Future Generation Computer Systems 68 (2017) 128–135

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Service composition based on multi-agent in the cooperative game

Yu Lei [a,b,*], Zhang Junxing [a]

[a] Inner Mongolia University, China

[b] State Key Laboratory of Networking and Switching Technology, BUPT, China

HIGHLIGHTS

- Agents obtain more learning experiences comparing with cooperative games for one fixed agent.
- Two-player cooperative games are used so that each agent can learn the strategy concurrently.
- Reinforcement learning is used in the multi-agents based service composition method.

ARTICLE INFO

Article history:
Received 30 September 2015
Received in revised form
16 May 2016
Accepted 27 June 2016
Available online 23 September 2016

Keywords:
Distributed artificial intelligence
Cooperative games
Multi-agent coordination
QoS
Reinforcement learning

ABSTRACT

The principle of service composition based on multi-agent is that multi-agent can coordinate to reach Pareto-optimal Nash equilibrium. Reinforcement learning algorithms can be used to deal with the coordination problem in cooperative games. In this paper, the multi-agent coordination problems in cooperative games for different user preference is investigated. In our case, each agent can represent a user's preference, and it finally learns a policy that is best fit for that user. Most previous works study the deterministic gain of a state. However, in practical service environments, the gain may be nondeterministic due to unstable Quality of Service (QoS). In addition, user preference should be considered. To avoid local optimal solution, we let each agent randomly change interacting partners in each iteration. Thus, an agent can learn its optimal strategy by interacting repeatedly with the rest of agents representing different user preference. The experimental results show that our reinforcement learning algorithm outperforms other learning methods.

1. Introduction

In the last decades, the technique of distributed cooperative multi-agent is used in many areas, such as computer science, mobile robots, spacecraft, sensor networks, etc.

For mutual benefits, an agent should effectively coordinate with other agents in multi-agent systems. The cooperative outcomes depend on an action the agent takes and actions taken by other interacting agents. One main issue is how multi-agents sharing common interests cooperate with each other. In cooperative multi-agent systems, the agents share common interests, but their reward functions could be either the same or different.

Generally, the beneficial increment of one agent also results in the beneficial increment of the whole group. When agents interact with each other to obtain the common profits, they have to solve some difficulties in cooperative environments. Because many actions can be selected in the cooperative game, one major challenge is how to select an equilibrium strategy for all agents. In this situation, effective coordination for multiple optimal joint actions between the agents is required. Because the cooperative game could be nondeterministic, another challenge is how to deal with the stochasticity. In this situation, agents must know whether the received payoffs are caused by the interaction with other agents or by the settings of the stochastic game.

Reinforcement learning is a kind of machine learning method, which can obtain satisfactory solutions without knowing the accurate system model [1]. Reinforcement learning algorithms can be used in cooperative environments. Several multi-agent reinforcement learning algorithms have been proposed recently. However, most previous works use the Q-learning algorithm as their basis and modify the Q-learning algorithm to change single-agent environments to cooperative multi-agent environments. Q-learning is a kind of Reinforcement Learning. In cooperative multi-agent environments, most of researchers consider just two or more players. In the end of repeated game the agents learn an optimal strategy by interacting with the same opponents.

[*] Corresponding author at: Inner Mongolia University, China.
E-mail address: yuleiimu@sohu.com (Y. Lei).

http://dx.doi.org/10.1016/j.future.2016.06.039
0167-739X/© 2016 Elsevier B.V. All rights reserved.

A large number of agents, which may behave independently and locally, do not have consciousness to interact with the other specific agents. They do not pay much effort to choose interaction target in some extent, and their actions are more or less random in the training process.

In some situations, the interactions between agents may be sparse. It is very possible that the interacting partners of the agent randomly change. The consequence is that an optimal joint action achieved by repeated coordination with one partner agent may fail if the agent interacts with a different partner agent in the future. Therefore, each agent needs to learn its strategy by interacting with different opponents. In addition, each agent can represent a user's preference, and it finally learns a policy that is best fit for that user. On behalf of the same preference of users, some agents may interact frequently for a similar aim.

Recently the coordination policies in cooperative multi-agent systems have been investigated. Besides the difficulties previously mentioned, achieving optimal coordination where the interacting partners are not fixed is more challenging. Non-fixed interacting partners introduce the additional stochasticity.

We study the multi-agent coordination problem in this paper The contributions of our work are as follows.

1. We propose a new agent selection strategy, where the partner agent is selected randomly to interact from the large number of agent population during each round. Comparing with repeated cooperative games for one fixed agent, the agent in our case can interact with more agents, thus it has more learning experiences from other agents.
2. According to cooperative game theory, two-player cooperative games are used to model the interactions of agents. To find Pareto-optimal Nash equilibrium, the proposed method records the history of actions to enable agents to learn the strategy concurrently. Using multi-agents reinforcement learning we propose a service composition method.

In Section 2, we review some background on reinforcement learning. In Section 3, we study previous work of multi-agent reinforcement learning in cooperative environments. In Section 4, the coordination problem in cooperative games is introduced, and we describe the multi-agent reinforcement learning algorithm. In Section 5, comparing other algorithms, the learning performance of our algorithm is presented. In Section 6, we summarize our paper.

2. Preliminaries

2.1. Reinforcement learning methods

Reinforcement Learning [1,2], which is a machine learning method, is used for decision-making. An agent repetitively interacts with the surroundings in a trial-and-error manner. The surrounding environment provides rewards to the agent when the agent performs actions. Through gaining the maximum reward, Reinforcement Learning can learn an optimal strategy [3]. Unlike any unsupervised and supervised learning techniques, Reinforcement Learning proceeds by sequentially interacting with the surroundings instead of using a training set.

Though it is well suitable to multi-agent applications, this learning method has its limits. It requires the knowledge of a complete model for the surroundings so that the agent can perform actions to affect the surroundings. In addition, the reward depends on system states and applied actions. Unlike supervised learning using a labeled set, Reinforcement Learning maximize the cumulative rewards [4,5].

Until now, several Reinforcement Learning algorithms have been proposed such as TD(λ), Sarsa, Q-learning, etc. Most of them use the Markov Decision Process (MDP). On the current state, the agent makes a decision to choose an action, and the result is that it transfers to a new state. A historical sequence of actions produces a cumulative reward to show the quality of these actions.

Reinforcement Learning includes two learning styles: model-free learning and model-based learning. Specifically, the agent learns the optimal policy according to the transition and reward functions [6], or the agent learns the policy without knowing the transition and reward functions [7].

There are three basic methods to solve the Reinforcement Learning problem. They are Temporal-Difference (TD) learning, Monte Carlo method, and dynamic programming. Each of them has its advantages and disadvantages. According to complete information of environment, dynamic programming methods use iterative mathematical formulations to obtain the solution. In contrast, Monte Carlo methods do not need a complete environmental model. Nevertheless, Monte Carlo methods learn the policy based on episodes rather than actions. To take advantage of above two methods, Temporal-Difference methods combine Monte Carlo method with dynamic programming method. TD methods also do not need a complete environmental model, and TD methods are based on actions. However, its disadvantages are more complex to analyze. In the following, we will specifically investigate a kind of TD learning: Q-learning.

2.2. Q-learning

A Q-learning system includes three major parts: environment, agent and policy. Environment, which an agent observes, provides the agent with the current state s_t and the immediate reward $R(s_t, a_t)$, which is the result from the action a_t just executed by the agent. The agent part [8] stores a table containing $\{Q(s_t, a_t)\}$ and an updating rule. The policy shows the agent should take which action in a given state.

In the Q-learning algorithm, the agent need to learn an optimal policy $\pi : S \to A$. Given the current state $s \in S$, π tells the agent which action $a \in A$ to take to maximize a cumulative reward in the learning process. Given a policy π and a state s_t at time step t, the value of s_t under π can be calculated as follows:

$$V^{\pi}(s_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \tag{1}$$

where γ is the discount factor $(0 < \gamma < 1)$. r_{t+1} is the immediate reward at time t. Thus by calculating optimal value function: $V^{\pi}(s_t)$, we can obtain an available optimal policy π. Unfortunately, the environment model (e.g. reward function) is unknown in many practical problems, which means Eq. (1) cannot be calculated. As mentioned earlier, we need the Q-learning in this case. In Eq. (2), $Q(s_t, a_t)$ is the maximum discounted cumulative reward achieved by optimally taking action a_t from state s_t.

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma V(s_{t+1}). \tag{2}$$

After taking action a_t, s_{t+1} is the next state from state s_t in the deterministic environment. $R(s_t, a_t)$ is the immediate reward function. Because $V(s_{t+1}) = \max\{Q(s_{t+1}, a_{t+1})\}$, Eq. (2) can be formulated as follows:

$$Q(s_t, a_t) = R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}). \tag{3}$$

The $\langle s_t; a_t; r_t; s_{t+1}\rangle$ tuples construct experience of the agent in each iteration, forming a table of estimated $Q(s_t, a_t)$ values. The $Q(s_t, a_t)$ values generally converge to their optimal values after repetitively trying different state–action pairs.

One of advantages of Q-learning method is that we can obtain the optimal policy even by randomly performing a sequence of actions, if only we try state–action pairs constantly. This feature makes Q-learning the simplest and model-free learning method. However, because the large number of state–action pairs is a