entropy

MDPI

Article

The Convergence of a Cooperation Markov Decision Process System

Xiaoling Mo [1], Daoyun Xu [1,*] and Zufeng Fu [1,2]

[1] College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; gs.xlmo19@gzu.edu.cn (X.M.); fuzufeng@outlook.com (Z.F.)
[2] Department of Electronics and Information Engineering, Anshun University, Anshun 561000, China
* Correspondence: dyxu@gzu.edu.cn; Tel.: +86-851-83620667

Received: 23 July 2020; Accepted: 27 August 2020; Published: 30 August 2020

Abstract: In a general Markov decision progress system, only one agent's learning evolution is considered. However, considering the learning evolution of a single agent in many problems has some limitations, more and more applications involve multi-agent. There are two types of cooperation, game environment among multi-agent. Therefore, this paper introduces a Cooperation Markov Decision Process $(CMDP)$ system with two agents, which is suitable for the learning evolution of cooperative decision between two agents. It is further found that the value function in the $CMDP$ system also converges in the end, and the convergence value is independent of the choice of the value of the initial value function. This paper presents an algorithm for finding the optimal strategy pair (π_k^0, π_k^1) in the $CMDP$ system, whose fundamental task is to find an optimal strategy pair and form an evolutionary system $CMDP^{(\pi_k^0, \pi_k^1)}$. Finally, an example is given to support the theoretical results.

Keywords: reinforcement learning; cooperation markov decision process; multi-agent; optimal pair of strategies

1. Introduction

Artificial intelligence technology has become one of the most important technologies, nowadays. AlphaGo, unmanned driving, voice recognition, face recognition and other well-known technologies involve artificial intelligence. Artificial intelligence technology has been widely used in economic construction, education, medical treatment, social life and other fields. Many industries have also begin to evolve to artificial intelligence, intelligence has become an important direction of industrial development. Machine learning generally includes supervised learning, unsupervised learning and reinforcement learning [1]. Among them, reinforcement learning is an important research field. Reinforcement learning is learned by interacting with a dynamic environment through trial and error, is an important branch in the field of machine learning and artificial intelligence. It is leap to control the behavior of agents through rewards and punishments. Reinforcement learning is widely used in many fields, such as: electronic game [2], board games [3], maze game, recommendation system [4], and so forth.

The important basis of reinforcement learning [5] is Markov Decision Process (MDP) system [6]. From the evolution process of the computational model, Markov Process (MP) system is the basic model, which introduces reward function and execution behavior, and introduces a mapping from state set to behavior set as a strategy to solve Markov decision problem. Under the given policy π, the Markov decision process system degenerates into a Markov process system with a reward function MDP^{π}. Once the policy is given, the main task of the Markov process system with the reward function is to evolve and generate the expected reward values collected in each state. The main objective of

the Markov decision system is to find an optimal strategy [7,8] π^*, so that the expected reward value collected on the Markov chain can be maximized under this strategy.

In the usual Markov decision process system, only an agent is learning and evolving. However, in real problems, the decision-making ability of an agent is far from enough, and many problems can be solved by using a multi-agent [9–11]. Therefore, we encounter two kinds of problems: one is that multi-agent restrict each other to accomplish their target tasks in the same environment. Such systems are called antagonistic systems [12] or game systems [13]. In this kind of system, all the intelligent agent executes the strategy behavior alone, according to mutually restricts or plays games with each other intelligent agent to finishes its own target task. The optimal strategy is represented as agent strategy vector group and the reward value is reflected as the reward value of its own execution. Littman [12] proposed minimax-q algorithm for multi-agent competitive reinforcement learning based on Markov decision framework based on zero-sum game in 1994. The other is that multi-agent work together to accomplish a target task in the same environment. Such systems are called collaborative systems [14] or cooperative [15]. In this kind of system, all the intelligence agent executes the strategy behavior alone, according to cooperates with each other to completes one target task jointly. The optimal strategy is the vector set of agent strategy, and the reward value is the social comprehensive value of combinatorial execution. Littman [16] also proposed a collaborative team q-learning algorithm in 2001. The multi-agent cooperative algorithm is used to solve problem about traffic signal signal coordination control [17], multi-agent path planning [18], multi-satellite collaborative task planning [19], and so forth. In the case of multi-agent, the state shift is the result of all agents acting together, so the rewards for agents also depend on the joint strategy. Iwata K et al. [6]analyzed the statistical properties of multi-agent learning based on Markov decision process. However, the multiple agents in this paper are independent, and the joint action will make the state transfer and reward value of the whole system, but only migrates the Markov decision process of a single agent to multi-agent. Zhang W et al. [20] has began to introduce the communication mechanism into the multi-agent system. Although agents begin to communicate with each other, the consumption of memory was enlarged, and agents could not predict the impact of their actions on the action execution of other agents.

This paper only considers the Cooperation Markov Decision Process $(CMDP)$ system of two agents, which is suitable for the evolutionary learning system of cooperative decision between two agents. Two-agent games for multi-agent reinforcement learning are similar to perceptrons for neural networks.In this kind of learning model, agents alternately execute behaviors, seek optimal criteria based on social value, seek optimal strategies (π_k^0, π_k^1) , and jointly complete the target task. This paper introduces a cooperation Markov decision process system in the form of definition, two trade agent (Alice and Bob) on the basis of its strategy to perform an action. that is, after Bob observes that Alice performs an action, Bob is deciding which action to perform, and further Bob's execution of the action will also affect the execution of Alice's next action. The Markov system learns evolutionally through a given strategy pair and looking for a pair of optimal strategy (π_k^0, π_k^1) , form an evolution system $CMDP^{(\pi_k^0, \pi_k^1)}$, and gives the algorithm [21,22] to search for the optimal strategy pair. In this paper, the convergence property of the value function of the MDP system with the participation of a single agent is given, the convergence phenomenon of the value function in the cooperation Markov decision process system proposed in this paper is further explored, and the correctness of the property is proved from both the experimental and theoretical perspectives.

2. Markov Reward Process System

A finite Markov process system is constituted by two tuples $< S, P >$, $S = \{s_1, \ldots\ldots, s_n\}$ is a set of finite states (containing n states), define a probability distribution P over S, $P(s, s')$ represents the probability of transition from state s to state s', what's more, $\forall s \in S : \Sigma_{s' \in S} P(s, s') = 1$. Markov Reward Process is a kind of Markov chain with value, which consisted of four tuples $< S, P, R, \gamma >$. Introduced a reward function, $R: \ S \times S \to R^+$ (R^+ means non-negative real number set). So $R(s, s')$ is the reward value obtained by transferring from state s to state s'; $0 < \gamma < 1$ is the attenuation factor,

also known as the discount factor. Recursively defines an expected reward value (value function) collected by Markov process system:

$$V(s) = \Sigma_{s' \in S} P(s, s')[R(s, s') + \gamma \cdot V(s')]. \tag{1}$$

Equation (1) recorded the reward value obtained on the Markov chain $X_0, X_1, X_2, \ldots, X_t, X_{t+1}, \ldots$ with the state s as the initial state. The term $R(s, s') + \gamma \cdot V(s')$ indicated that: starting from the state s, the reward value obtained by one-step transition to s' and plus the discount of the expected reward value collected from the state s'. Equivalently, writen (1) as the following iterative formula:

$$V_{k+1}(\vec{s}) = diag(PR^T) + \gamma \cdot PV_k(\vec{s}). \tag{2}$$

Among them, R^T represents the transpose of the matrix R and $diag(A)$ represents the column vector formed by the diagonal elements of the matrix A.

Now we make an experiment, experiment 1: take probability matrix P , reward function R, and discount factor γ as follows. Observe the convergence of the value function V through an example.

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1/3 & 2/3 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 1/4 & 0 & 3/4 & 0 \end{pmatrix} R = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 & 0 \end{pmatrix} \gamma = 0.85$$

Initially, initial vector is set as $V_0 = [0, 0, 0, 0, 0]^T$, the convergence of the value function is shown in the following Figure 1:

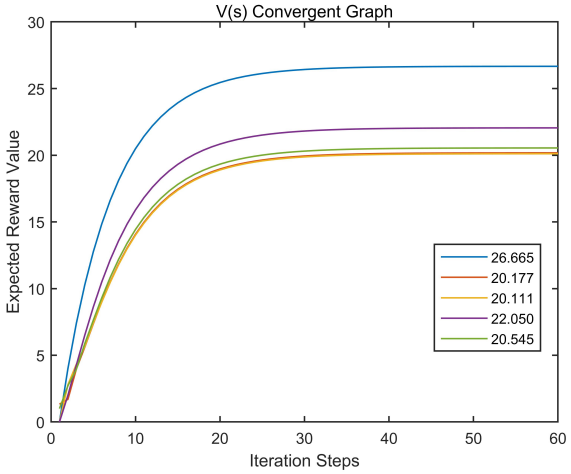

Figure 1. V-function convergence diagram.

According to Figure 1, we can observed that starting from the initial state, the value function (V-function) will eventually converge as the number of iterations steps. We further speculate that the final convergence value has nothing to do with the initial value setting.

Theorem 1. *Under any initial state* $V_0(\vec{s})$ *, randomly walk on the Markov reward process system, the value function* $V_k(\vec{s})$ *finally obtain converges.*

Proof. From the iterative Formula (2), it is easy to obtain that,

$$\begin{aligned} V_k(\vec{s}) &= diag(PR^T) + \gamma \cdot PV_{k-1}(\vec{s});\ V_{k-1}(\vec{s}) = diag(PR^T) + \gamma \cdot PV_{k-2}(\vec{s}); \\ V_2(\vec{s}) &= diag(PR^T) + \gamma \cdot PV_1(\vec{s});\ V_1(\vec{s}) = diag(PR^T) + \gamma \cdot PV_0(\vec{s}); \end{aligned} \tag{3}$$

simplify Equation (3) to calculate the difference in value function between $(k+1)$-step and k-step,

$$\begin{aligned} V_{k+1}(\vec{s}) - V_k(\vec{s}) &= \gamma \cdot P\left(V_k(\vec{s}) - V_{k-1}(\vec{s})\right) = \gamma^2 \cdot P^2\left(V_{k-1}(\vec{s}) - V_{k-2}(\vec{s})\right) \\ &= \gamma^3 \cdot P^3\left(V_{k-2}(\vec{s}) - V_{k-3}(\vec{s})\right) = \gamma^k \cdot P^k\left(V_1(\vec{s}) - V_0(\vec{s})\right) \\ &= \gamma^k \cdot P^k \cdot diag(PR^T) + \gamma^k \cdot P^k \cdot (\gamma \cdot P - E)\, V_0(\vec{s}). \end{aligned} \tag{4}$$

Let P is the probability transfer matrix, R is the reward value matrix, the discount factor satisfy $0 < \gamma < 1$. We have

$$\begin{aligned} &\begin{bmatrix} R^1_{\min} & \cdots & R^n_{\min} \end{bmatrix}^T - (V_{\max} - \gamma \cdot V_{\min}) \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \le diag(PR^T) - (E - \gamma \cdot P)\, V_0(\vec{s}) \\ &\le \begin{bmatrix} R^1_{\max} & R^2_{\max} & \cdots & R^n_{\max} \end{bmatrix}^T - (V_{\min} - \gamma \cdot V_{\max}) \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T, \end{aligned} \tag{5}$$

where $R^i_{\min}$ and $R^i_{\max}$ record the minimum and maximum value of the reward value in the i-th row of the reward matrix R, respectively. $V_0(\vec{s})$ represent the value of the initial moment: $V_0(\vec{s}) = \begin{bmatrix} V_{01} & V_{02} & \cdots & V_{0n} \end{bmatrix}^T$, the minimum and maximum value of V are $V_{\min} = \min\{V_0(\vec{s})\}$ and $V_{\max} = \max\{V_0(\vec{s})\}$, respectively. Reduce the left side of the inequality (2.5) and enlarge the right side. Then we conclude,

$$\begin{aligned} &\gamma^k \cdot (R^*_{\min} - V_{\max} + \gamma \cdot V_{\min}) \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T \le V_{k+1}(\vec{s}) - V_k(\vec{s}) \\ &V_{k+1}(\vec{s}) - V_k(\vec{s}) \le \gamma^k \cdot (R^*_{\max} - V_{\min} + \gamma \cdot V_{\max}) \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T, \end{aligned}$$

where $R^*_{\min}$ and $R^*_{\max}$ record the minimum and maximum value in the reward matrix R, respectively.

Finally, because of the discount factor $0 < \gamma < 1$, with the constant increase of k, the value of γ^k approaches zero infinitely, so the influence of $R^*_{\min}$, $R^*_{\max}$, $V_{\max}$ and $V_{\min}$ are much smaller than γ^k under general constants number, therefore $0 \le V_{k+1}(\vec{s}) - V_k(\vec{s}) \le 0$. Obviously, when the value of k is large, $V_{k+1}(\vec{s}) = V_k(\vec{s})$, that is to say the value of the final value function converges. □

Theorem 2. *The choice of initial value $V_0(\vec{s})$ does not affect the convergence value of the final value function $V_k(\vec{s})$.*

Proof. Simplifying Equation (3):

$$\begin{aligned} V_k(\vec{s}) &= diag(PR^T) + \gamma \cdot PV_{k-1}(\vec{s}) = diag(PR^T) + \gamma \cdot P \cdot \left(diag(PR^T) + \gamma \cdot PV_{k-2}(\vec{s})\right) \\ &= diag(PR^T) + \gamma \cdot P \cdot \left(diag(PR^T) + \gamma \cdot P \cdot \left(diag(PR^T) + \gamma \cdot PV_{k-3}(\vec{s})\right)\right) \\ &= \left(E + \gamma \cdot P + (\gamma \cdot P)^2 + (\gamma \cdot P)^3 + ... + (\gamma \cdot P)^{k-1}\right) \cdot diag(PR^T) + (\gamma \cdot P)^k \cdot V_0(\vec{s}). \end{aligned} \tag{6}$$

Because P is a probability transition matrix, then P^k is a probability transition matrix also. Then we can get,

$$\begin{bmatrix} R^*_{\min} \\ R^*_{\min} \\ \vdots \\ R^*_{\min} \end{bmatrix} \leq \begin{bmatrix} R^1_{\min} \\ R^2_{\min} \\ \vdots \\ R^n_{\min} \end{bmatrix} \leq p^i \cdot diag(PR^T) \leq \begin{bmatrix} R^1_{\max} \\ R^2_{\max} \\ \vdots \\ R^n_{\max} \end{bmatrix} \leq \begin{bmatrix} R^*_{\max} \\ R^*_{\max} \\ \vdots \\ R^*_{\max} \end{bmatrix}, i \in \{1,2,\cdots,k\}, \tag{7}$$

$$\begin{bmatrix} V_{\min} & V_{\min} & \cdots & V_{\min} \end{bmatrix}^T \leq P^k \cdot V_0(\overrightarrow{s}) \leq \begin{bmatrix} V_{\max} & V_{\max} & \cdots & V_{\max} \end{bmatrix}^T, \tag{8}$$

applying (7), (8) and simplifying gives

$$V_k(\overrightarrow{s}) \geq (1+\gamma+\gamma^2+\cdots+\gamma^{k-1}) \cdot R^*_{\min} + \gamma^k \cdot V_{\min}, \tag{9}$$

$$V_k(\overrightarrow{s}) \leq (1+\gamma+\gamma^2+\cdots+\gamma^{k-1}) \cdot R^*_{\max} + \gamma^k \cdot V_{\max}, \tag{10}$$

expand the inequality (9),

$$\begin{aligned} &(1+\gamma+\gamma^2+\cdots+\gamma^{k-1}) \cdot R^*_{\min} + \gamma^k \cdot V_{\min} = (\tfrac{1}{\gamma^k} + \tfrac{1}{\gamma^{k-1}} + \cdots + \tfrac{1}{\gamma}) \cdot \gamma^k \cdot R^*_{\min} + \gamma^k \cdot V_{\min} \\ &= \gamma^k[(\tfrac{1}{\gamma^k} + \tfrac{1}{\gamma^{k-1}} + \cdots + \tfrac{1}{\gamma}) \cdot R^*_{\min} + V_{\min}] \approx \gamma^k[(\tfrac{1}{\gamma^k} + \tfrac{1}{\gamma^{k-1}} + \cdots + \tfrac{1}{\gamma}) \cdot R^*_{\min}]; \end{aligned} \tag{11}$$

expand the inequality (10),

$$\begin{aligned} &(1+\gamma+\gamma^2+\cdots+\gamma^{k-1}) \cdot R^*_{\max} + \gamma^k \cdot V_{\max} = (\tfrac{1}{\gamma^k} + \tfrac{1}{\gamma^{k-1}} + \cdots + \tfrac{1}{\gamma}) \cdot \gamma^k \cdot R^*_{\max} + \gamma^k \cdot V_{\max} \\ &= \gamma^k[(\tfrac{1}{\gamma^k} + \tfrac{1}{\gamma^{k-1}} + \cdots + \tfrac{1}{\gamma}) \cdot R^*_{\max} + V_{\max}] \approx \gamma^k[(\tfrac{1}{\gamma^k} + \tfrac{1}{\gamma^{k-1}} + \cdots + \tfrac{1}{\gamma}) \cdot R^*_{\max}]. \end{aligned} \tag{12}$$

It is found that, because of the discount factor $0 < \gamma < 1$, the value of $\frac{1}{\gamma^k}$ continuously approaches infinity as the value of k increases, so $\frac{1}{\gamma^k} + \frac{1}{\gamma^{k-1}} + \cdots + \frac{1}{\gamma} \gg 1$ and monotonically increase on the Formulas (11) and (12). Therefore, the effects of $V_{\min}$ and $V_{\max}$ in the initial value function $V_0(\overrightarrow{s})$ can be ignore for the final value function $V_k(\overrightarrow{s})$. so the choice of the initial $V_0(\overrightarrow{s})$ does not affect the convergence value of the final value function $V_k(\overrightarrow{s})$. It further proves that our conjecture is correct. □

3. Markov Decision Process System

Markov decision process system introduces the behavior in Markov reward process system and modifies the reward function to the behavior execution step. *MDP* system is a random process on state, reward, action sequences, and consists of five-tuples $< S, A, P, R, \gamma >$. $A = \{a_1, ..., a_m\}$ is a set of finite action, $P(s, a, s')$ represents the probability of transition from state s to state s' by performing action a. $P : S \times A \times S \to [0,1]$, $P(s,a,s') = P(s_{t+1} = s' | s_t = s', a_t = a)$, where for each pair of "state-behavior" (s, a), $\Sigma_{s' \in S} P(s, a, s') = 1$. The reward function R define with, $R : S \times A \times S \to R$, where $R(s, a, s') = E(R_{t+1} | s_t = s, a_t = a)$. Similarly, γ is the attention factor also called discount factor, satisfy $0 < \gamma < 1$. π is a strategy of *MDP* system, $\pi : S \to A$. Such as, $\pi(s) = a$ represents the execution of action a in state s. So, let $V^\pi(s)$ is the value of s under a given policy π, we can include

$$V^\pi(s) = \Sigma_{s' \in S} P(s, \pi(s), s')[R(s, \pi(s), s') + \gamma \cdot V^\pi(s')] \,. \tag{13}$$

Equation (13) is called the Bellman recursive equation. In the function V^π, if see behavior as a variable, we can induce a state-action value function, that is, a Q-function, $Q : S \times A \to R$

$$Q^\pi(s, a) = \Sigma_{s' \in S} P(s, a, s')[R(s, a, s') + \gamma \cdot V^\pi(s')] \,. \tag{14}$$

$Q^{\pi}(s,a)$ is understood as follows: starting from state s, after executing action a and following executing strategy π , the expected reward value is collected. The goal for any given Markov decision process is to find an optimal strategy, that is, to obtain the most reward strategy.

Policy function $\pi : S \to A$, determines the V-function and Q-function that indicate the expected collection of reward values, and Q^{π} can be determined by V^{π}. Therefore, our goal is to find a strategy π that maximizes the function V^{π}. A V-function is optional means if for any policy π, for any state $s \in S$, satisfies $V^{*}(s) \geq V^{\pi}(s)$. A policy is optimal indicates if for any policy π, for any state $s \in S$, satisfies $V^{\pi *}(s) \geq V^{\pi}(s)$. So, if $V^{\pi *}$ function is an optional V-function V^{*}, there is $V^{*}(s) = V^{\pi *}(s)$, for any state $s \in S$.

A natural problem is the existence of optimal strategy π^{*}. Second, if it exists, how should we to calculate π^{*} ?

According to Theorem 1 in Section 2, the value function of the Markov reward process system will converge, and the value function of the Markov decision process system formed by introducing behaviors on Markov reward process system must also converge. Theoretical results [23,24] show that: an optimal strategy for a finite Markov decision process system $< S, A, P, R, \gamma >$ exists, and the optimal V-function V^{*} and the optimal strategy π^{*} have the following relationship:

(i) If the optimal function V^{*} has been obtained, the optimal strategy π^{*} can be obtained:

$$\pi^{*}(s) = \arg\max_{a} \Sigma_{s' \in S} P(s,a,s')[R(s,a,s') + \gamma \cdot V^{*}(s')] \; ; \tag{15}$$

(ii) If the optimal policy π^{*} has been obtained, the optimal function V^{*} can be obtained:

$$V^{*}(s) = V^{\pi *}(s) = \Sigma_{s' \in S} P(s, \pi^{*}(s), s')[R(s, \pi^{*}(s), s') + \gamma \cdot V^{\pi *}(s')] \; . \tag{16}$$

Equations (15) and (16) give a cross-iterative solution process, and finally the optimal strategy π^{*} and the optimal function V^{*} are obtained.

The execution process of the MDP system is shown in the following Figure 2:

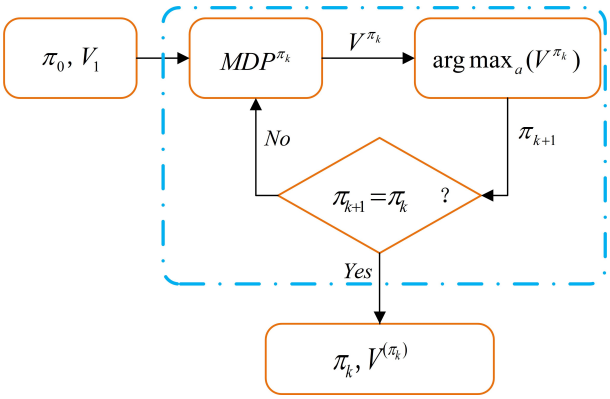

Figure 2. Optimal strategy solution algorithm framework.

4. Cooperation Markov Decision Process System

This section introduces a joint formal system that two agents run in the same Markov decision process system, called the cooperation Markov decision process system.

Suppose $A = (a_{i,j})_{n \times n}$, $B = (b_{i,j})_{m \times m}$ are two matrices, the tensor product of matrix A and B is defined as: $A \otimes B = (a_{i,j}B)$. If P_1, P_2 are two probability matrices but not necessarily the same number of rows, so $P_1 \otimes P_2$ is a probability matrix.

Assume $S^1 = \{s_1^1, \ldots, s_n^1\}$, $S^2 = \{s_1^2, \ldots, s_m^2\}$ are two state sets, (S^1, P_1), (S^2, P_2) are two MDP systems, state set $S = S^1 \times S^2$, S's state transition matrix is $P = P_1 \otimes P_2$, then MDP system (S, P) is the product system of MDP system (S^1, P_1) and MDP system (S^2, P_2).

4.1. Cooperation Markov Decision Process System with Two Agents

We now consider that, in the same system, there are two agents A and B, each agent performing their own action and completing the target task in a cooperative manner. For the "cooperative" type, the two agents cooperate to accomplish the same goal. Formally, such a system is defined as the following quintuple:

$$< S,\ A,\ \{Agent_0, Agent_1\},\ \{P_0,\ P_1\},\ \{R_0,\ R_1\} >$$

Among them, S is a finite state set, here represents the state pair of two agents; A is an action set, here represents the action pair of two agents; P_i is the probability transfer function of $Agent_i$, $i = (0,1)$, define as $P_i : S \times S \times A \times S \times A \rightarrow [0,1]$. For any $(s_1, s_2, a) \in S \times S \times A$, there hold that $\Sigma_{(s',a') \in S \times A} P_i(s_1, s_2, a, s', a') = 1$. In other words, for any fixed (s_1, s_2, a), the $P_i(s_1, s_2, a, \bullet, \bullet)$ specifies a probability distribution for $S \times A$. Intuitively, $P_0(s_1, s_2, a, s', a')$ express that when $Agent_0$, $Agent_1$ in state pair (s_1, s_2), when $Agent_0$ after executes action a transitions to s' state, and the probability that collaborator $Agent_1$ responds to execution of action a'. R_i is the reward function of $Agent_i$, $i = (0,1)$, defined as: $R_i : S \times S \times A \times S \times A \rightarrow R$ (real number set). Intuitively, $R_0(s_1, s_2, a, s', a')$ express that when $Agent_0$, $Agent_1$ in state pair (s_1, s_2), $Agent_0$ after executes action a transitions to s' state, and the reward value that collaborator $Agent_1$ responds to execution of action a'.

Obviously, two agents $Agent_0$, $Agent_1$ correspond to two V-functions: $V_i : S \times S \rightarrow R$ $(i = 0,1)$, in the joint form, we use the social value to describe the joint V-function $V : S \times S \rightarrow R$. Formally defined as:

$$V(s,t) = \alpha V_0(s,t) + (1-\alpha)V_1(s,t),\ (0 < \alpha < 1). \tag{17}$$

Among them, parameter α is called balance parameter.

Policy function defined as: $\pi_i : S \times S \rightarrow A$ $(i = 0,1)$, $\pi_i(s,s') = a$ understand as: when $Agent_i$ is in s state and observe that when $Agent_{1-i}$ is in s' state, $Agent_i$ executes action a. For a given strategy pair (π_0, π_1) , the evolutionary cooperation Markov decision process system $CMDP^{(\pi_0,\pi_1)}$ obtains a stable V-function whose iterative equation is:

$$\begin{aligned} V^0_{k+1}(s,t) &= \Sigma_{s' \in S} \Sigma_{a \in A} P_0(s,t,\ \pi_0(s,t), s',\ a)[R_0(s,t,\ \pi_0(s,t), s', a) + \gamma \cdot V^k(s',t)] \\ V^1_{k+1}(s,t) &= \Sigma_{t' \in S} \Sigma_{a \in A} P_1(s,t,\ \pi_1(s,t),\ t',\ a)\ [R_1(s,t,\ \pi_1(s,t), t',\ a) + \gamma \cdot V^k(s,t')] \\ V_{k+1}(s,t) &= \alpha V^0_{k+1}(s,t) + (1-\alpha)V^1_{k+1}(s,t). \end{aligned} \tag{18}$$

4.2. Convergence of the Social Value Function of CMDP System

Since the value function will eventually converge during Markov's decision-making process, we guess whether the value function also has the same properties in $CMDP$ system. We first observe by an experiment:

Suppose a $CMDP$ system $< S,\ A,\ \{Agent_0, Agent_1\},\ \{P_0,\ P_1\},\ \{R_0,\ R_1\} >$, where the discount factor γ=0.95, error coefficient setting ε=0.0001, a state set

$$S = \{(s_0,s_0), (s_0,s_1), (s_0,s_2), (s_1,s_0), (s_1,s_1), (s_1,s_2), (s_2,s_0), (s_2,s_1), (s_2,s_2)\},$$

an action set $A = \{(a_0,a_0), (a_0,a_1), (a_1,a_0), (a_1,a_1)\}$, the probability transition function P and the reward function R are as follows:

$P_1(S,a_0,S) = P_0(S,a_0,S)$:

	s_0	s_1	s_2
s_0	0.5	0	0.5
s_1	0.7	0.1	0.2
s_2	0.4	0	0.6

$P_1(S,a_1,S) = P_0(S,a_1,S)$:

	s_0	s_1	s_2
s_0	0	0	1
s_1	0	0.95	0.05
s_2	0.3	0.3	0.4

$$R_1(S,a_0,S)=R_0(S,a_0,S):\quad \begin{array}{c|ccc} & s_0 & s_1 & s_2 \\ s_0 & 1 & 0 & 1 \\ s_1 & 5 & 1 & 1 \\ s_2 & 1 & 0 & 1 \end{array} \qquad R_1(S,a_1,S)=R_0(S,a_1,S):\quad \begin{array}{c|ccc} & s_0 & s_1 & s_2 \\ s_0 & 0 & 0 & 1 \\ s_1 & 0 & 1 & 1 \\ s_2 & -1 & 1 & 1 \end{array}$$

There are 9 state groups in the $CMDP$ system. Figure 3 depicts the change curve of the social value function under the strategy of $(\pi_0,\pi_1)=\begin{pmatrix} s_0 & s_1 & s_2 \\ a_0 & a_1 & a_1 \end{pmatrix}, \begin{pmatrix} s_0 & s_1 & s_2 \\ a_1 & a_0 & a_1 \end{pmatrix}$. As can be seen from the Figure 3, in the cooperation Markov decision process system introduced in this paper, as the number of iterations increases, the social value of the system increases and finally converges to a stable value.

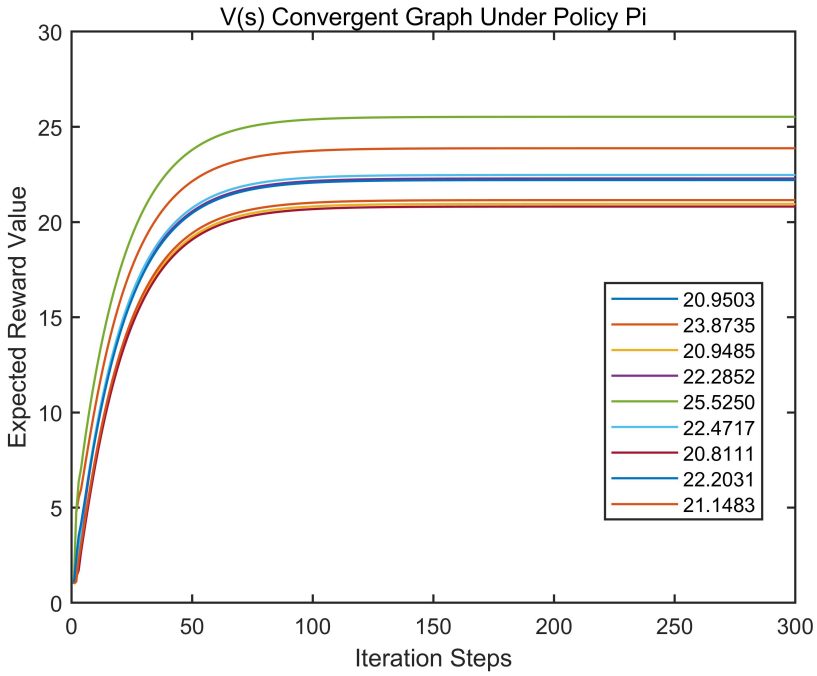

Figure 3. Convergence diagram of the social value function of the Cooperation Markov Decision Process ($CMDP$) system.

In the $CMDP$ system, agent $Agent0$ has $NS0$ states and agent Agent1 has $NS1$ states, where $NS0 = NS1$. The set of actions that two agents can perform are $\{1,2,3,\ldots,a\}$ and each agent has an action that can be performed. Let s denote the state of agent $Agent0$, and t denote the state of agent $Agent1$, for a given strategy pair (π_0,π_1), let $i = Pi0(s)$ denote the action performed by $Agent0$ in the s state under the π_0 strategy, let $j = Pi0(t)$ denote the action perform by $Agent1$ in the t state under the π_1 strategy, among them, $i,j \in \{1,2,3,\ldots,a\}$. The cooperation Markov decision process system with the participation of two agents performs iterative calculation according to Formula (18).

Theorem 3. *In a Cooperation Markov Decision Process system in which two agents participate, the two agents perform coordinated actions on the CMDP system, and the social value function converges.*

Proof. According to the iterative evolution process in Equation (18), we can get the calculation idea as follows:

$$\begin{array}{l}
\vec{V}_1^0 = A_1 + r \cdot P_1 \cdot \vec{V}_0, \vec{V}_1^1 = A_2 + r \cdot P_2 \cdot \vec{V}_0, \vec{V}_1 = \alpha \vec{V}_1^0 + \beta \vec{V}_1^1; \\
\vec{V}_2^0 = A_1 + r \cdot P_1 \cdot \vec{V}_1, \vec{V}_2^1 = A_2 + r \cdot P_2 \cdot \vec{V}_1, \vec{V}_2 = \alpha \vec{V}_2^0 + \beta \vec{V}_2^1; \\
\vec{V}_3^0 = A_1 + r \cdot P_1 \cdot \vec{V}_2, \vec{V}_3^1 = A_2 + r \cdot P_2 \cdot \vec{V}_2, \vec{V}_3 = \alpha \vec{V}_3^0 + \beta \vec{V}_3^1; \\
\cdots\cdots \\
\vec{V}_k^0 = A_1 + r \cdot P_1 \cdot \vec{V}_{k-1}, \vec{V}_k^1 = A_2 + r \cdot P_2 \cdot \vec{V}_{k-1}, \vec{V}_k = \alpha \vec{V}_k^0 + \beta \vec{V}_k^1; \\
\vec{V}_{k+1}^0 = A_1 + r \cdot P_1 \cdot \vec{V}_k, \vec{V}_{k+1}^1 = A_2 + r \cdot P_2 \cdot \vec{V}_k, \vec{V}_{k+1} = \alpha \vec{V}_{k+1}^0 + \beta \vec{V}_{k+1}^1.
\end{array} \tag{19}$$

In Formula (19), A_1 and P_1 respectively represent the expected reward value matrices and probability transition matrix obtained by $Agent0$ under a certain state and a known strategy pair (π_0, π_1), among them, $A_1 = diag(P_1 \bullet {R_1}^T)$ represents the probability of the state performing the action multiplied by the reward value for performing the action. A_2 and P_2 respectively represent the expected reward value matrix and probability transition matrix obtained by Agent0 under a certain state and a known strategy pair (π_0, π_1), among them, $A_2 = diag(P_2 \bullet {R_2}^T)$ represents the probability of the state performing the action multiplied by the reward value for performing the action. The dimensions of A_1 and A_2 are $NS0 * NS1$ rows and 1 column, P and Q are square matrices of $NS0 * NS1$ dimension.

Step 1 Calculate $\vec{V}_1$. Given any initial value function $\vec{V}_0$, it can be calculated using Equation (19),

$$\begin{aligned}
\vec{V}_1 = \alpha \vec{V}_1^0 + \beta \vec{V}_1^1 &= \alpha \cdot (A_1 + r \cdot P_1 \cdot \vec{V}_0) + \beta \cdot (A_2 + r \cdot P_2 \cdot \vec{V}_0) \\
&= \alpha A_1 + \beta A_2 + r(\alpha P_1 + \beta P_2) \cdot \vec{V}_0.
\end{aligned} \tag{20}$$

Step 2 Find $\vec{V}_2^0$ and $\vec{V}_2^1$ according to $\vec{V}_1$ in Equation (19)

$$\begin{aligned}
\vec{V}_2^0 = A_1 + r \cdot P_1 \cdot \vec{V}_1 &= A_1 + r \cdot P_1 \cdot (\alpha A_1 + \beta A_2 + r \cdot (\alpha P_1 + \beta P_2) \cdot \vec{V}_0) \\
&= (1 + r\alpha \cdot P_1) A_1 + (r\beta \cdot P_1) A_2 + (r^2 \alpha {P_1}^2 + r^2 \beta \cdot P_1 \cdot P_2) \vec{V}_0,
\end{aligned} \tag{21}$$

$$\begin{aligned}
\vec{V}_2^1 = A_2 + r \cdot P_2 \cdot \vec{V}_1 &= A_2 + r \cdot P_2 \cdot (\alpha A_1 + \beta A_2 + r \cdot (\alpha P_1 + \beta P_2) \cdot \vec{V}_0) \\
&= (r\alpha \cdot P_2) A_1 + (1 + r\beta \cdot P_2) A_2 + (r^2 \alpha P_2 \cdot P_1 + r^2 \beta {P_2}^2) \cdot \vec{V}_0.
\end{aligned} \tag{22}$$

It can be known from Equation (19) that by substituting $\vec{V}_2^0$ in Equation (21) and $\vec{V}_2^1$ in Equation (22) into $\vec{V}_2$, the expression of $\vec{V}_3$ can be derived as follows:

$$\begin{aligned}
\vec{V}_2 &= \alpha \vec{V}_2^0 + \beta \vec{V}_2^1 \\
&= \alpha \cdot [(1 + r\alpha \cdot P_1) A_1 + (r\beta \cdot P_1) A_2 + (r^2 \alpha {P_1}^2 + r^2 \beta \cdot P_1 \cdot P_2) \vec{V}_0] + \\
&\quad \beta \cdot [(r\alpha \cdot P_2) A_1 + (1 + r\beta \cdot P_2) A_2 + (r^2 \alpha P_2 \cdot P_1 + r^2 \beta {P_2}^2) \cdot \vec{V}_0] \\
&= (\alpha + r\alpha^2 \cdot P_1 + r\alpha\beta \cdot P_2) A_1 + (\beta + \alpha r\beta \cdot P_1 + r\beta^2 \cdot P_2) A_2 + \\
&\quad (r^2 \alpha^2 {P_1}^2 + r^2 \alpha\beta \cdot P_1 \cdot P_2 + r^2 \alpha\beta P_2 \cdot P_1 + r^2 \beta^2 {P_2}^2) \cdot \vec{V}_0 \\
&= [\alpha + r\alpha(\alpha P_1 + \beta P_2) A_1] + [\beta + r\beta(\alpha P_1 + \beta P_2] A_2 + r^2 (\alpha P_1 + \beta P_2)^2 \cdot \vec{V}_0.
\end{aligned} \tag{23}$$

Step 3 According to equation (19), $\vec{V}_2$ can be used to find $\vec{V}_3^0$ and $\vec{V}_3^1$

$$\begin{aligned}\vec{V_3^0} &= A_1 + r \cdot P_1 \cdot \vec{V_2} \\ &= A_1 + rP_1 \cdot \left\{ \alpha + r\alpha(\alpha P_1 + \beta P_2)A_1] + [\beta + r\beta(\alpha P_1 + \beta P_2]A_2 + r^2(\alpha P_1 + \beta P_2)^2 \cdot \vec{V_0} \right\} \\ &= (1 + r\alpha P_1 + r^2\alpha^2 \cdot {P_1}^2 + r^2\alpha\beta P_1 \cdot P_2)A_1 + (r\beta P_1 + \alpha r^2\beta \cdot {P_1}^2 + r^2\beta^2 P_1 \cdot P_2)A_2 \\ &\quad + r^3 \cdot P_1 \cdot (\alpha P_1 + \beta P_2)^2 \cdot \vec{V_0},\end{aligned} \tag{24}$$

$$\begin{aligned}\vec{V_3^1} &= A_2 + r \cdot P_2 \cdot \vec{V_2} \\ &= A_2 + r \cdot P_2 \cdot [(\alpha + r\alpha^2 P_1 + r\alpha\beta P_2)A_1 + (\beta + \alpha r\beta P_1 + r\beta^2 P_2)A_2 + r^2(\alpha P_1 + \beta P_2)^2 \cdot \vec{V_0}] \\ &= (r\alpha P_2 + r^2\alpha^2 P_2 \cdot P_1 + r^2\alpha\beta \cdot {P_2}^2)A_1 + (1 + r\beta P_2 + \alpha r^2\beta P_2 \cdot P_1 + r^2\beta^2 \cdot {P_2}^2)A_2) \\ &\quad + r^3 \cdot P_2(\alpha P_1 + \beta P_2)^2 \cdot \vec{V_0}\,.\end{aligned} \tag{25}$$

We can known from Equation (19) that by substituting $\vec{V_3^0}$ in Equation (24) and $\vec{V_3^1}$ in Equation (25) into $\vec{V_3}$, the expression of $\vec{V_3}$ can be further obtained

$$\begin{aligned}\vec{V_3} &= \alpha\,\vec{V_3^0} + \beta\,\vec{V_3^1} \\ &= \alpha[(1 + r\alpha P_1 + r^2\alpha^2 \cdot {P_1}^2 + r^2\alpha\beta P_1 \cdot P_2)A_1 + (r\beta P_1 + \alpha r^2\beta \cdot {P_1}^2 + r^2\beta^2 P_1 \cdot P_2)A_2 + \\ &\quad r^3 \cdot P_1 \cdot (\alpha P_1 + \beta P_2)^2 \cdot \vec{V_0}] + \beta[(r\alpha P_2 + r^2\alpha^2 P_2 \cdot P_1 + r^2\alpha\beta \cdot {P_2}^2)A_1 + \\ &\quad (1 + r\beta P_2 + \alpha r^2\beta P_2 \cdot P_1 + r^2\beta^2 \cdot {P_2}^2)A_2) + r^3 \cdot P_2(\alpha P_1 + \beta P_2)^2 \cdot \vec{V_0}] \\ &= (\alpha + r\alpha^2 P_1 + r^2\alpha^3 \cdot {P_1}^2 + r^2\alpha^2\beta P_1 \cdot P_2 + r\alpha\beta P_2 + r^2\alpha^2\beta P_2 \cdot P_1 + r^2\alpha\beta^2 \cdot {P_2}^2)A_1 + \\ &\quad (r\alpha\beta P_1 + \alpha^2 r^2\beta \cdot {P_1}^2 + r^2\alpha\beta^2 P_1 \cdot P_2 + \beta + r\beta^2 P_2 + \alpha r^2\beta^2 P_2 \cdot P_1 + r^2\beta^3 \cdot {P_2}^2)A_2 + \\ &\quad [\alpha r^3 \cdot P_1 \cdot (\alpha P_1 + \beta P_2)^2 + \beta r^3 \cdot P_2(\alpha P_1 + \beta P_2)^2] \cdot \vec{V_0} \\ &= [\alpha + r\alpha(\alpha P_1 + \beta P_2) + r^2\alpha(\alpha P_1 + \beta P_2)^2]A_1 + [\beta + r\beta(\alpha P_1 + \beta P_2) + r^2\beta(\alpha P_1 + \beta P_2)^2] \cdot \\ &\quad A_2 + r^3(\alpha P_1 + \beta P_2)^3 \cdot \vec{V_0}\,.\end{aligned} \tag{26}$$

Finally, from the expressions of $\vec{V_1}$, $\vec{V_2}$ and $\vec{V_3}$, it is not difficult to find the laws through observation. The value function $\vec{V_k}$ at time k and the value function $\vec{V_{k+1}}$ at time $k+1$ can be deduced by the following Formulas (27) and (28) as shown.

$$\begin{aligned}\vec{V_k} &= \alpha\,\vec{V_k^0} + \beta\,\vec{V_k^1} \\ &= [\alpha + r\alpha(\alpha P_1 + \beta P_2) + r^2\alpha(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}\alpha(\alpha P_1 + \beta P_2)^{k-1}]A_1 + \\ &\quad [\beta + r\beta(\alpha P_1 + \beta P_2) + r^2\beta(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}\beta(\alpha P_1 + \beta P_2)^{k-1}]A_2 + \\ &\quad r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V_0},\end{aligned} \tag{27}$$

$$\begin{aligned}\vec{V_{k+1}} &= \alpha\,\vec{V_{k+1}^0} + \beta\,\vec{V_{k+1}^1} \\ &= [\alpha + r\alpha(\alpha P_1 + \beta P_2) + \cdots + r^{k-1}\alpha(\alpha P_1 + \beta P_2)^{k-1} + r^k\alpha(\alpha P_1 + \beta P_2)^k]A_1 + \\ &\quad [\beta + r\beta(\alpha P_1 + \beta P_2) + + \cdots + r^{k-1}\beta(\alpha P_1 + \beta P_2)^{k-1} + r^k\beta(\alpha P_1 + \beta P_2)^k]A_2 + \\ &\quad r^{k+1}(\alpha P_1 + \beta P_2)^{k+1} \cdot \vec{V_0}\,.\end{aligned} \tag{28}$$

Subtract the two equations to calculate the value function difference between time k and $k+1$:

$$\begin{aligned}&\vec{V_{k+1}} - \vec{V_k} \\ &= [r^k\alpha(\alpha P_1 + \beta P_2)^k]A_1 + r^k\beta(\alpha P_1 + \beta P_2)^k]A_2 + r^{k+1}(\alpha P_1 + \beta P_2)^{k+1} \cdot \vec{V_0} - r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V_0} \\ &= [r^k\alpha(\alpha P_1 + \beta P_2)^k]A_1 + r^k\beta(\alpha P_1 + \beta P_2)^k]A_2 + [r^k(\alpha P_1 + \beta P_2)^k \cdot (r(\alpha P_1 + \beta P_2) - E)] \cdot \vec{V_0} \\ &= r^k(\alpha P_1 + \beta P_2)^k \cdot \left\{ \alpha A_1 + \beta A_2 + [r(\alpha P_1 + \beta P_2) - E] \cdot \vec{V_0} \right\}.\end{aligned}$$

P_1 and P_2 are two probability transition matrices, $\alpha + \beta = 1$, thus, $\alpha P_1 + \beta P_2$ is a probability transition matrix and the k-th power of the probability transition matrix is still the probability matrix, so $(\alpha P_1 + \beta P_2)^k$ is a probability transition matrix. Because of discount factor $0 < r < 1$, then r^k decreases monotonically with the increase of k. When k is large enough, $r^k \to 0$. That is to say:

$$\lim_{k\to\infty}(\vec{V}_{k+1} - \vec{V}_k) = \lim_{k\to\infty} r^k(\alpha P_1 + \beta P_2)^k \cdot \{\alpha A_1 + \beta A_2 + [r(\alpha P_1 + \beta P_2) - E] \cdot \vec{V}_0\} = 0.$$

So, for any given strategy pair (π_0, π_1), $\vec{V}_{k+1} = \vec{V}_k$, in the $CMDP$ system introduced in this paper, the final value function V will converge. If the discount factor is larger, the convergence speed is slower, the smaller the discount factor is, the faster the convergence speed is. □

Theorem 4. *In the Cooperation Markov Decision Process system, the final value $\vec{V}_k$ of the value function is independent of the initial value of $\vec{V}_0$.*

Proof. From Theorem 3, the value function at time can be expressed, and the expression (27) can be further simplified to obtain:

$$\begin{aligned}
\vec{V}_k &= \alpha\,\vec{V}_k^0 + \beta\,\vec{V}_k^1 \\
&= [\alpha + r\alpha(\alpha P_1 + \beta P_2) + r^2\alpha(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}\alpha(\alpha P_1 + \beta P_2)^{k-1}]A_1 + \\
&\quad [\beta + r\beta(\alpha P_1 + \beta P_2) + r^2\beta(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}\beta(\alpha P_1 + \beta P_2)^{k-1}]A_2 + r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V}_0 \\
&= (\alpha A_1 + \beta A_2) + [r(\alpha P_1 + \beta P_2) \cdot (\alpha A_1 + \beta A_2)] + [r^2(\alpha P_1 + \beta P_2)^2 \cdot (\alpha A_1 + \beta A_2)] + \\
&\quad + \cdots + [r^{k-1}(\alpha P_1 + \beta P_2)^{k-1} \cdot (\alpha A_1 + \beta A_2)] + r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V}_0 \\
&= [E + r(\alpha P_1 + \beta P_2) + r^2(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}(\alpha P_1 + \beta P_2)^{k-1}] \cdot (\alpha A_1 + \beta A_2) + \\
&\quad r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V}_0 .
\end{aligned}$$

Obviously, P_1 and P_2 are two probability transition matrices, $\alpha + \beta = 1$, thus $(\alpha P_1 + \beta P_2)$ is a probability transition matrix, $\forall k \in N$, $(\alpha P_1 + \beta P_2)^k$ is a probability transition matrix. r is the discount factor and $0 < r < 1$, then with the increase of k, the r^k decreases monotonically. When k is large enough, $r^k \to 0$, that $\lim_{k\to\infty} r^k = 0$, so $\lim_{k\to\infty} r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V}_0 = \vec{0}$, we can conclude,

$$\begin{aligned}
\lim_{k\to\infty} \vec{V}_k &= \lim_{k\to\infty} [E + r(\alpha P_1 + \beta P_2) + r^2(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}(\alpha P_1 + \beta P_2)^{k-1}] \cdot (\alpha A_1 + \beta A_2) + \\
&\quad r^k(\alpha P_1 + \beta P_2)^k \cdot \vec{V}_0 \\
&= \lim_{k\to\infty} [E + r(\alpha P_1 + \beta P_2) + r^2(\alpha P_1 + \beta P_2)^2 + \cdots + r^{k-1}(\alpha P_1 + \beta P_2)^{k-1}] \cdot (\alpha A_1 + \beta A_2).
\end{aligned}$$

Therefore, the choice of the initial function $\vec{V}_k$ does not affect the convergence value of the final value function $\vec{V}_0$. □

4.3. Algorithm for Optimal Strategy Pairs in Cooperation Type CMDP System

From the theoretical results in Section 4.2, it can be known that under the strategy pair (π_0, π_1), the V-function generated by the iteration of Equation (18) is recorded as $V^{(\pi_0,\pi_1)}$ to form a stable value. Similarly, according function V and improve (π_0, π_1) to obtain (π_0', π_1') by the following greedy method.

$$\begin{aligned}
\pi_0'(s,t) &= \arg\max_a \Sigma_{(s',a')\in S\times A} P_0(s,t,a,s',a')[R_0(s,t,a,s',a') + \gamma \cdot V^{(\pi_0,\pi_1)}(s',t)] \\
\pi_1'(s,t) &= \arg\max_a \Sigma_{(t',a')\in S\times A} P_1(s,t,a,t',a')[R_1(s,t,a,t',a') + \gamma \cdot V^{(\pi_0,\pi_1)}(s,t')] .
\end{aligned} \tag{29}$$

In order to obtain the algorithm of the optimal strategy pair (π_0^*, π_1^*) in the Cooperation Markov Decision Process system. We modeled the optimal strategy of a single agent. Let Formula (18) be the iterative calculation formula of the "evolution module" part of the algorithm, Formula (29) be the strategy improvement method of the "strategy improvement module" in the algorithm. According the method of solving the single-agent Markov decision process system's optimal strategy, this section presents the algorithm for solving the optimal strategy of the joint type cooperation Markov decision process system, the optimal strategy pair is (π_0^*, π_1^*).

The detail code of Algorithm 1 can see in Appendix A. The execution process of the above algorithm is shown in the following Figure 4:

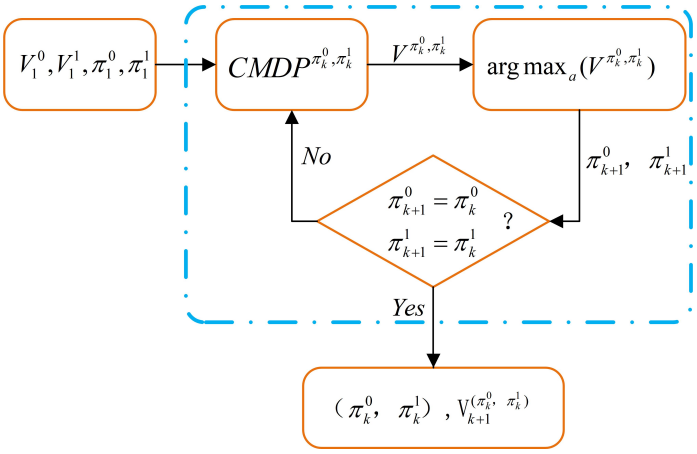

Figure 4. Optimal Strategy Solution Algorithm Framework of *CMDP* System.

Algorithm 1 The Optimal Strategy Pairs

Input: input *CMDP* quintuple $< S,\ A,\ \{Agent_0, Agent_1\},\ \{T_0,\ T_1\},\ \{R_0,\ R_1\} >$: error parameter , $\varepsilon > 0$ balance parameter $0 < \alpha < 1$, discount factor $0 < \gamma < 1$.

Output: output Optimal strategy pair $(\pi_*^0, \pi_*^1) = (\pi_k^0, \pi_k^1)$, optimal value function $V_{k+1} = V_{k+1}^{(\pi_*^0, \pi_*^1)}$.

1. Initialization V-function V_1^0, V_1^1, take the initial value randomly. Such as ${V_1}^1(s) = 0\ (s \in S)$; suppose $V_1 = \alpha V_1^0 + (1-\alpha) V_1^1$;
2. Use Equation (19) to greedy improvement strategy pair (π_1^0, π_1^1): $(V_1^{(\pi_1^0, \pi_1^1)} = V_1)$;
3. Use the updated strategy pair (π_1^0, π_1^1)in step 2 and Formula (18) to find the V-function V_2;
4. Repeat steps 2, 3;
5. Step $k+1$: assuming $(\pi_k^0,\ \pi_k^1\ )$, $V_k^{(\pi_k^0,\ \pi_k^1\ )}$ has been obtained, at this step, do the following two steps of calculation:
   (a) Evolutionary calculation of *CMDP* system: find V_{k+1} by $(\pi_k^0,\ \pi_k^1\ )$ and Formula (18), when $||V_{k+1} - V_k|| < \varepsilon$, defined $V_{k+1}^{(\pi_k^0,\ \pi_k^1\ )}(s,t) = V_k(s,t)\ \ ((s,t) \in S \times S)$.
   (b) Greedy computing $(\pi_{k+1}^0,\ \pi_{k+1}^1\ )$:

$$\pi_{0,k+1}(s,t) = \arg\max_a \Sigma_{(s',a') \in S \times A} P_0(s,t,a,s',a')[R_0(s,t,a,s',a') + \gamma \cdot V_0^{(\pi_{0k},\ \pi_{1k})}(s',t)]\ ,$$
$$\pi_{1,k+1}(s,t) = \arg\max_a \Sigma_{(t',a') \in S \times A} P_1(s,t,a,t',a')[R_1(s,t,a,t',a') + \gamma \cdot V_0^{(\pi_{0k},\ \pi_{1k})}(s,t')]\ .$$

6. If $\pi_{k+1}^0 = \pi_k^0$, $\pi_{k+1}^1 = \pi_k^1$, terminate calculation.
7. **Return** result.

5. An Application Example of Cooperation Markov Decision Process System

Reinforcement learning is one of the most concerned topic in artificial intelligence after deep learning. Reinforcement learning agents interact with the environment and take actions that receive feedback similar to learning strategies in medicine. In fact, many applications of reinforcement learning in health care have to do with finding the best treatment strategy. Many reading learning software also use intensive learning to provide customized learning materials and content for the tutoring system to meet the specific needs of students. The recommendation system of e-commerce changes the recommendation strategy in real time according to the feedback of users (ignore, click and buy).

The core problem of multi-agent system research is to seek to establish an effective cooperation mechanism, so that multi-agent with simple function and independent function can complete complex target tasks or solve complex problems through negotiation, coordination and cooperation. In the multi-agent system, each agent learns and improves its own strategy by interacting with the environment to obtain the reward value, so as to obtain the optimal strategy under the environment is the process of multi-agent reinforcement learning. The transfer of multi-agent state depends on the actions of all agents, and the joint actions determine the transfer of the whole system. The return of each agent is not only related to its own actions, but also the actions of other agents will affect the return of the current agent. If multi-agent are involved, there are three types of cooperative game, competition game and mixed game combined between agents in the same environment.

The proposed joint multi-agent in this paper only analyzes the relationship between two agents, two agents take interactive actions and complete cooperation to achieve the goal together. When intelligence agent B observed intelligence agent A take an action, intelligence agent B will perform an action. At the same time, the next action of agent A also depends on the agent B's action. Repeat the process of perform action above, the behavior of the two agents influence each other and evolve and learn in the $CMDP$ system of a given strategy pair to find a pair of optimal strategies finally.

Background: a couple found a gold bar not far from home at the same time, this gold bar needs two people together, one carried one end to carry home, they want to carry the gold bar to home, they need avoid the trap and then everyone arrives at the side of the gold bar, after lifting the gold bar, they also have to bypass the obstacles in front of the home, finally, the gold bar is successfully transported home.

The background is transformed into the environment as shown in the Figure 5, the simulation is carried out on the figure. Two people are compared to two agents respectively. In addition, the figure can only move up, down, or left, and each position can only accommodate one agent. In this experiment, agents in the same environment, through mutual cooperation, the gold bars to the designated location. The whole process is divided into the following three steps:

Step 1 Bypass the trap. When passing through the trap area, each position can only accommodate one agent, two agents cannot be in the same position at the same time, so they need to interact with each other which agent passe first. When agent B observes that agent A is going to pass through the trap position, agent B cannot pass through the trap position.

Step 2 Reach the designated position. The two agents must reach the two ends of the gold bar to pick up the gold bar, so the two agents can not reach the same place at the same time. When one of the agents reaches one end of the gold bar, the other must walk to the other end of the gold bar before the two agents can work together to pick up the gold bar.

Step 3 Delivery of gold bar. Because the gold bar has a certain length, in the process of transporting the gold bar to the designated position, the two agents are at different positions. In the process of transporting the gold bar, the two agents also need to consider how to bypass the obstacles and the positions of the two agents. The two agents need to cooperate and interact to complete the task.

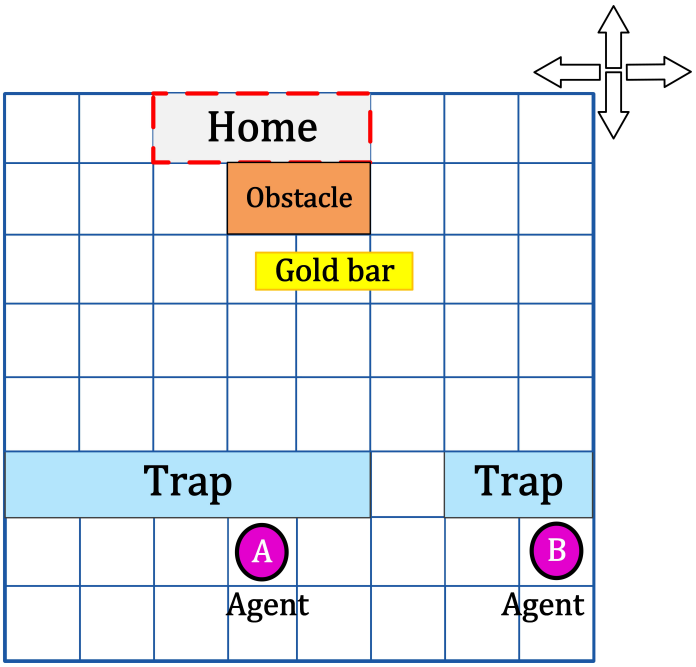

Figure 5. The environment of two agents in this example.

In the process of gold bar transportation, through continuous exploration and trial and error in the environment, the two intelligence agent will find the best way to move. Finally, the two agents cooperate through the trap to reach the two ends of the gold bar, and then bypass the obstacles to successfully transport the gold bar home. In such an example, each agent can only move up, down, left, and right to adjacent positions in the environment, and the sum of the probabilities of moving to all adjacent positions is equal to 1. The agent obtains certain benefits by moving the position. In order to avoid falling into traps and avoiding obstacles, the probability and benefits of moving to the vicinity of traps and obstacles are also relatively small compared to other locations. The goal is that through constant trial and error and learning in the environment, the two agents can take an optimal way of movement to get the most benefit. We express the probability and reward value of moving adjacent positions in this problem with a square matrix, and the probability and reward value of the joint action of two agents are the tensor product of two probability matrices and two reward matrices. Using the algorithm 1 in this article, given an initial strategy, the system will obtain a value function under this strategy. Under the value function at this time, the agent performs all joint actions and selects the joint action that enables the system to obtain the highest reward value. Use this to modify the initial strategy, and then obtain a new value function under the modified strategy. According to this method, iteratively modify the strategy and value function, and finally evolve to learn a strategy that can obtain the maximum reward value. That is, the optimal strategy.

Conversely, if the two agents do not cooperate in any of the steps described in the appeal, it is difficult to successfully transport the gold bar home. Firstly, there will be a conflict when they bypass the trap, both agents cannot simultaneously adopt their own optimal way to pass the trap position. Secondly, after passing through the trap, then something happens that the two agents are on the same side of the bar. After that, the two agents may not move in the same direction, which will also cause the task to fail. So in such an instance, the interaction between two agents becomes extremely important.

6. Conclusions

This paper first proves the convergence of the value function of the monomer Markov decision process system. Furthermore, a cooperation Markov decision process system of two agents is introduced, which is suitable for an evolutionary learning system of cooperative (or antagonistic) decision between two agents. The main difference between cooperation and confrontation lies in the

way the value function is defined and the way of combination (or restriction). In this paper, only $CMDP$ system of cooperative type is considered. In this kind of learning model, agents alternately execute behaviors, seek the optimal criterion with social value, find the optimal strategy (π_0^*, π_1^*), and jointly complete the target task. This paper presents an algorithm for finding the optimal strategy pair (π_0^*, π_1^*), whose fundamental task is to find a pair of optimal strategies to form an evolutionary system $CMDP^{(\pi_0^*, \pi_1^*)}$. Finally, the convergence of the value function in system $CMDP$ is proved, and the convergence value is not affected by the value of the initial value function. This paper only introduces the cooperation Markov decision process system involving two agents, but many problems in reality involve more agents. As the number of agents increases, the interaction between agents in the system becomes more complex, and the difficulty of analysis also increases exponentially. Based on the fact that deep learning has strong perceptual ability, but lacks certain decision-making ability, while reinforcement learning has decision-making ability but short of doing anything about perceptual problems. Therefore, combining the two and complementing each other's advantages, a new concept–deep reinforcement learning is proposed. so we will use the next deep reinforcement learning method to study the properties of the cooperation Markov decision process system.

Author Contributions: Formal analysis, X.M.; Investigation, X.M. and D.X.; Methodology, X.M. and D.X.; Software, X.M.; Writing—Original Draft, X.M.; Writing—Review & Editing, X.M., Z.F. and D.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the National Natural Science Foundation of China under grant numbers Nos. 61762019, 61862051.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The algorithm in the simulation experiment in Section 4.2 of this paper is completely suitable for solving the problem of two agents. The specific implementation of the algorithm, the data and code of all the experiments in the article can be found in the following link: https://github.com/moxiaoling/Cooperation-Markov-Decision-Process-System.

References

1. Shalev-Shwartz, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: New York, NY, USA, 2014; pp. 19–29. [CrossRef]
2. Tran, T.; Cohen, R. A reputation-oriented reinforcement learning approach for agents in electronic marketplaces. *Am. Assoc. Artif. Intell.* **2002**, *18*, 550–565. [CrossRef]
3. Jonathan, B.; Andrew, T.; Lex, W. Reinforcement learning and chess. In *Machines that Learn to Play Games*; Nova Science Publishers, Inc.: New York, NY, USA, 2001; pp. 91–116. [CrossRef]
4. Liebman, E.; Stone, P. DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, Istanbul, Turkey, 4–8 May 2015; Volume 1, pp. 591–599. [CrossRef]
5. Van Hassel, H. Reinforcement Learning in Continuous State and Action Spaces. In *Reinforcement Learning*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 207–251. [CrossRef]
6. Iwata, K.; Ikeda, K.; Sakai, H. A statistical property of multiagent learning based on Markov decision process. *IEEE Trans. Neural Netw.* **2006**, *17*, 829–842. [CrossRef] [PubMed]
7. Julien, P.; Bilal, P.; Matthieu, G.; Bruno, S.; Olivier, P. Softened approximate policy iteration for Markov games. In Proceedings of the 33rd International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1860–1868. [CrossRef]
8. Littman, M.L . Reinforcement learning improves behaviour from evaluative feedback. *Nature* **2015**, *521*, 445–451. [CrossRef] [PubMed]
9. Leibo, J.Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; Graepel, T. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), Sao Paulo, Brazil, 8–12 May 2017.

10. Ligtenberg, A.; Wachowicz, M.; Bregt, A.K.; Beulens, A.; Kettenis, D.L. A design and application of a multi-agent system for simulation of multi-actor spatial planning. *J. Environ. Manag.* **2004**, *72*, 43–55. [CrossRef] [PubMed]
11. Bloembergen, D.; Tuyls, K.; Hennes, D.; Kaisers, M. Evolutionary Dynamics of Multi-Agent Learning: A Survey. *J. Artif. Intell. Res.* **2015**, *53*, 659–697. [CrossRef]
12. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the Eleventh International Conference on International Conference on Machine Learning, San Francisco, CA, USA, 10–13 July 1994; pp. 157–163. [CrossRef]
13. Littman, M.L. Friend-or-foe Q-learning in general-sum games. In Proceedings of the Eighteenth International Conference on Machine Learning, San Francisco, CA, USA, 26 August 2001; Volume 1, pp. 322–328. [CrossRef]
14. Puppala, S.N.; Gordin, S.M. Shared memory based cooperative coevolution. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA, 4–9 May 1998; pp. 570–574. [CrossRef]
15. Mahmoud, S.; Miles, S.; Luck, M. Cooperation emergence under resource-constrained peer punishment. In Proceedings of the 2016 International Conference on Autonomous Agents Multiagent Systems, Richland, SC, Singapore, 9–13 May 2016; pp. 900–908. [CrossRef]
16. Littman, M.L. Value-function reinforcement learning in Markov games. *Cogn. Syst. Res.* **2001**, *1*, 55–66. [CrossRef]
17. Hamidi, H.; Kamankesh, A. An Approach to Intelligent Traffic Management System Using a Multi-agent System. *Int. J. Intell. Transp. Syst. Res.* **2018**, *16*, 1–13. [CrossRef]
18. Cruz, D.L.; Yu, W. Multi-agent path planning in unknown environment with reinforcement learning and neural network. In Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, 5–8 October 2014; pp. 3458–3463. [CrossRef]
19. Wang, C.; Jing, N.; Li, J. An algorithm of cooperative multiple satellites mission planning based on multi-agent reinforcement learning. *J. Natl. Univ. Def. Technol.* **2011**, 33, 53–58. [CrossRef]
20. Zhang, W.; Ma, L.; Li, X. Multi-agent reinforcement learning based on local communication. *Clust. Comput.* **2018**, *22*, 1–10. [CrossRef]
21. Zawadzki, E.; Lipson, A.; Leyton-Brown, K. Empirically evaluating multiagent learning algorithms. *arXiv* **2014**, arXiv:1401.8074.
22. Bošanský, B.; Lisý, V.; Lanctot, M.; Čermák, J.; Winands, M.H. Algorithms for computing strategies in two-player simultaneous move games. *Artif. Intell.* **2016**, *237*, 1–40. [CrossRef]
23. Heris, S.; Kalami, M.; Mohammad-Bagher, S.; Naser, P. Using Control Theory for Analysis of Reinforcement Learning and Optimal Policy Properties in Grid-World Problems. In *International Conference on Intelligent Computing (ICIC'09)*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 276–285. [CrossRef]
24. Gheorghe, C.; Doina. P. Optimal policy switching algorithms for reinforcement learning. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, ON, Canada, 10–14 May 2010; Volume 1–3, pp. 709–714. [CrossRef]