint action-value function with a shared reward which is decomposed into a sum of individual agent terms. QMIX [Rashid *et al.*, 2018] tries to use a hypernetwork to predict the weights of each agent individual state-action value in the joint action-value function. In this paper, we propose novel transfer algorithms based on VDN and QMIX.

3 Our Method

In this section, we propose new transfer approaches which reutilize single-agent knowledge in multi-agent systems with sparse interactions.

3.1 Direct Value Function Transfer

The simplest way of transferring source task knowledge is to directly transfer all state-action values or policy to multi-agent environment. [Hu *et al.*, 2015] proposed *value function transfer* (VFT), which uses each agent's local value function to initialize joint state-action value functions in multi-agent environment. However, the method is only suitable for tabular environments and cannot be extended directly with deep neural networks. This is because the values of all state-action pairs are encoded in the neural network parameters, which means that they are not independent of each other and cannot be accessed as easily as in the tabular case. In addition, we can't directly initialize the multi-agent network by the op-

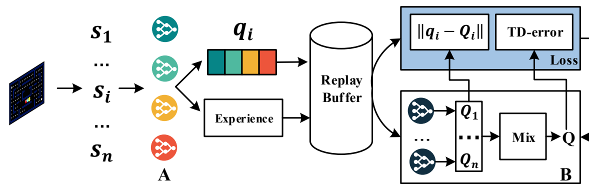


Figure 2: Direct Value Function Transfer Network Architecture. Model A represents single-agent expert policy network and model B represents multi-agent network.

timized single-agent networks since their network structures are different. Inspired by policy distillation, we firstly propose a transfer method called *direct value function transfer* (DVFT), which trains a pre-trained multi-agent network to fit the single-agent knowledge.

Suppose we have n expert policies (one for each agent) which are trained in single-agent source tasks. Under this setting, the multi-agent model can be pre-trained with the single-agent expert policies in the multi-agent environment. As shown in Figure 2, for any global state s, the local state s_i contained in s is input to the network of each agent i. By interacting with the environment, we can obtain the experience $(s_i, a_i, r_i, s_i', done)$ in source tasks. By forward calculation, $q_i(s_i, a_i)$ can be obtained from the existing expert network. Then we can mix them to store in the memory buffer. In this way, the multi-agent network can train joint $Q(s, \vec{a})$ value by sampling data from the memory buffer. Since the basic method is VDN and QMIX, we will obtain individual value function $Q_i(s_i, a_i)$ for each agent i in these methods which represents the joint policy for agent i in multi-agent environment. By utilizing individual value function $Q_i(s_i, a_i)$, we design a novel loss function (line 16 in Algorithm 1), which contains two components: policy loss which aims to output a similar policy to the single-agent expert policy and TD-error loss which aims to train the joint action-value function.

3.2 NSR-based Value Function Transfer

Direct value function transfer provides an appropriate way of network initialization for multi-agent reinforcement learning. However, the method has many limitations. First, it transfers all state-action values, which may cause negative transfer. Second, it needs to learn in the entire state space, which may cause performance degradation since the output of multi-agent model may change in the states where a single-agent source policy performs well. In this section, we propose a novel value function transfer algorithm based on a novel MDP similarity concept. By dividing the state space, the agents just need to learn in the interaction areas and can directly use the single-agent source policy in most time. Three main ideas underly our method: 1) using N-step return to measure MDP similarity, 2) dividing the state space and narrowing down the state space which the agents need to learn, 3) selective transfer and avoiding negative transfer.

First, we should find appropriate representation which summarizes the local environmental dynamics related to each state in an MDP. Note that in previous work, bisimulation metric defines the distance between any two probability dis-

Algorithm 1: Direct Value Function Transfer

```
Input: local value function $q_i(s_i, a)$ for each agent $i$, discount
       factor $\gamma$, exploration factor $\epsilon$
1  Initialization. $Q(s,\vec{a}) \leftarrow \theta$, $\hat{Q}(s,\vec{a}) \leftarrow \theta'$, $Q_i(s_i,\vec{a}) \leftarrow \theta_i$;
2  foreach episode do
3      Initialize state $s$;
4      repeat
5          foreach agent $i$ do
6              $(s_i, a_i) \leftarrow$ the component of agent $i$ in $(s, \vec{a})$;
7              $a_i \leftarrow$ max $q_i$ index with $\epsilon$-greedy policy;
8          $\vec{a} \leftarrow [a_1, ..., a_n]$;
9          store experience $(s, \vec{a}, r, s', done, [q_1, ..., q_n])$;
10         $s \leftarrow s'$;
11         Sample training experience from buffer;
12         if not done then
13             $y = r + \gamma \hat{Q}_{max}(s, \vec{a}; \theta')$;
14         else
15             $y = r$;
16         $L(\theta) = \alpha \sum_1^N (q_i(s_i, a_i) - Q_i(s_i, a_i; \theta_i))^2 + (1 - \alpha)(y - Q(s, \vec{a}; \theta))^2$;
17         Update $\theta$ by a gradient method w.r.t. $L(\theta)$;
18         Every C steps reset $\hat{Q} = Q$;
19     until $s = terminal$;
```

tributions. However, this causes high computational cost and cannot be directly used in complicated environments (e.g., image-based video game). We propose a novel concept for modeling local environmental dynamics which is computationally efficient and can be easily combined with deep neural networks for solving large-scale problems. Our idea is to model the local environmental dynamics of any state s_i using some variables only related to that state. A straightforward solution is the immediate rewards in each state, which is adopted in CQ-learning algorithm [De Hauwere *et al.*, 2010]. However, the immediate rewards do not contain sufficient information since the delayed future rewards and state transitions are also an important part of the environmental dynamics. Thus, one may directly consider using the long-time accumulative rewards to represent the environmental dynamics. But here comes a paradox. The long-time accumulative rewards are the value functions of the agents. If the value functions of two MDPs are learned, there seems no need to compute the MDP similarity for knowledge transfer. In fact, both the immediate rewards and the long-time accumulative rewards are special cases of N-step return with the step number $N = 1$ and $N = \infty$.

Definition 2 **(N-Step Return)** *Let $M_i = \langle S_i, A_i, R_i^l, T_i^l \rangle$ be an MDP of agent i and π_i be learned policy of agent i in M_i. Suppose the state visited at time step t (t $\geq$ 0) is s_i. For a given step number N (N $\geq$ 1), the N-step return of s_i at time t under policy π_i is $\mathcal{R}_{\pi_i,t}^N(s_i) = \left(\sum_{k=0}^{N-1} \gamma^k r_{t+k} \middle| s_t = s_i, \pi_i\right)$, where r_{t+k} is the reward at time step $(t + k)$.*

It is natural to consider the N-step returns with a step number N larger than 1. Due to the discount rate γ, the rewards sampled at larger time steps must contribute less to the N-step return value $\mathcal{R}_{\pi_i,t}^N(s_i)$ than the rewards sampled at time

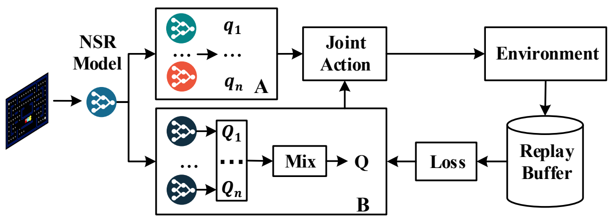


Figure 3: NSR-based Value Function Transfer Network Architecture. Model A represents single-agent expert policy network and model B represents the multi-agent network.

steps closer to t. Therefore, if an appropriate step number N is chosen, the corresponding N-step returns may contain the most information about the local environmental dynamics.

Let $M = \langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T\rangle$ be a Markov game. Assume that the state space S can be factored as $S = \times_{i=1}^n S_i$, where S_i stands for agent i's local state space. Previous single-agent task can be modeled by an MDP $M_i = \langle S_i, A_i, R_i^l, T_i^l\rangle$. A naive way of defining MDP similarity using N-step return (NSR) would be to learn the NSR model separately in source single-agent environment M_i and a virtual target $\hat{M}_i$ [2]. And then comparing the state-wise difference between these models:

$$\mathcal{D}_{M_i,\hat{M}_i}(s_i) = |\mathcal{R}^N_{\pi_i,t}(s_i) - \hat{\mathcal{R}}^N_{\pi_i,t}(s_i)|, \qquad (2)$$

where $\hat{\mathcal{R}}^N_{\pi_i,t}(s_i)$ is the NSR model in multi-agent environment. However, such an approach needs to learn an NSR model in each source single-agent environment, which is not reasonable and increases the burden of the source task. If we can directly use the results learnt by each individual agent, the problem will be solved. We notice that the single-agent source policy model can output its $q_i(s_i, a_i)$ value (e.g., DQN model). We can use it to approximate the local environment dynamics. On the one hand, the value is easy to obtain, on the other hand, it is not necessary to learn the NSR model in the source task. For the environment in Figure 4, the two agents need to reach its own goal. However, only one agent can pass the doorway at each step. If both the agents reach the doorway at the same time, collision will occur and receiving a very negative reward will be given to each of them. That will cause a large difference between the target-task NSR value and the single-agent source-task Q-value, since there is no collision in a single-agent environment. Therefore, we define the N-step return-based MDP similarity as the difference between the single-agent state-action value and multi-agent environment NSR value under single-agent policy π_i.

Definition 3 (N-Step Return-Based MDP Similarity) *Let $M_i = \langle S_i, A_i, R_i^l, T_i^l\rangle$ and $\hat{M}_i = \langle S_i, A_i, \hat{R}_i, \hat{T}_i\rangle$ be two MDPs of agent i in single-agent environment and multi-agent environment, respectively. Let π_i be an policy of agent i in single-agent M_i. Given a step number N ($N > 1$), denote the expectation of the N-step returns of any state s_i in $\hat{M}_i$ by $\hat{\mathcal{R}}^N(s_i)$ and the state-action value function under π_i in M_i by $q_i(s_i, \pi_i)$. The similarity between M_i and $\hat{M}_i$ in state s_i is $\mathcal{D}_{M_i,\hat{M}_i}(s_i) = |\hat{\mathcal{R}}^N(s_i) - q_i(s_i,\pi_i)|$.*

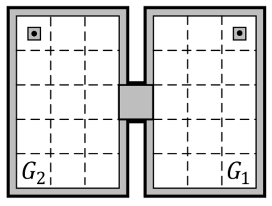
Based on the novel MDP similarity, we propose our value function transfer method, which is shown in Algorithm 2 and Figure 3. Our method first learn an NSR model in the multi-agent environment with a single-agent expert policy. According to the NSR model, we can divide state space into two parts: the states where single-agent knowledge can be transferred and the states where multi-agent learning should be conducted. Specifically, given any local state s_i of agent i and a similarity threshold value τ, the local environmental dynamics of s_i in the single-agent source task is regarded as similar to those in the multi-agent environmental if the corresponding NSR-based MDP similarity $\mathcal{D}_{M_i,\hat{M}_i}(s_i) \le \tau$. In this case, the single-agent policy π_i learnt in the source task can be directly executed in the multi-agent environment. And when $\mathcal{D}_{M_i,\hat{M}_i}(s_i) > \tau$, which means that the local environmental dynamics related to s_i in the single-agent source task and the multi-agent target tasks are very different, multi-agent learning will be conducted. In this way, the state space which the agents need to learn is greatly reduced. The multi-agent learning algorithm just needs to learn in a few states and the model can converge faster.

Algorithm 2: NSR-based Value Function Transfer

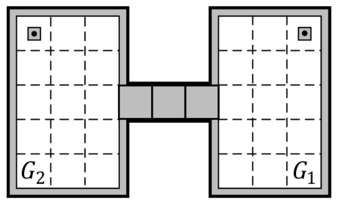
```
Input: local value function $q_i(s_i, a)$ and single-agent policy $\pi_i$
       for each agent $i$, N (N $\ge$ 1), discount factor $\gamma$,
       exploration factor $\epsilon, \delta$
1  Initialization. NSR value function $\hat{\mathcal{R}}^N_{\pi_i} \leftarrow \psi_i$, Replay Buffer
   $\mathcal{B}_i$ for each agent $i$ ;
2  foreach episode do
3      Initialize state $s_t$, $t = 0$;
4      repeat
5          foreach agent $i$ do
6              $(s_{i,t}, a_i) \leftarrow$ the state of agent $i$ in $(s_t, \vec{a})$;
7              $a_i \leftarrow$ max $q_i$ index with probability $\delta$;
8              $y_i = r_{t-N+1} + \gamma r_{t-N} + ... + \gamma^{N-1} r_t$;
9              Store $(s_{i,t-N+1}, y_i)$ in $\mathcal{B}_i$;
10             Update $\psi_i$ by a gradient method w.r.t.
                 $(y_i - \hat{\mathcal{R}}_{\pi_i}(s_{i,t-N+1}; \psi))^2$;
11         $t = t + 1$
12     until $s_t$ = terminal;
13 foreach episode do
14     Initialize state $s$;
15     repeat
16         $(s_i, a_i) \leftarrow$ the state of agent $i$ in $(s, \vec{a})$;
17         if $\max_i \left|\hat{\mathcal{R}}_{\pi_i}(s_i) - q_i(s_i, \pi_i(s_i))\right| \le \tau$ then
18             $a_i \leftarrow$ argmax $q_i(s_i, a_i)$ for each agent $i$;
19         else
20             $a_i \leftarrow$ argmax $Q_i(s, a_i)$ with $\epsilon$ greedy for each
               agent $i$;
21         Learning by multi-agent method;
22     until $s$ = terminal;
```

According to the NSR model, we can measure MDP simi-

[2]The local environment perceived by agent i in M is also modeled by a virtual MDP $\hat{M}_i = \langle S_i, A_i, \hat{R}_i^l, \hat{T}_i^l\rangle$

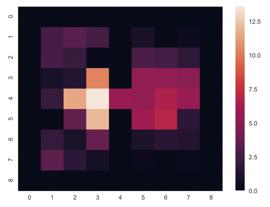


(a) map 1

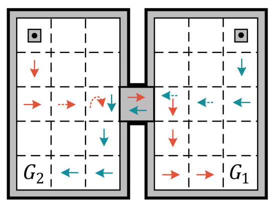


(b) map 2

Figure 4: Grid World environment with two robots. The left is robot 1 and the right is robot 2 in each map.



(a) Heatmap



(b) Policy Display

Figure 5: (a) The heatmap shows the distribution of $D(s_i)$ for robot 1 in map 1. (b) The final learned policy in map 1. The red line is the policy of robot 1 and the green line is the policy of robot 2.

larity between the single-agent and multi-agent environment. In this way, we can identify interaction areas and obtain whether the single-agent policy is appropriate for the current joint state. Hence, we can achieve selective transfer and better avoid negative transfer.

4 Experiments

In this section, we evaluate the performance of our knowledge transfer algorithms in several test scenarios of different complexity. The first one is conducted in benchmarks (grid world games based on image input). The second is multi-agent particle environment (MPE) [Lowe *et al.*, 2017]. The last is conducted in a game called *Ms. Pac-Man* [3].

4.1 Grid World

In the first scenario, two robots must move into a room environment and receive a shared reward. As depicted in Figure 4, we consider two maps with different size. Each agent starts in a corner and the goal is the opposite corner, receiving a reward of 10 for succeed. If both agents end up in the shaded state (the doorway), we think collision occurs and the agents both receive a penalty of -10. Each agent can choose between 5 possible actions (North, South, East, West, Stop).

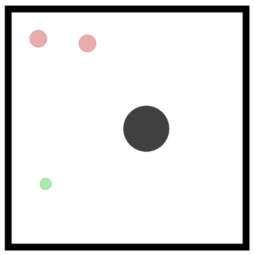
We adapt VDN and QMIX as the basic learning algorithms and implement the proposed two knowledge transfer methods on each of them. The single-agent expert policy is trained by DQN. Figure 7(a-d) exhibits the average reward per episode performance of all tested algorithms in the image-based grid world (The cost of training NSR model and DVFT model is considered in all experiments). It can be found that the two transfer methods perform better than the method which learns from scratch and meanwhile converges faster. In addition, we try to analysis the distribution of distance $D_{M_i,\hat{M}_i}(s_i)$ in each state for robot 1 and we find that the large difference value concentrated on the states near to the doorway. That verifies the effectiveness of NSR-based MDP similarity. As shown in Figure 5(a), the four states which near to the doorway are significantly highlighted and we just need to learn in these states for robot 1. The two agents are heterogeneous and they do not use shared networks. In Figure 5(b), we show the final optimal policy for NSR-VFT. The solid line is represented to transfer policy and the dashed line indicates policy learned by the multi-agent model. We can find the number of states which should learn by multi-agent model is just 2 for robot 1. So, the difficulty of learning is greatly reduced.

4.2 Multi-Agent Particle Environment

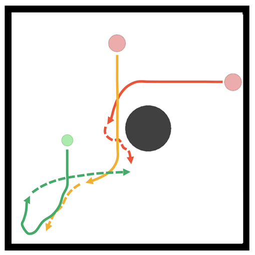
The second scenario in this paper is Multi-Agent Particle Environment. As shown in Figure 6, we choose $predator-prey$ as the test environment, where the adversary agent (red) is slower and needs to capture the good agent (green), the good agent is faster and need to escape. In this paper, we fix the policy (trained by DQN) of a good agent and capturing the good agent is a multi-agent learning task for adversary agents. At each time step, adversary agents receive a reward of -1 except capturing good agent which receives a reward of +10.

Figure 7(e-f) exhibits the average reward per episode of all tested algorithms in MPE. It can be found that the two knowledge transfer algorithms significantly improve the performance of all the tested MARL algorithms. For example in 7(e), the average reward of the basic learning algorithm VDN is below -12. However, with the knowledge transfer algorithms, all of them can finally achieve average rewards around -7 to -12 and the performance of NSR-VDN is better than VFT. That verifies the effectiveness of the NSR transfer mechanism. From the perspective of convergence step performance, the two knowledge transfer methods are significantly faster than the basic algorithms. For example, in Figure 7(f), the basic method QMIX is up to 270000 and is 3 times more than NSR-QMIX method.

In order to analyze the influence of the value of τ on the learning results, the experiments on a different value of threshold τ are performed with the algorithm unchanged in Figure 7(g-h). The τ value of 2, 3, 5, and 7 are used for comparative experiments, respectively. For value of 2, 3, 5, as



(a) predator-prey



(b) Policy Display

Figure 6: Multi-agent Particle Environment

[3]http://ai.berkeley.edu/

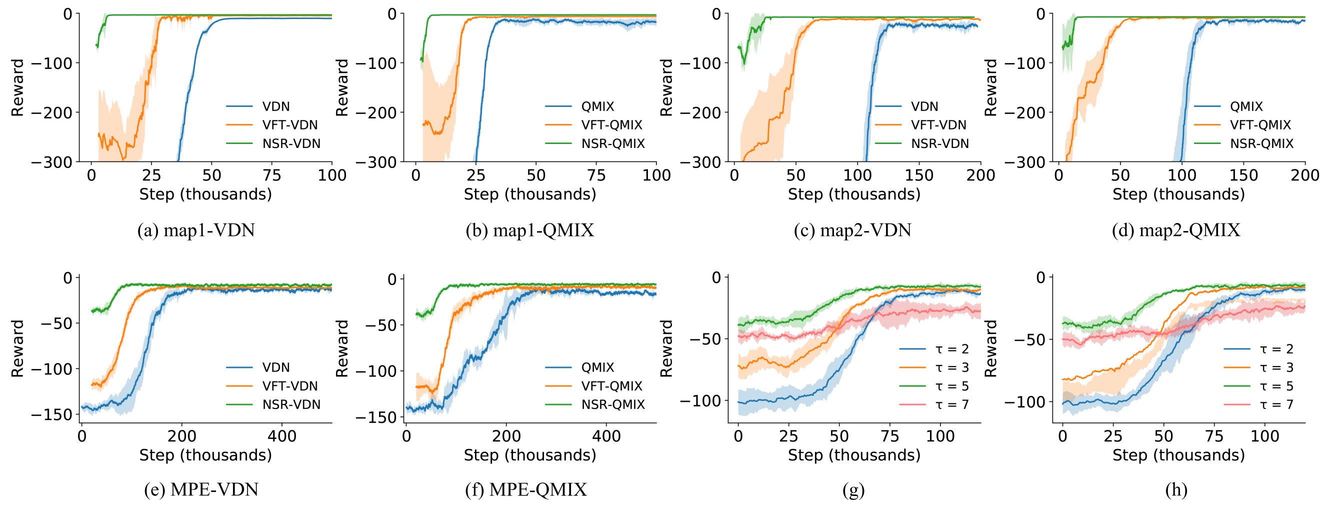


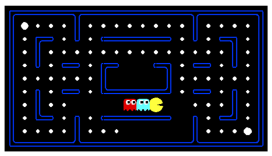
Figure 7: (a-f) is experiment results in Grid World (map1, map2) and MPE, (g) is the influence of τ under VDN in MPE, (h) is the influence of τ under QMIX in MPE. Shaded regions are one standard deviation over 10 runs.

τ value increases, the jump start and final performance gradually increase. That's because more single-agent policy is transferred. However, when the value increases to a certain extent (e.g.,τ = 7), the jump start and final performance is dropped and even cannot learn well. That means the negative transfer occurs. In addition, Figure 6(b) plots the trajectory of the final learned policy, where the solid line represents the policy transferred from single-agent policy and the dotted line represents the learned policy with multi-agent model. We can find that the agents select transfer single-agent policy at the beginning stage because the distance of location between the good agent and adversary agents is far and the adversary agents just need to be close to the good agent. When the distance is close enough, the adversary agents select multi-agent model for learning.

4.3 Ms. Pac-Man

The last experiment scenario in this paper is conducted in a famous video game called *Ms. Pac-Man*. As shown in Figure 8, *Ms. Pac-Man* is a maze game where the goal of the game player is to let the PacMan Scores as many points as possible by eating pills and avoiding the pursuit of the ghosts. For the ghosts in the game, capturing the PacMan is a multi-agent learning task. We choose a pre-trained pacman controller by Deep Q-Network (DQN) to play games.

Figure 8 plots the average reward of all tested algorithms in *Ms. Pac-Man*. we can find the two knowledge transfer approaches significantly improve the final performance and meanwhile learn faster especially for the NSR-VDN method. For example, the average reward per episode achieved by all the basic learning algorithms is below -50 during the learning process. However, with the knowledge transfer mechanism, all of them can finally achieve reward per episode around -20 to -35 and the performance of NSR-VDN is better. As for the perspective of convergence step performance, the convergence steps of NSR-VDN is about 25000. The convergence step of DVFT is 50000 and is at least 2 times more. In contrast, the convergence step of VDN is up to 130000 and is at least 5 times more.



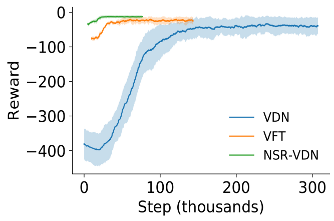


Figure 8: *Ms. Pac-Man* Environment (left), the experiment result in *Ms. Pac-Man* (right).

5 Conclusions

In this paper, we focus on the problem of transferring knowledge in multi-agent systems with sparse interactions. We try to identify the interaction areas by defining a novel concept of MDP similarity in deep multi-agent reinforcement learning domain and propose more scalable knowledge transfer methods. Our major contributions include the novel concept of the N-step return-based MDP similarity, and the two knowledge transfer methods DVFT, NSR-VFT. Experimental results in grid-world, multi-agent particle environment (MPE) and *Ms. Pac-Man* show that with the N-step return-based MDP similarity, the DVFT, and NSR-VFT methods drastically reduce the learning time and meanwhile get better performance.

Acknowledgments

The authors would like to acknowledge support for this project from the National Key R&D Program of China (2017YFB0702600, 2017YFB0702601), the National Natural Science Foundation of China (Nos. 6143200).

References

[De Hauwere *et al.*, 2010] Yann-Michaël De Hauwere, Peter Vrancx, and Ann Nowé. Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 715–722, 2010.

[Ferns *et al.*, 2004] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*, pages 162–169, 2004.

[Foerster *et al.*, 2018] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Guestrin *et al.*, 2002a] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *Proceedings of the 9th International Conference on Machine Learning*, pages 227–234, 2002.

[Guestrin *et al.*, 2002b] Carlos Guestrin, Shobha Venkataraman, and Daphne Koller. Context-specific multiagent coordination and planning with factored MDPs. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 253–259, 2002.

[Hauwere *et al.*, 2011] Yann-Michaël De Hauwere, Peter Vrancx, and Ann Nowé. Solving sparse delayed coordination problems in multi-agent reinforcement learning. In *International Workshop on Adaptive and Learning Agents (ALA 2011)*, pages 114–133, 2011.

[Hu *et al.*, 2015] Yujing Hu, Yang Gao, and Bo An. Learning in multi-agent systems with sparse interactions by knowledge transfer and game abstraction. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 753–761, 2015.

[Kok and Vlassis, 2004] Jelle R. Kok and Nikos A. Vlassis. Sparse cooperative Q-learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, pages 61–68, 2004.

[Kok *et al.*, 2005] Jelle R. Kok, Pieter Jan't Hoen, Bram Bakker, and Nikos A. Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG05)*, pages 29–36, 2005.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[Melo and Veloso, 2009] Francisco S Melo and Manuela Veloso. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 773–780, 2009.

[Melo and Veloso, 2011] Francisco S. Melo and Manuela M. Veloso. Decentralized MDPs with sparse interactions. *Artifitial Intelligence*, 175(11):1757–1789, 2011.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Peng *et al.*, 2017] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4292–4301, 2018.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Song *et al.*, 2016] Jinhua Song, Yang Gao, Hao Wang, and Bo An. Measuring the distance between finite Markov decision processes. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, pages 468–476, 2016.

[Sukhbaatar *et al.*, 2016] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[Sunehag *et al.*, 2018] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[Watkins, 1989] C.J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[Yu *et al.*, 2015] C Yu, M Zhang, F Ren, and G Tan. Multiagent learning of coordination in loosely coupled multiagent systems. *IEEE Transactions on Cybernetics*, 45(12):2853–2867, 2015.