Learning in Multi-agent Systems with Sparse Interactions by Knowledge Transfer and Game Abstraction

Yujing Hu, Yang Gao*
State Key Laboratory for Novel Software Technology, Collaborative Innovation Center of Novel Software Technology and Industrialization
Nanjing University, Nanjing, China
huyujing.yujing.hu@gmail.com, gaoy@nju.edu.cn

Bo An
School of Computer Engineering
Nanyang Technological University
Singapore
boan@ntu.edu.sg

ABSTRACT

In many multi-agent systems, the interactions between agents are sparse and exploiting interaction sparseness in multi-agent reinforcement learning (MARL) can improve the learning performance. Also, agents may have already learnt some single-agent knowledge (e.g., local value function) before the multi-agent learning process. In this work, we investigate how such knowledge can be utilized to learn better policies in multi-agent systems with sparse interactions. We adopt game theory-based MARL as the basic learning approach since it can coordinate agents better. We contribute three knowledge transfer mechanisms. The first one is value function transfer, which directly transfers agents' local value functions to the learning algorithm. The second one is selective value function transfer, which only transfers the value functions in states where the environmental dynamics change slightly. The last mechanism is model transfer-based game abstraction, which further improves the former two mechanisms by abstracting the one-shot game in each state and reducing equilibrium computation. Experimental results in benchmarks show that with the three knowledge transfer mechanisms, all of the tested game theory-based MARL algorithms are drastically improved and also achieve better asymptotic performance than the state-of-the-art algorithm CQ-learning.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

Keywords

Multi-agent Reinforcement Learning, Knowledge Transfer, Multi-agent Systems, Sparse Interactions

*Corresponding Author

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*
Copyright © 2015, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1. INTRODUCTION

Multi-agent reinforcement learning (MARL) has been widely studied in recent years. It is an important approach for solving sequential decision problems in which multiple agents interact with each other and the environment simultaneously. A variety of mathematical models are adopted to represent a multi-agent system (MAS), such as Markov games [16], and decentralized partially observable Markov decision process (Dec-POMDP) [1]. In most of these models, agents are coupled with each other in the joint state space and joint action space. However, such coupling is so tight that it seldom applies in practice since interactions between agents may not happen in all states and always involve all agents. In fact, in many multi-agent systems, the interactions between agents are sparse. Imagine that two intelligent rovers are working together to mine valuable ore on the Mars surface. The small rover is for ore discovery and the big one is for transportation. The interaction between the two rovers occurs only when the small one picks up one valuable stone and hands it to the big one.

Explicitly exploiting sparse interactions in a MAS can greatly improve the performance of multi-agent reinforcement learning. Earlier work makes use of coordination graphs to specify the coordination dependencies between agents in particular states [7, 8, 14]. However, constructing such graphs requires to specify the interaction areas in advance. This motivates the more recent work that learns the interaction areas during the learning process [4, 9, 13, 17, 18]. For example, CQ-learning [4] and FCQ-learning [9] identify the coordinating states from statistical information of rewards and learn an agent-centric representation of the state space. Based on this representation, agents can determine in each state whether to learn independently or jointly.

However, it is still possible to further improve the performance of MARL in a sparse-interaction MAS. Interestingly, we find that in many tasks, agents have already learnt (or are able to learn) some kind of single-agent knowledge (e.g., local value function, local policy) before the multi-agent learning process and utilizing such knowledge during learning may be beneficial. Suppose a housewife buys a robot vacuum cleaner to clean the house. After the robot works for a few months, she decides to buy another robot so that her house can be cleaned more quickly. In this situation, the two robots have no need to learn from scratch

in the two-agent cleaning task since the old robot has learnt some knowledge about the house.

Therefore, in this work, we investigate how such single-agent knowledge can be utilized in multi-agent reinforcement leaning to learn better (joint) policies in multi-agent systems with sparse interactions. The underlying idea is that since the interactions are sparse, each agent only needs to adapt to the slightly changed environmental dynamics based on the knowledge in hand.

We propose to use game theory-based multi-agent reinforcement learning (MARL) [6, 10, 11] as the basic learning approach, which requires agents to play equilibrium strategies in each state. Compared with the Q-learning-like coordinating rule in previous algorithms [4, 17, 18], playing equilibrium strategies is a more better way for coordination since agents may still have conflicts in some states even if they are working cooperatively. We then contribute three knowledge transfer mechanisms to accelerate game theory-based MARL. The first one is value function transfer (VFT), which directly uses the agents' previous local value functions to initialize the value functions of the MARL algorithm. The second mechanism, which is called selective value function transfer (SVFT), conducts value function transfer only in states where the changes of the agents' local environmental dynamics are small. We define similarity between MDPs to model such changes and propose an approach to evaluate the similarity value. Our last mechanism, model transfer-based game abstraction (MTGA), further improves the former two by reducing equilibrium computation. Like SVFT, MTGA also evaluates the changes of the agents' local environmental dynamics. But the evaluated results are used to abstract the one-shot game played in each state.

Experimental results on benchmarks show that with the three knowledge transfer mechanisms, all tested learning algorithms (NashQ [10], CEQ [6], NegoQ [11]) are dramatically improved (e.g., achieving a jump start, better asymptotic performance, and higher total reward) and also achieve better asymptotic performance than the state-of-the-art algorithm CQ-learning [4].

2. BACKGROUND

We review some key concepts in multi-agent reinforcement learning and related work in this section.

2.1 MDP and Markov Games

We start by reviewing the concept of Markov decision process (MDP) as we will use it later. It 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.

The objective of an agent in an MDP is to find an optimal policy which maximizes the expected discounted sum of rewards for each state s at each time step t:

$$V^*(s) = \max_\pi E_\pi\Big\{\sum_{k=0}^{\infty}\gamma^k r^{t+k}|s^t = s\Big\}, \quad (1)$$

where $\pi : S \times A \to [0, 1]$ denotes the policy of an agent, E_π stands for expectation under policy π, $\gamma \in [0, 1)$ is a discount factor, k is a future time step, and r^{t+k} is the immediate reward at the time step $(t+k)$. This goal can be formulated equivalently by

$$Q^*(s, a) = \max_\pi E_\pi\Big\{\sum_{k=0}^{\infty}\gamma^k r^{t+k}|s^t = s, a^t = a\Big\}, \quad (2)$$

where $Q^*(s, a)$ stands for the optimal value of the state-action pair (s, a). An optimal policy can be found by computing the *optimal state-value function* V^* or the *optimal action-value function* Q^*. One classic RL algorithm is Q-learning [24], which iteratively approximates Q^* by a simple updating rule:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')], \quad (3)$$

where Q is the state-action value function, $\alpha \in [0, 1]$ is the learning rate, (s, a) is the state-action pair visited currently, r is the immediate reward given by the environment, and s' is the next state.

Markov game is widely adopted as the model of multi-agent reinforcement learning (MARL) [6, 10, 16]. It can be treated as an extension of MDP to the multi-agent domain.

Definition 2. An n-agent ($n \geq 2$) Markov game is a tuple $\langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T\rangle$, where N is the set of agents, S is the state space, A_i is the action space of agent $i (i = 1, \ldots, n)$. Let $A = A_1 \times \cdots \times A_n$ be the joint action space. $R_i : S \times A \to \Re$ is the reward function of agent i and $T : S \times A \times S \to [0, 1]$ is the transition function.

In a Markov game, the accumulative discounted reward of each agent is determined by the joint policy of all agents. Denote the policy of agent i by $\pi_i : S \times A_i \to [0, 1]$ and the joint policy of all agents by $\pi = (\pi_1, ..., \pi_n)$. The state-action value function of an agent i under a joint policy π can be defined as

$$Q_i^\pi(s, \vec{a}) = E_\pi\Big\{\sum_{k=0}^{\infty}\gamma^k r_i^{t+k}|s^t = s, \vec{a}^t = \vec{a}\Big\}, \quad (4)$$

where $\vec{a} \in A$ denotes a joint action and r_i^{t+k} is the reward received by agent i at time step $(t + k)$. However, since Q_i^π now depends on the actions of all agents, the concept of optimal policy should be replaced with equilibrium policy. To this end, several game theory-based MARL algorithms are proposed [6, 10, 11], which combines equilibrium solution concepts in game theory with multi-agent reinforcement learning. The key idea of these algorithms is to construct a one-shot game in each state, let agents play an equilibrium strategy, and update the value functions according to the computed equilibrium.

2.2 MAS with Sparse Interactions

According to Definition 2, it can be found that both the reward and transition functions are defined on the whole joint state space and joint action space, which means that in every joint state, all agents are coupled with each other. However, such coupling is so tight that it seldom applies in practice. In many multi-agent systems, the interactions between agents are limited in particular areas.

Melo and Veloso [17] present a simple example of the so-called multi-agent systems with sparse interactions. Figure 1 shows a map in which four robots should find a way to their goals. In most areas, the robots can walk freely. However, for each of the narrow doorways (the shaded grids), only one robot can pass through it at one time, which means that the robots should coordinate around the doorway. In this system, the interactions between agents do not occur in all states and do not involve all agents all the time.

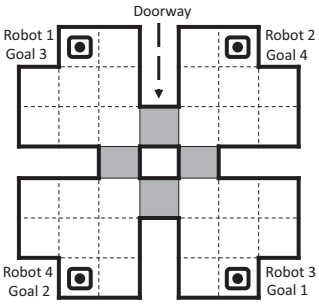


Figure 1: A simple example of multi-agent systems with sparse interactions

2.3 Related Work

Several ways have been investigated to exploit the sparsity of interactions in a multi-agent system. Earlier work adopts coordination graphs (CG) to represent the interactions between agents [7, 8, 13, 14]. For example, based on the coordination graphs constructed beforehand, Kok and Vlassis [14] represent the global value function by a set of local value rules. Each agent only needs to update the value rules in which it is involved. Later, they propose another approach that learns the coordination graphs from experience [13]. However, the CG-based approaches are limited to cooperative tasks where the agents have common interests.

More recently, Melo and Veloso [17] introduced a two-layer extension of Q-learning for exploiting the potential sparse interactions in a Markov game. In this algorithm, each agent should learn in each state whether to act independently or execute the additional action COORDINATE with other agents. De Hauwere et. al propose a similar algorithm CQ-learning [4], which identifies the coordinating states from statistical information of the rewards. Later, CQ-learning was extended to FCQ-learning [9] with an enhanced mechanism for detecting the coordinating states.

As illustrated by the example of robot vacuum cleaners in Section 1, in some multi-agent systems, agents may have learnt single-agent knowledge before the multi-agent learning process. In this next section, we discuss how such knowledge can be utilized to improve the performance of multi-agent reinforcement learning. A few approaches for knowledge transfer in multi-agent systems have been proposed, such as parallel transfer [20], equilibrium transfer [12], and training a classifier to generalize knowledge [23]. In this work, we mainly focus on knowledge transfer in multi-agent systems with sparse interactions and investigate what kind of knowledge and how the knowledge can be utilized.

3. KNOWLEDGE TRANSFER

Most previous algorithms simply use a Q-learning-like rule to coordinate agents [4, 17, 18]. Instead, we adopt game theory-based MARL [6,10,11] as the basic learning approach. The reason is that the equilibrium solution concepts adopted in game theory-based MARL can essentially deal with the potential conflicts between agents (e.g., the doorways in Figure 1) while the Q-learning-like rule cannot.

In general, game theory-based MARL can be generalized as the high-level framework in Algorithm 1. In each state s, a joint action $\vec{a}$ is chosen according to the equilibrium computed by the process Ω (line 5). Here $\Omega(Q_1(s), \ldots, Q_n(s))$ represents computing an equilibrium according to the state-action values of each agent in s. After taking the joint action $\vec{a}$, each agent i receives a reward r_i and the next state s' is observed (line 7). The value function Q_i of each agent i then is updated according to the rule

$$Q_i(s, \vec{a}) \leftarrow (1-\alpha)Q_i(s, \vec{a}) + \alpha(r_i + \gamma\Phi_i(s')), \qquad (5)$$

where $\Phi_i(s')$ is the expected value of the equilibrium in state s' for agent i.

Algorithm 1: The general framework of game theory-based MARL

Input: Learning rate α, discount factor γ, exploration factor ϵ

1 Initialization. $\forall s \in S, \forall i \in N, \forall\, \vec{a}, Q_i(s, \vec{a}) \leftarrow 0$;
2 **foreach** *episode* **do**
3 Initialize state s;
4 **repeat**
5 $\vec{a} \leftarrow \Omega(Q_1(s), \ldots, Q_n(s))$ with ϵ-greedy policy;
  `/* Ω is for computing an equilibrium */`
6 **foreach** *agent* $i \in N$ **do**
7 Receive the experience $(s, \vec{a}, r_i, s')$;
8 $Q_i(s, \vec{a}) \leftarrow (1-\alpha)Q_i(s, \vec{a}) + \alpha(r_i + \gamma\Phi_i(s'))$;
  `/* Φ_i is the expected value of the equilibrium in state s' for agent i */`
9 $s \leftarrow s'$;
10 **until** s *is a terminal state*;

Let $M = \langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T\rangle$ be a Markov game. Like the previous work [4,17], we also assume that each agent $i \in N$ can perceive its local state and the state space S can be factored as $S = \times_{i=1}^n S_i$. As stated in previous sections, before learning in the Markov game M, each agent i may have already learnt some kind of "single-agent knowledge" with respect to its local states in S_i and local actions in A_i, such as a local value function, a local acting policy, and empirical models of the local reward function and transition function. Since the agents can act independently in most of the (joint) states, reusing such single-agent knowledge may accelerate the multi-agent learning process. Inspired by transfer learning [21], in the following, we propose three knowledge transfer mechanisms for reusing different types of single-agent knowledge.

3.1 Value Function Transfer

The most important knowledge learnt by a reinforcement learning algorithm is the value function, which is also used widely as the transferred knowledge in transfer learning [15, 22]. Our first knowledge transfer mechanism is **value function transfer (VFT)**, which directly uses each agent's local value function learnt previously to initialize its value function of joint state-action pairs in the MARL algorithm. The underlying idea is that since the interactions between agents are sparse, the multi-agent system can be treated as an approximation of multiple independent MDPs, each of which corresponds to one agent. In most joint states, the agents can act as they do in the single-agent case according to the transferred value functions. In the states where interaction occurs, the MARL algorithm is able to learn a coordinating policy for them. The corresponding algorithm is shown in Algorithm 2.

After the initialization of the value function Q by each agent i's local value function q_i, Algorithm 2 follows the same process of game theory-based MARL in Algorithm 1. Essentially, the transferred local value functions provide a

Algorithm 2: Value function transfer

Input: Learning rate α, discount factor γ, exploration factor ϵ, local value function q_i for each agent i

Initialize the state-action value function Q as follows;
foreach *agent* $i \in N$ **do**
  foreach *state* $s \in S$ **do**
    foreach *joint action* $\vec{a} \in A$ **do**
      $s_i \leftarrow$ agent i's local state in s;
      $a_i \leftarrow$ agent i's action in $\vec{a}$;
      $Q_i(s, \vec{a}) \leftarrow q_i(s_i, a_i)$;
The following is the same as in Algorithm 1;

relatively better initial acting policy, based on which the MARL algorithm can learn an optimal policy (or rather equilibrium policy) more quickly.

3.2 Selective Value Function Transfer

The first knowledge transfer mechanism VFT directly uses the agents' local value functions to initialize the values of all joint state-action pairs. Although such initialization can lead the agents to choose better actions in most states, however, it may cause a big penalty for the agents in some particular states. For example, in Figure 1, suppose that Robot 2 is in the grid right above the leftmost doorway and Robot 3 is in the grid right under this doorway. In the single-agent case, the best action for both the two robots is to go through the doorway, which makes them closer to their goals. But in the situation described above, if they still perform the same action, collision will occur and neither of them can successfully pass the doorway.

To avoid the "unchecked" initialization in VFT, in this subsection, we propose our second knowledge transfer mechanism, which is called **selective value function transfer (SVFT)**. The key idea of SVFT is that for each agent i, knowledge is transferred only in states where the local environmental dynamics perceived by agent i are similar to those in the single-agent case. Here, the local environmental dynamics of agent i refer to its reward and transition models with respect to its local state space S_i and action space A_i. In the following, we first propose a method for evaluating the changes of the agents' local environmental dynamics and then introduce the details of SVFT.

3.2.1 Evaluating The Changes of Local Environmental Dynamics

For a given Markov game $M = \langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T\rangle$, we can construct each agent's empirical local environment model by conducting Monte Carlo trials with a random policy. For each agent i, let $M_i = \langle S_i, A_i, R_i^l, T_i^l\rangle$ be its empirical local environment model constructed in the Markov game M and let $\hat{M}_i = \langle S_i, A_i, \hat{R}_i^l, \hat{T}_i^l\rangle$ be the MDP model of its previous single-agent task. So the actual problem is to evaluate how the two MDPs are similar in each state in the local state space S_i.

Note that a similar problem has been investigated in CQ-learning [4] and FCQ-learning [9]. In CQ-learning, a local state s_i of agent i is considered as unchanged if no difference between the sampled immediate rewards and the expected rewards in s_i is detected. However, the changes of immediate rewards cannot reflect all information about how the environmental dynamics change. FCQ-learning [9] takes future rewards into consideration and uses a Kolmogorov-Smirnov test (KS-test) to evaluate the changes in Q-values. As reported by the authors, the KS-test cannot identify additional changes caused by small fluctuations in the Q-values.

In an MDP, the states are not independent, but are related to each other through the state transitions. Therefore, the "relationship" between the states and how it changes are the key to accurately evaluating the changes of the environmental dynamics. We adopt the concept of state distance proposed by Ferns et. al [5] to model the relationship between the states in an MDP.

Definition 3. Let $M_i = \langle S_i, A_i, R_i^l, T_i^l\rangle$ be an MDP of agent i. For any two states $s_i, s_i' \in S_i$, the state distance $d_{M_i}(s_i, s_i')$ between s_i and s_i' is defined as

$$d_{M_i}(s_i, s_i') = \max_{a_i \in A_i} \Big\{ |R_i^l(s_i, a_i) - R_i^l(s_i', a_i)| \quad + \gamma \mathcal{T}^K_{(d_{M_i})}(T_i^l(s_i, a_i), T_i^l(s_i', a_i)) \Big\}, \tag{6}$$

where γ is the discount factor, and $\mathcal{T}^K_{(d_{M_i})}(T_i^l(s_i, a_i), T_i^l(s_i', a_i))$ is the Kantorovich distance between the probabilistic distributions $T_i^l(s_i, a_i)$ and $T_i^l(s_i', a_i)$.

In short, the state distance $d_{M_i}(s_i, s_i')$ between s_i and s_i' is their maximal weighted sum of the reward difference and the Kantorovich distance[1] between the state transition probabilities. All information about the environmental dynamics along the path between the two states is summarized into this distance value.

Recall that our goal is to evaluate the similarity between two MDPs in each state. Based on the state distance defined above, we define this concept as follows.

Definition 4. Given two MDPs $M_i = \langle S_i, A_i, R_i^l, T_i^l\rangle$ and $\hat{M}_i = \langle S_i, A_i, \hat{R}_i^l, \hat{T}_i^l\rangle$, for any state $s_i \in S_i$, the similarity between M_i and $\hat{M}_i$ in s_i is defined as

$$\mathcal{D}_{M_i, \hat{M}_i}(s_i) = \sqrt{\sum_{s_i' \in S_i} \left(d_{M_i}(s_i, s_i') - d_{\hat{M}_i}(s_i, s_i')\right)^2}, \tag{7}$$

where $d_{M_i}(s_i, s_i')$ and $d_{\hat{M}_i}(s_i, s_i')$ are the state distances between s_i and s_i' in M_i and $\hat{M}_i$, respectively.

From Eq. 7, it can be found that the changes of the distances between the state s_i and the other states all contribute as a part of the similarity value $\mathcal{D}_{M_i, \hat{M}_i}(s_i)$. That is to say, $\mathcal{D}_{M_i, \hat{M}_i}(s_i)$ reflects how the relationship between states changes as we required. Algorithm 3 summarizes the process of computing the similarities between two MDPs. Compared with the KS-test on Q-values in FCQ-learning [9], the MDP similarities quantify the changes of local environmental dynamics and thus are more flexible. Furthermore, the computing process in Algorithm 3 only runs once.

3.2.2 The SVFT Algorithm

There are mainly three steps in our second knowledge transfer mechanism **selective value function transfer (SVFT)**. Firstly, we construct for each agent i an empirical MDP model M_i by Monte Carlo sampling in the Markov game. Secondly, for each agent i, the similarities between M_i

[1]The Kantorovich metric was originally used in mass transportation theory to find the best total flow for a given network. It exactly captures the property of state transitions in an MDP.

Algorithm 3: Computing the similarities between two MDPs

Input: Two MDP models $M_i = \langle S_i, A_i, R_i^l, T_i^l \rangle$ and $\hat{M}_i = \langle S_i, A_i, \hat{R}_i^l, \hat{T}_i^l \rangle$
Output: The set of similarity values $\mathcal{D}$

1 **foreach** *state* $s_i \in S_i$ **do**
2 $\mathcal{D}(s_i) \leftarrow 0$;
3 **foreach** *state* $s_i' \in S_i$ **do**
4 Compute the state distance $d_{M_i}(s_i, s_i')$ and $d_{\hat{M}_i}(s_i, s_i')$ according to Eq. 6;
5 $\mathcal{D}(s_i) \leftarrow \mathcal{D}(s_i) + (d_{M_i}(s_i, s_i') - d_{\hat{M}_i}(s_i, s_i'))^2$;
6 $\mathcal{D}(s_i) \leftarrow \sqrt{\mathcal{D}(s_i)}$;

and the MDP in its previous single-agent task $\hat{M}_i$ are computed. Note that in many single-agent RL algorithms [2,19], the environment model is also learnt. Lastly, based on the similarities computed in the second step, the local value function of each agent is transferred only in the local states where the similarity is small. The corresponding algorithm is shown in Algorithm 4.

Algorithm 4: Selective value function transfer

Input: Learning rate α, discount factor γ, exploration factor ϵ, local value function q_i and local MDP model $\hat{M}_i$ for each agent i, a threshold value τ, an integer L for Monte Carlo sampling

1 **for** *episode* $= 1, \ldots, L$ **do**
2 Perform Monte Carlo sampling with a random policy, recording the rewards and state transitions;
3 **foreach** *agent* $i \in N$ **do**
4 $M_i \leftarrow$ the empirical MDP model of agent i;
5 $\mathcal{D}_i \leftarrow$ the set of similarities between M_i and $\hat{M}_i$, computed by Algorithm 3;
6 Transfer the local value functions as follows;
7 **foreach** *agent* $i \in N$ **do**
8 **foreach** *state* $s \in S$ **do**
9 **foreach** *joint action* $\vec{a} \in A$ **do**
10 $s_i \leftarrow$ agent i's local state in s;
11 $a_i \leftarrow$ agent i's action in $\vec{a}$;
12 **if** $\mathcal{D}_i(s_i) \leq \tau$ **then**
13 $Q_i(s, \vec{a}) \leftarrow q_i(s_i, a_i)$;
14 The following is the same as in Algorithm 1;

Compared with value function transfer (VFT) in Algorithm 2, the agents' previous MDP models are also required as knowledge in SVFT. The parameter τ is a threshold value for controlling the value function transfer process. For each agent i, value function transfer can be conducted in its local state s_i only if the similarity between M_i and $\hat{M}_i$ in s_i is smaller than τ (lines $12-13$). Note that a smaller similarity value means "more similar" between two MDPs.

3.3 Model Transfer-Based Game Abstraction

The former two knowledge transfer mechanisms only modify the initialization of value functions in game theory-based MARL. However, the learning process of game theory-based MARL can be also improved by knowledge transfer. Recall that a game theory-based MARL algorithm needs to compute an equilibrium in each visited state, which is highly computationally expensive (e.g., computing a Nash equilibrium is a PPAD-hard problem [3]). In multi-agent systems with sparse interactions, equilibrium computation can be significantly reduced since in many states, there is no game between the agents or the game does not involve all the agents. To this end, we propose our third knowledge transfer mechanism, **model transfer-based game abstraction (MTGA)** in this subsection.

The basic idea of MTGA is to abstract the one-shot game in each state into a smaller one. That is, to remove the agents which do not have to join in the game. With fewer number of agents, an equilibrium can be computed with much less computational cost. In the context of game theory-based MARL, the definition of one-shot game is as follows.

Definition 5. Given a Markov game $M = \langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T \rangle$, the one-shot game corresponding to a state $s \in S$ is a tuple $G(s) = \langle N, \{A_i\}_{i=1}^n, \{Q_i(s)\}_{i=1}^n \rangle$. Let A be the joint action space. For each agent i, $Q_i(s) : A \rightarrow \Re$ is a set containing all its state-action values in state s.

Obviously, if an agent i is independent to other agents in a state s, it can be removed from the corresponding one-shot game $G(s)$. More accurately, if the local environmental dynamics of agent i in state s are very similar to those in its previous single-agent task, then it can be removed from the game. Thus, the method for evaluating the similarities between two MDPs can be also utilized to control the game abstracting process. Based on the concept of MDP similarity in Definition 4, we define an abstracted game as follows.

Definition 6. Given a Markov game $M = \langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T \rangle$, Let $G(s) = \langle N, \{A_i\}_{i=1}^n, \{Q_i(s)\}_{i=1}^n \rangle$ be the one-shot game in any state $s \in S$. Define a subset of N by $X = \{i \in N | \mathcal{D}_i(s_i) > \tau\}$, where s_i is the local state of agent i in state s, $\mathcal{D}_i(s_i)$ is the MDP similarity defined in Definition 4, and τ is a threshold value. The abstracted game of $G(s)$ derived from X is defined as $G_X(s) = \langle X, \{A_i\}_{i \in X}, \{Q_i(s)\}_{i \in X} \rangle$.

In this definition, the set X contains all agents that are considered to be related in the state s. If $|X| < |N|$, then the abstracted game $G_X(s)$ has a smaller size than $G(s)$.

By applying game abstraction in each state, the way of learning in game theory-based MARL can be changed: (1) for agents involved in the abstracted game, an equilibrium is computed for them and learning can be conducted only in their reduced joint state-action space, (2) and for the agents which are not in the abstracted game (i.e., independent agents), learning is conducted only in their local state-action space. Based on this idea, we propose the model transfer-based game abstraction (MTGA) in Algorithm 5.

Like the SVFT mechanism in Algorithm 4, MTGA also needs the agents' previous MDP models for computing the MDP similarities. The difference is that MTGA does not transfer value functions and learns in a totally different way. Also, each agent is not required to observe all joint states and joint actions of the Markov game. In MTGA, each agent i maintains two value functions Q_i and q_i, which are used for joint learning and independent learning, respectively. The state-action space of Q_i is empty initially (line 1) and can be extended dynamically (lines $8-12$). According to the abstracted game $G_X(s)$ in each state s, joint learning and independent learning are conducted for the agents involved in $G_X(s)$ and independent agents, respectively (lines $20-24$). Note that the term $\Phi_i(s')$ (at lines 21 and 24) has different meanings in different situations. If agent i is independent in state s', then $\Phi_i(s') = \max_{a_i'} q_i(s_i', a_i')$, where s_i' is the local state of agent i in state s'. Otherwise, it repre-

Algorithm 5: Model transfer-based game abstraction

Input: Learning rate α, discount factor γ, exploration factor ϵ, local MDP model $\hat{M}_i$ for each agent i, a threshold value τ, an integer L for Monte Carlo sampling

Initialize the joint state-action value function, $\forall i \in N, Q_i \leftarrow \emptyset$;
Initialize the local state-action value function, $\forall i \in N$, $\forall s_i \in S_i$, $\forall a_i \in A_i$, $q_i(s_i, a_i) \leftarrow 0$;
foreach *episode* **do**
  Initialize state s;
  repeat
    $X \leftarrow \{i \in N | \mathcal{D}_i(s_i) > \tau\}$;
    $s_g \leftarrow$ the joint state of the agents in X;
    foreach *agent* $i \in X$ **do**
      if s_g *is not in the state space of* Q_i **then**
        foreach *joint action* a_g *of the agents in* X **do**
          Extend Q_i to include (s_g, a_g);
          $Q_i(s_g, a_g) \leftarrow 0$;
    $G_X(s) \leftarrow \langle X, \{A_i\}_{i \in X}, \{Q_i(s)\}_{i \in X} \rangle$;
    foreach *agent* $i \notin X$ **do**
      $a_i \leftarrow \arg\max_{a_i'} q_i(s_i, a_i')$;
    foreach *agent* $i \in X$ **do**
      $a_i \leftarrow$ the action sampled by the equilibrium of $G_X(s)$ for agent i;
    $\vec{a} \leftarrow (a_1, ..., a_n)$;
    Receive the experience $(s, \vec{a}, r_i, s')$ for each agent i;
    foreach *agent* $i \notin X$ **do**
      $q_i(s_i, a_i) \leftarrow (1-\alpha) q_i(s_i, a_i) + \alpha(r_i + \gamma \Phi_i(s'))$;
    foreach *agent* $i \in X$ **do**
      $a_g \leftarrow$ the joint action taken by the agents in X;
      $Q_i(s_g, a_g) \leftarrow (1-\alpha) Q_i(s_g, a_g) + \alpha(r_i + \gamma \Phi_i(s'))$;
    $s \leftarrow s'$;
  until s *is a terminal state*;

sents the expected value of the equilibrium computed in the abstracted game of state s'.

4. EXPERIMENTS

In this section, the proposed knowledge transfer mechanisms, value function transfer (VFT), selective value function transfer (SVFT), and model transfer-based game abstraction (MTGA), are evaluated in the multi-agent grid world games presented by Melo and Veloso [17, 18], which are shown in Figure 2. The game NTU has 4 agents and the game NJU has 3 agents. All the other games have 2 agents. In all the games, agents are required to reach their goals within a few steps and avoid collisions. In the 2-agent games, the initial locations of the agents are denoted by cross symbols and each of their goals is the initial location of the other. In NTU and NJU, the initial location of each agent i is denoted by the number i and its goal is denoted by G_i. In each grid of a game, an agent can choose to move in one of the four directions (UP, DOWN, LEFT, RIGHT). The shaded grids are areas where any two agents pass simultaneously will cause collision. In other grids, agents can walk freely.

We adopt Nash Q-learning (NashQ) [10], correlated Q-learning (CEQ) [6], and negotiation-based Q-learning (NegoQ) [11] as the basic learning algorithms and implement the three knowledge transfer mechanisms on each of them. For CEQ, we implement the utilitarian version (uCEQ) proposed in literature [6]. Taking NashQ for example, the corresponding algorithms with the three knowledge transfer mechanisms are denoted by NashQ-VFT, NashQ-SVFT, and NashQ-MTGA, respectively. For comparison, we also implement CQ-learning [4], which is the state of the art.

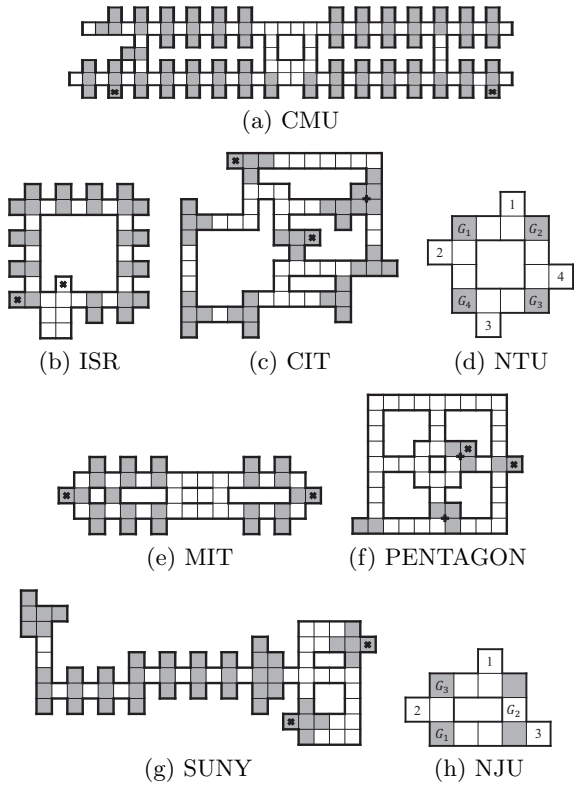
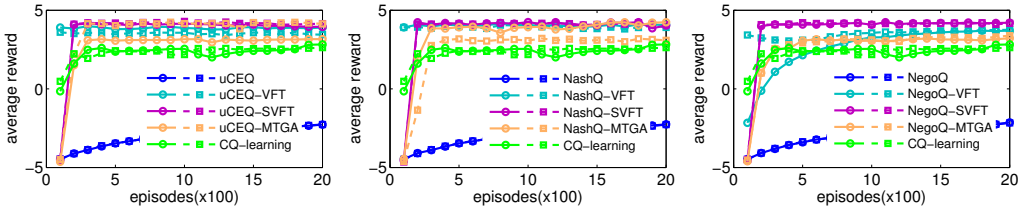


Figure 2: The multi-agent grid world games used in experiments

All the tested algorithms are run for 2000 iterations with an initial learning rate of 1.0 and a discount factor of 0.9. The learning rate has a decay of 0.99958 by each step. Exploration is executed by a fixed ϵ-policy with an exploration factor of 0.01. For algorithms with the SVFT and MG-TA mechanisms, the parameter L is 100 and the threshold value τ for controlling value function transfer (or game abstraction) is $\frac{1}{5}$ of the maximal value of the computed MDP similarities. As the games in Figure 2 have different sizes, the reward for an agent to reach its goal is set to +200 in the 2-agent games and +20 in the games NTU and NJU. The rewards for collisions and going into the wall are −10 in all games. In other cases, agents receive −1 for not reaching the goal. State transitions are made stochastic by assigning a 0.2 probability of failure to the agents' actions. In each of the 2000 learning iterations, the average reward per step (ARPS) achieved by each agent in the game is recorded.

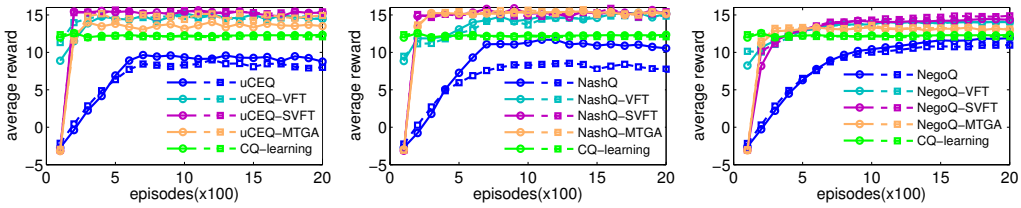
Due to space limitation, we only show part of the experimental results. For the 2-agent games, we show the results of the two most complicated games CMU and SUNY in Figures 3 and 4. For the 3-agent game NJU and the 4-agent game NTU, we only show the learning curves of CQ-learning, NegoQ, and all variants of NegoQ with knowledge transfer mechanisms in Figures 5 and 6, respectively. The results are averaged over 50 runs.

The learning curves belonging to the same algorithm are plotted in the same color. We first examine the results in the 2-agent games. It can be found that the three knowledge transfer mechanisms significantly improve the performance of all the tested game theory-based MARL algorithms. For example, in CMU, the ARPS achieved by all the basic learning algorithms are below 0 during the learning process. But



(a) uCEQ (b) NashQ (c) NegoQ

Figure 3: The learning curves of each tested algorithm in CMU.



(a) uCEQ (b) NashQ (c) NegoQ

Figure 4: The learning curves of each tested algorithm in SUNY.

with the knowledge transfer mechanisms, all of them can finally achieve an ARPS around 3.2 to 4.2. Some criteria of transfer learning [21] are satisfied, such as a jump start, better asymptotic performance, and higher total rewards. The ARPS achieved by the state-of-the-art algorithm CQ-learning finally reaches about 3.0, showing that our algorithms perform better.

By comparing the three knowledge transfer mechanisms, it can be found that the two value function transfer mechanisms generally perform better than MTGA on the aspect of ARPS. This is reasonable since with VFT and SVFT, the learning algorithms are learning based on a near-optimal initial policy while with MTGA, they still have to learn from scratch. SVFT performs slightly better than VFT as it does not transfer knowledge in states where doing so may cause a big loss. However, VFT and SVFT should compute an equilibrium in each state and requires observability of joint states and actions. In contrast, MTGA significantly reduces equilibrium computation and agents only need to learn in its local state-action space in most states.

Although NJU and NTU are small, learning in the two games are more challenging than in the 2-agent games since their joint state-action spaces are much larger. There are about 1.3×10^5 and 8×10^6 joint state-action pairs in NJU and NTU, respectively. For better exhibiting the results, each subfigure in Figures 5 and 6 only plots the learning curves of one learning algorithm. In NJU, the basic learning algorithm NegoQ cannot converge within the 2000 iterations as all its curves still rise at the end. With the two value function transfer mechanisms (VFT and SVFT), the learning curves of the agents 1, 2, and 3 finally converge to 2.3, 1.8, and 1.8, which are very close to the optimal ARPS values[2] 2.5, 2.0, and 2.0. NegoQ-MTGA also performs much better than NegoQ in NJU, with its three learning curves converging to 2.2, 1.55, and 1.75. The final ARPS values achieved by the state-of-the-art algorithm CQ-learning for the three agents are 1.85, 1.75, and 1.5, which indicates that all three variants of NegoQ (NegoQ-VFT, NegoQ-SVFT, and NegoQ-MTGA) perform better than CQ-learning in the game NJU. In the game NTU, similar results can be observed in Figure 6 and we do not repeat here.

[2]The optimal ARPS values are obtained by conducting Monte Carlo trails using an optimal policy.

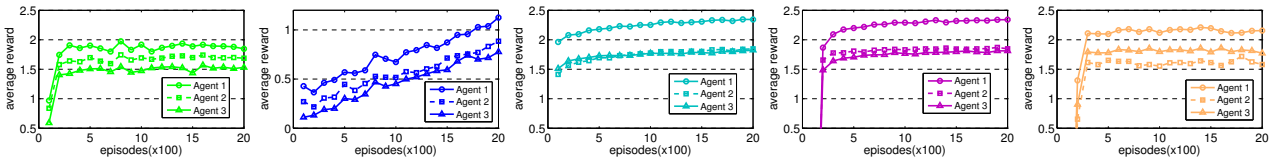
Table 1: The average runtime used by each algorithm in CMU, SUNY, NJU, and NTU

Algorithm	CMU	SUNY	NJU	NTU
CQ-learning	1.85s	0.39s	0.18s	6.92s
NegoQ	28.80s	3.42s	7.10s	409.70s
NegoQ-VFT	8.70s	2.48s	4.74s	112.11s
NegoQ-SVFT	9.43s	3.10s	4.40s	81.35s
NegoQ-MTGA	4.85s	1.88s	1.73s	31.99s

Table 1 shows the runtime results of the tested algorithms. Due to space limitation, we only show the average runtime of CQ-learning, NegoQ and all variants of NegoQ obtained in the maps CMU, SUNY, NJU, and NTU. As expected, CQ-learning and NegoQ are the fastest and the slowest, respectively. With knowledge transfer, NegoQ is greatly accelerated. For example, in CMU, the runtime taken for the original NegoQ algorithm is 28.80s. By the VFT, SVFT, and MTGA mechanisms, the corresponding runtimes are 8.70s, 9.43s, and 4.85s. The process of computing MDP similarities in SVFT and MTGA adds them extra runtime. Therefore, it can be found that NegoQ-SVFT needs more time to finish learning than NegoQ-VFT in CMU and SUNY. For MTGA, since equilibrium computation is avoided in most states, such extra time does not prevent it from being the fastest among the three knowledge transfer mechanisms. However, NegoQ-MTGA is still slower than CQ-learning as equilibrium computation cannot be totally avoided.

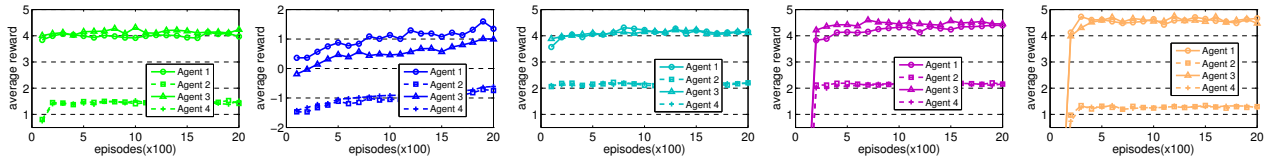
5. CONCLUSIONS

In this work, we contribute three knowledge transfer mechanisms, value function transfer (VFT), selective value function transfer (SVFT), and model transfer-based game abstraction (MTGA) to improve multi-agent reinforcement learning (MARL) in multi-agent systems with sparse interactions. Both VFT and SVFT utilize the agents' single-agent value functions to initialize their value functions in the MARL al-



(a) CQ-learning (b) NegoQ (c) NegoQ-VFT (d) NegoQ-SVFT (e) NegoQ-MTGA

Figure 5: The learning curves of CQ-learning, NegoQ, and all variants of NegoQ in NJU.



(a) CQ-learning (b) NegoQ (c) NegoQ-VFT (d) NegoQ-SVFT (e) NegoQ-MTGA

Figure 6: The learning curves of CQ-learning, NegoQ, and all variants of NegoQ in NTU.

gorithm. VFT conducts knowledge transfer in all states, but SVFT only transfers knowledge in states where the agents' local environmental dynamics are similar to those in the previous single-agent tasks. MTGA utilizes the agents' previous environment models to abstract the one-shot game in each state. Compared with VFT and SVFT, MTGA does not need to compute an equilibrium in most of the states and does not require observability of all joint states and actions.

We choose NashQ [10], CEQ [6], NegoQ [11] as the basic learning algorithms and test the three knowledge transfer mechanisms in benchmarks. The results show that the three knowledge transfer mechanisms significantly improve the learning performance of all the basic learning algorithms (a jump start, better asymptotic performance, and higher total rewards are observed). Also, they achieve better asymptotic performance and higher total rewards than the state-of-the-art algorithm CQ-learning [4].

Since VFT and SVFT only modify the process of value function initialization, their convergence properties totally depend on the corresponding learning algorithms. For the mechanism MTGA, there is no formal convergence guarantees currently since it changes the learning way of game theory-based MARL. Therefore, one interesting future direction is to theoretically analyze the convergence property of MTGA. Besides transferring value functions and environment models, there are many other effective knowledge transfer mechanisms in the transfer learning domain (e.g., transferring options and features) [21]. It would also be interesting to explore the performance of other knowledge transfer mechanisms in multi-agent systems with sparse interactions in the future.

6. ACKNOWLEDGMENTS

We would like to acknowledge the support for this work from the National Science Foundation of China (Grant Nos. 61432008, 61175042, 61321491, 61202212), and the Program for Research and Innovation of Graduate Students in General Colleges and Universities, Jiangsu (Grant No. CXLX13_049).

REFERENCES

[1] B. Banerjee, J. Lyle, L. Kraemer, and R. Yellamraju. Sample bounded distributed reinforcement learning for decentralized pomdps. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1256–1262, 2012.

[2] R. I. Brafman and M. Tennenholtz. R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[3] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[4] Y.-M. De Hauwere, P. Vrancx, and A. Nowé. Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 715–722, 2010.

[5] N. Ferns, P. Panangaden, and D. Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence (UAI '04)*, pages 162–169, 2004.

[6] A. Greenwald, K. Hall, and R. Serrano. Correlated Q-learning. In *Proceedings of International Conference on Machine Learning*, pages 242–249, 2003.

[7] C. Guestrin, M. G. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of the 9th International Conference on Machine Learning (ICML 2002)*, pages 227–234, 2002.

[8] C. Guestrin, S. Venkataraman, and D. Koller. Context-specific multiagent coordination and planning with factored mdps. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 253–259, 2002.

[9] Y. D. Hauwere, P. Vrancx, and A. 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.

[10] J. Hu and M. Wellman. Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.

[11] Y. Hu, Y. Gao, and B. An. Multiagent reinforcement learning with unshared value functions. *IEEE Transactions on Cybernetics*, doi:10.1109/TCYB.2014.2332042, accepted.

[12] Y. Hu, Y. Gao, and B. An. Accelerating multi-agent reinforcement learning by equilibrium transfer. *IEEE Transactions on Cybernetics*, doi:10.1109/TCYB.2014.2349152, accepted.

[13] J. R. Kok, P. J. Hoen, B. Bakker, and N. 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.

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

[15] G. Kuhlmann and P. Stone. Graph-based domain mapping for transfer learning in general games. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 188–200, 2007.

[16] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[17] F. S. Melo and M. 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.

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

[19] I. Szita and C. Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1031–1038, 2010.

[20] A. Taylor, I. Dusparic, E. Galván-López, S. Clarke, and V. Cahill. Transfer Learning in Multi-Agent Systems Through Parallel Transfer. In *Workshop on Theoretically Grounded Transfer Learning at the 30th International Conference on Machine Learning (Poster)*, 2013.

[21] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

[22] M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8:2125–2167, 2007.

[23] P. Vrancx, Y. D. Hauwere, and A. Nowé. Transfer learning for multi-agent coordination. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART 2011)*, pages 263–272, 2011.

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