Bankruptcy-Evolutionary Games based Solution for the Multi-agent Credit Assignment Problem

Hossein Yarahmadi[1], Mohammad Ebrahim Shiri[1*†], Hamidreza Navidi[2†], Arash Sharifi[1†] and Moharram Challenger[3†]

[1*]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
[2]Department of Mathematics and Computer Science, Shahed University, Tehran, Iran.
[3]Department of Computer Science, University of Antwerp and Flanders Make, Flanders, Belgium.

*Corresponding author(s). E-mail(s): shiri@aut.ac.ir;
Contributing authors: hs.yarahmadi@srbiau.ac.ir; navidi@shahed.ac.ir; a.sharifi@srbiau.ac.ir; moharram.challenger@uantwerpen.be;
†These authors contributed equally to this work.

Abstract

Multi-Agent Systems (MASs) are symbols of Distributed Artificial Intelligence (DAI), made of entities called agents. One of the challenging problems about MASs is their learning, which often occurs in the form of Reinforcement Learning (RL). Multi-agent Credit Assignment problem (MCA) is a part of Multi-Agent Reinforcement Learning (MARL) process addressing how the global reward, that the MAS receives from the environment, should be distributed among the agents. In this paper, to solve this problem and enhance the efficiency of the MAS, the Maximum Performance Power (MPP) constraint is introduced. Due to this constraint, the MCA will turn into a bankruptcy problem. There are numerous instances of the answers to solve MCA as a bankruptcy problem, which makes it difficult to find the best answer. To solve the MCA, which has now become a bankruptcy problem, we use the Evolutionary

Game Theory (EGT). In this game, the players are answering instances for MCA that were obtained in the previous step. These players apply a mixed strategy to distribute the global rewards among the agents in a way that the payoff of the game will be a global reward that the environment returns to the MAS due to the interaction of the agents. To evaluate the proposed method, six parameters including the group learning rate, confidence, expertness, efficiency, certainty, and correctness were used. The simulation results indicated a better performance in five parameters, and poorer performance only in one of the parameters.

Keywords: Multi-agent Systems, Credit Assignment Problem, Reinforcement Learning, Bankruptcy Game, Evolutionary Game, Game Theory

1 Introduction

Multi-Agent Systems (MASs) are symbols of Distributed Artificial Intelligence (DAI) [1] that are used to model and solve complex problems and systems such as aerial robots [2], Vehicle speed management [3], and so on. MASs are made of smaller entities called agents that make them suitable for bottom-up problem solving [4].

One of the most critical and challenging problems for MASs is their learning [5], which is usually occurs based on Reinforcement Learning [6]. The basis of RL is to receive rewards/punishments from the environment with which agent(s) interact, so that the agent(s) try to increase their received reward from the environment [7]. Each agent, based on its received reward from the environment, selects its next state and action in a way that ultimately leads to the maximum received reward from the environment. To achieve this purpose, in the RL, the reward and the transition functions are used to determine the reward and select the next state [8]. RL can usually be examined in two ways:Single-agent RL and Multi-agent RL.

In single-agent RL, the agent interacts with the environment, and based on the action that it performs in the environment, it receives reward/punishment from the environment. Based on this reward/punishment and according to the reward and the transition functions, the agent selects the next state in a way that leads to an increase in the reward [9].

In Multi-Agent Reinforcement Learning (MARL), because there are several agents work together in the environment [6], designing the reward and transition functions is more complicated than single-agent RL. In MARL, agents interact with the environment and a result of these partial interactions, the environment returns a global reward to the MAS, which is a resultant of the individual rewards/punishments of each agent. The MAS now faces the question of how to distribute this global reward among agents. This problem is known as the MCA [10], and various methods have been proposed to solve it [10–12].

In general, these methods can be classified in "fairly" and "efficient" groups. The purpose of the "fair" view is to assign rewards according to the impact of the agent on the global reward. In contrast, in the second view, the goal is to increase the efficiency of the MAS regardless of whether the reward distribution is fair or not. Most studies that have been done in this field have looked for methods to solve the MCA from the first perspective [13, 14] and have paid less attention to the second perspective.

In this paper, our goal is to propose a method based on the bankruptcy game and the Evolutionary Game Theory (EGT) to solve the MCA from the second perspective. Given that MASs are appropriate tools for real-world modeling, it should be possible to bring this modeling closer to the real world [15]. To achieve this purpose, we address two issues in this paper.

The first is the operating environment, which is a Multi-score Puzzle (MsP). MsP is a puzzle in which solving each part has a different point than the other part.

The second issue is a constraint called the Task Start-up Threshold (TST). This constraint causes, like many real-world affairs, an agent to start working if it receives a suitable reward. Otherwise, it will not start working.

In this paper, we turn the TST into the Maximum Performance Power (MPP) constraint. The existence of this constraint means that each agent will have its best performance, in order to obtain the highest reward, when it receives a reward according to the MPP. In contrast, if it receives less reward than the MPP, following this reward, it will have the performance and receives the reward.

Given that the global reward that must be distributed among the agents is less than the sum of the MPP of the agents, we are facing a bankruptcy problem. The bankruptcy problem, which was first posed by O'Neill [16], addresses distributing a finite asset among claimants whose total demand is greater than the intended asset. So, in MCA, considering the global reward as a finite asset that must be distributed among agents as claimants, and by considering the agent's MPP as the claimant's request amount, MCA will be turned into a bankruptcy game. Many methods, such as P, AP, CEA, CEL, Talmud, and so on, have been proposed to solve the bankruptcy problem [17], and it is still an interesting and practical problem for researchers [18]. Thus, to solve the MCA, turning the MCA into a bankruptcy game using the MPP is the first contribution of this paper.

On the other hand, to solve the MCA as a bankruptcy problem, there are several answer instances, and each of them has different effects on the agent's performance. Therefore, finding the best answer instance is not easy. To solve such problems, EGT can be used, which is another branch of game theory and is derived from Darwin's theory of evolution [19, 20].

Therefore to solve the MCA that is now mapped to a bankruptcy game, in a way that will result in more rewards from the environment by the MAS, we turn it into an Evolutionary Game (EG). In this EG, the instances of answers for global reward distribution among the agents are considered as players of this game. Also, different methods of solving the bankruptcy problem are considered as game strategies, where each player uses a mixed strategy to solve the MCA problem as a bankruptcy problem. In addition, the reward that the MAS due to this reward distribution, and following that MAS's action, received from the environment is considered as the payoff of this EG. Given the mentioned process, solving the MCA as an EG is introduced as the subsequent contribution of this paper.

We will use this EG to train the critic. After this training, the critic will be able to distribute the received reward from the environment among the agents in a way that, according to the criteria defined in [10], improves MAS performance to be rewarded for the highest. Finally, critic training using the EGT to assign a global reward to agents is introduced as the last contribution of this paper.

The rest of the paper organized as follows: In Section 2, we review the related work. The required preliminaries and definitions are introduced in Section 3. The proposed bankruptcy-evolutionary game based method for solving the MCA problem is presented in Section 4. The results of the simulation of the proposed method and its comparison with the existing methods can be found in Section 5. Finally, Section 6 provide conclusions as well as future work.

2 Related Work

RL stems from learning about living things, such as animals, in nature [21]. This type of learning is based on their performance in nature and receiving rewards or punishments in proportion with that performance [22]. One of the goals and functions of a MAS is to be placed in unknown environments and learning in those environments [23]. Given the ability and power of the RL in unknown environments, this method was used for autonomous systems, including MAS [24]. The problem of RL in MAS is known as MARL in literature [25, 26]. MARL consists of three parts, which are as follows,

1. Interaction of agents with the environment with attention to their states and rewards
2. Receiving the resultant vector of rewards/punishments of each agent interacting with the environment;
3. Distribution of the received output vector between the agents.

In other words, In MARL, agents interact with each other and with the environment in MAS. This interaction with the environment causes it to return a global reward to the MAS owing to the agents' performance, which must be appropriately distributed among the agents. Plenty of studies have been presented on credit assignment problem. Based on the classification made by Rahaie [10], the credit assignment problem in RL can be divided into two general categories: single-agent credit assignment and multi-agent credit assignment. The single-agent credit assignment problem can be classified into three modes of Temporal CA [27], structural CA [28], and social CA [29].

The second main type of credit assignment problem in this categorization is the multi-agent credit assignment, which is a part of the MARL process. In the related literature, this problem is considered as the "MCA" [10, 30].

Skinner [31] was the first to address the problem. He asserted that the success of a system depended on the cooperation of its components, and accordingly introduced the MCA problem. Solving the MCA problem has been considered from several points of view. From a point of view, the MCA can be divided into two categories [32]:

1. Shared reward approach
2. Local reward approach

In the shared reward approach, the returned resultant vector is equally distributed among the agents. This approach is unfair as it ignores the role of individual agents in achieving the goal. Besides, since there is an even distribution among the agents and features such as the agent's knowledge are not considered, the approach is inefficient.

In the local reward approach, the received global reward from the environment is distributed based on how which a given agent participates in achieving the MAS. Compared to the previous approach, this one seems fair, but it is difficult to determine the level of the agent's participation. Methods have been proposed to assess this level of participation, including the Kalman filter [33], differential reward [34], and the use of the Shapley method [35]. Moreover, this method merely deals with the role and presence of an agent in achieving the goal and ignores the success rate of the system. Therefore, it may decrease the efficiency of the MAS.

From another point of view, the MCA problem may be considered implicitly or explicitly [36]:

1. Explicit Credit Assignment
2. Implicit Credit Assignment

Explicit credit assignment introduces strategies for assigning rewards to the agents, at least locally optimized. In contrast, the implicit methods do not purposefully assess the agent actions based on a specific baseline, but they use former methods to assign credits in such a way that the agents' learning from the distribution of the global rewards among them occurs based on their individual functions.

One of the first works in the field of MCA was assigning rewards based on knowledge so that rewards were distributed among agents based on their knowledge [30]. The agents' knowledge was extracted according to specific criteria and then arranged accordingly. In [10], two methods were proposed to solve the MCA problem, which was an extension of the method presented in [30]. These two methods were:

- History-based method

- Ranking method

In the history-based method, the assignment of credits to the agents is based on prior knowledge of agents. In this method, an undirected graph is used for modeling and the credit assignment to the agents in MAS by the critic.

The method is implemented based on the fact that first the interaction between the agents and the environment is mapped to an undirected graph. In this graph, each node initially contains a set of indefinite values, which are gradually completed over time. Then, as the values of the variables are determined, the environment model is constructed. The critic can then decide on assigning credits between the agents.

If MARL is examined as model-based and model-free, then this method will turn out to be model-based. This method exploits the history of agents' interactions with the environment to distribute the reward. Therefore, it needs to build the model based on a graph of the interactions between the agent and environment. Since in this method, the number of actions can be large, the number of nodes is increased, and consequently, the graph will be difficult and time-consuming to process. Thus, one can say that this method is weak in terms of scalability.

The following method to solve the MCA problem proposed in [10] is the ranking method, which is an extension of the knowledge-based method [30]. In this method, the most essential criterion for rewards distribution among agents is their knowledge. In this method, the agents' knowledge is initially assessed based on the criteria introduced in, [30]. When the knowledge of the agents has been evaluated, the next step is to rank the agents according to their knowledge. The final step in this method will be to distribute rewards among the agents based on their rank.

In this method, it is possible that the critic will assign the whole reward received from the environment to the most knowledgeable agent, and no reward will be allocated to other agents. This will cause the MAS to turn into a single agent system, and consequently, other agents will not play any role in solving the problem, which contradicts the very nature of MAS.

Besides, this method also tries to distribute the reward based on the participation of agents in the MAS, and the efficiency of the MAS is not much considered.

The next method for solving the MCA problem is the dynamic method [37]. In this method, the share of each agent in the global reward is determined based on one of the criteria presented in [37]. Although it is attempted to use one of the criteria for the agent's knowledge, this reward will be merely based on only one parameter it may not be accurate enough in assigning the reward to the agent in question. Therefore, this method may not only lead to unfair distribution of rewards among the agents, but also may reduce the efficiency of MAS.

3 Preliminaries

3.1 MAS and MCA

MAS consists of several agents that work together to achieve a goal. Therefore, if the MAS consists of K agents, and we represent each agent with Ag and the i$^{\text{th}}$ agent with Ag_i, then we have:

$$MAS = \{\text{Ag}_1, \text{Ag}_2, \ldots, \text{Ag}_K\}$$

The credit assignment problem in MAS, which is called MCA, is a part of the MARL process. In this process, the agents of MAS interact with the environment and the environment returns a global reward to the MAS as a result of these interactions. Now the critic must distribute this global reward at time t, which we represent with $Reward^{\text{t}}$, among the agents.

If r_i^t indicates the share of each agent of the global reward at time t, then the critic's object is to find the vector $R^t\left(r_1^t, r_2^t, \ldots, r_K^t\right)$. In other words, if we consider K as the number of agents, which the global reward is to be distributed between them, then the vector R^tdenotes the share of each agent of the global reward.

In the MARL process, each agent acts an action in the environment; as a result, the environment returns a partial reward/punishment for this action of the agent.

The resultant of these partial rewards/punishments will be delivered to MAS in the form of a global reward. After this stage, the critic distributes the global reward among the agents, and each agent acts a new action in order to increase the received reward, taking into the received reward and the state in which it finds itself.

Therefore, each instance of global reward distribution between agents causes different performance and result, will create new and different rewards.

So, if the action of each agent, due to the reward r_i at time t, is indicated with a_i^t, then the vector $\text{Act}^t\left(a_1^t, a_2^t, \ldots, a_K^t\right)$ denotes the action of MAS in the environment due to the reward vector $R^t(r_1^t, r_2^t, \ldots, r_K^t)$. If the Mas(.) function denotes the action of the agents due to the reward vector $R^t$and En(.) function represents the reward that the environment returns to the MAS due to the action of the agent (s), then we have:

$$Mas : R \rightarrow Act$$

$$En : Act \rightarrow \mathbb{R}$$

$$\text{En}\left(\text{Mas}\left(R\right)\right) = Reward \qquad (1)$$

As mentioned earlier, two approaches can be considered to solve the MCA. In this paper, we follow the approach of increasing the efficiency of MAS.

In this paper, we use a task start threshold constraint for each agent, such as agent i, which is denoted by TST_i.

The existence of this constraint means that an agent will start working if it receives a suitable reward. Otherwise it will not start working. The SAg(.) function indicates the operation of this constraint:

$$\mathrm{SAg}\left(r\right)=\begin{cases} True, & r\geq TST\\ False, & r<TST \end{cases} \tag{2}$$

Eq. (2) states that an agent will start working if the received reward by the agent (i.e., r) is higher than its TST; otherwise it will not start. We divide the agents into three categories based on their TST:

The first category is the agents that will start working regardless of the reward received from the critic.

The second category is the agents that start working if their received reward from the critic is more than a certain value. In this paper, the certain value is the TST.

The third category is the agents whose their performance is commensurate with the amount of reward that the critic delivers to them.

In this paper, we operate based on the third case and, based on the TST constraint, introduce the performance power criterion and denote it with P.

The value of P will be between 0 and 100 percent so that the agents that do not receive any reward will have zero performance power, and the agents that receive equal reward or more than their TST will have 100 performance power ,i.e. MPP.

If the reward is between 0 and TST, then the agent's P will be the coefficient of the received reward.

Eq. (3) indicates the performance power of each agent based on the received reward and the agent's TST.

$$P_i(r_i^t)=\begin{cases} \frac{r_i^t}{\mathrm{TST}_i}\times AgRew\ , & r_i^t\leq \mathrm{TST}_i\\ AgRew, & O.W \end{cases} \tag{3}$$

In Eq. (3), P_i is the performance power of the i-th agent, r_i^t is the received reward by the i-th agent at time t, and AgRew is the partial reward/punishment that the environment returns it completely due to the action of the agent due to a received reward equal to or greater than the agent's TST.

In order to agent learning in RL, according to the Eq. (4) that, known as Bellman Equation, we use the Q-Learning method [38] as:

$$Q\left(s,a\right)=Q\left(s,a\right)+\alpha\left[r\left(s,a\right)+\gamma\max\left(Q\left(s^{'},a^{'}\right)\right)-Q\left(s,a\right)\right] \tag{4}$$

In Eq. (4), the s is the agent's current state. The action that the agent act in the state s is denoted with a. s' is the state that the agent transmits to it after action a in the state s. r represents the reward given to the agent due to performing action a in state s. Q indicates the Q-Table of agent, and γ is the discount factor. In addition, the learning rate is denoted with α.

3.2 Bankruptcy Game

The bankruptcy problem, we use in this paper, deals with how the assets of a debtor, which are less than the total demand of the claimants, should be divided among the claimants.

Although this is an old problem, it is still a practical and challenging issue and is used in many areas, from water allocation [39] to the allocation of resources on the Internet of Things and cloud computing [40].

In the bankruptcy problem, which is represented as <E, D>, if we denote the amount of each creditor's claim to d_i, then the vector $D(d_1, d_2, \ldots, d_n)$ is considered as the claimant vector. In addition, if we denote the share of each claimant of E by x_i, we will have:

$$0 \leq x_i \leq d_i \qquad (5)$$

$$\sum_{i=1}^{n} d_i \geq E \qquad (6)$$

The first condition is called the reasonable condition, and the second condition is called the feasible equity condition.

In the bankruptcy game, which is considered as a cooperative game, the claimants are considered as players. So if we have n claimants, then $N = \{1,2,3,\ldots,$ n$\}$. To assign a share to each claimant, the assignment function $c : 2^n \to R$ is used so that $c(\varnothing) = 0$.

In addition, the value of each subset of N, such as S, which is represented by $c(S)$, is equal to the sum of the share of the players in S, obtained using the allocation function, and in fact it is the payoff of players.

In other words, the strategy of each player that follows the allocation function determines the share of each player. The share allocated to each player is result from the player's action.

The allocation function is called "rule", too, in the bankruptcy game. Some of the rules, which we use in this paper will describe in follows [17, 41]:

Proportional Rule (P rule) : The ratio or proportionality method is the simplest and most famous method of the bankruptcy theory. In this method, the allocation coefficient is obtained through dividing the inventory by the amount claimed by the claimants according to Eq. (7). Therefore, the share of every claimant is calculated using Eq. (8) and with an equal coefficient of their needs.

$$\beta = E/D \qquad (7)$$

$$x_i = \beta d_i \qquad (8)$$

Adjusted Proportional Rule (AP rule) : In the AP method, as an allocation to the person i, the needs of all claimants but the person i, according to Eq. (9), are satisfied first. Then, concerning Eq. (10), the remainder is allocated to claimant i, if nothing is left or a negative value is calculated as the remainder, a zero value is allocated to that claimant.

$$x_j = v_i + (d_j - v_i)\left(\sum_{j\in N}(d_j - v_i)\right)^{-1}\left(E - \sum_{j\in N} v_j\right) \quad (9)$$

$$v_i = \text{Max}\left\{0, E - \textstyle\sum_{j\neq i} x_j\right\} \quad (10)$$

In Eq. (9), x_j denotes the share of other claimants, and in Eq. (10), v_i indicates the share of claimant i.

Constrained Equal Award (CEA) : The basic idea behind the CEA is to meet the levels of the needs in an equal way that the amount allocated to each individual does not surpass the level of the need. The following steps are taken for the calculation of the CEA:

In the first step, the lowest claims are considered as an initial allocated value for all creditors. After fulfilling this request, through the elimination of the claimant with the minimum allocation, the process continues with other claimants. Eq. (11) shows how the assignment is worked out in this method:

$$x_i = CEA\left(d_i, \zeta\right) = Min\left(d_i, \zeta\right), \quad s.t. \ \sum_{i\in N} Min(d_i, \zeta) = E \qquad (11)$$

Constrained Equal Loss (CEL) : In the CEL method, it is attempted to distribute the value of the existing deficit evenly among all claimants. Based on Eq. (12), the difference between the number of claims and the source inventory is computed and divided by the number of claimants. The calculated value, which is indeed considered as an equal loss, is deducted from the claims by all claimants and considered as the amount allocated to each claimant.

$$x_i = CEL\left(d_i, E\right) = \text{Max}(0, d_i - \lambda), \qquad s.t. \ \sum_{i\in N} \text{Max}\left(0, d_i - \lambda\right) = E \qquad (12)$$

The Talmud Rule : The Talmud rule is based on the principle that if the amount of assets is less than half of the total debts, none of the creditors will receive more than half of their needs . Conversely, if the amount of assets is more than half of the total debts, all creditors will receive more than half of their claims. Eq. (13) is a mathematical expression of the Talmud rule:

$$\text{x}_i = \begin{cases} \text{CEA}\left(\frac{d_i}{2}, E\right) & \text{if } \frac{1}{2}D \geq E \\ \frac{d_i}{2} + CEL\left(\frac{d_i}{2}, E - \frac{D}{2}\right) & \text{if } \frac{1}{2}D \leq E \end{cases} \qquad (13)$$

3.3 Evolutionary Game

The EG, which introduced by John Maynard Smith, is based on Darwin's theory of evolution [42]. The basis of this game is that every living thing, as a player, follows a predefined strategy. According to the unwritten law of nature, any creature that has a better strategy so as a result has better performance, has a better chance of surviving in nature. As a result, successful organisms reproduce and unsuccessful organisms die without reproduction. Creatures that are created by reproduction inherently follow their parents' strategies, but in the meantime, the presence of mutations causes different behaviors among them. The existence of mutation is the most essential principle in the process of evolution.

EG is a subset of game theory that follows the process of natural evolution. In EG, players first start the game randomly with a pure or mixed strategy. After that, the players try to change their strategy by imitating the players who act well. Some concepts have been introduced to analyze the EG, which describe follows:

Evolutionary Stable Strategy (ESS) Strategy σ^* is an ESS if, for any strategy such as σ which is not equal to σ^* and is chosen by a part of the population such as ϵ, there is a population such as $\bar{\epsilon}>\epsilon$ in a way that:

$$\sigma \neq \sigma^*$$

$$\bar{\epsilon}>\epsilon$$
$$u(\sigma^*, \epsilon\sigma + (1-\epsilon)\,\sigma^*) > u(\sigma, \epsilon\sigma + (1-\epsilon)\,\epsilon^*) \qquad (14)$$

Interpretation of Eq. (14) is that no strategy can overcome an ESS in an evolutionary game. In other words, the part of the population that chooses the strategy σ, cannot make more profit than the other part of the population that chooses the strategy σ^*. It has been proven that every EES is a Nash equilibrium [43]. It is important to note that ESS does not necessarily exist, and even if it exists, there is no guaranteed way to calculate it. Therefore, we can use dynamics to solve an evolutionary game.

Replicator Dynamic Replicator dynamics is a model of evolution that explains how population share associated with different strategies grows over time [44]. Replicator dynamics assumes population size as unlimited, the time as infinite and continuous, and complete mixing. Complete mixing means that completely paired strategies are chosen from the population at random. There are some evolutionary dynamics such as replicator dynamics (RD), Brown-von Neumann-Nash dynamics (BNN), logit dynamics, and Smith dynamics (Smith) [45]. In addition, one can use of a combination of them. We use of a combination of them in this paper.

3.4 Multi-score Puzzle

Multi-score Puzzle, which is called MsP in this paper, is our operating environment. MsP includes some cells and pieces, and any agent is responsible for filling some of these cells with appropriate pieces. The most important issue about this puzzle is that filling any cell, with right or wrong pieces, has a different point compared to other cells; for this very reason, we called it MsP. It should be noted that the punishment for filling a cell with a wrong piece was zero in our work. There is a pool of pieces that any agent can select of them and, using its Q-Table, put the chosen pieces in a suitable cell. Any agent is responsible for solving a section of the puzzle that contains one or many cells.

4 Methodology

In this section, we propose a method to solve MCA based on both bankruptcy and EGT. First, based on MPP, the MCA will be turned into a bankruptcy game. Then, considering that there are numerous answers instances to solve the MCA as a bankruptcy problem, we turn it into an EG and use it to solve the MCA. Therefore, the proposed method consists of two parts: the bankruptcy game to extract the EG players and the EG to solve the MCA.

4.1 Phase 1: Bankruptcy game to extract EG players

We use the bankruptcy game, in phase 1, to find any reward distribution instance that acts as a player in the EG to solve the MCA. To turn the MCA into a bankruptcy problem, we use the MPP and TST, which were described before.

If the global reward to be distributed among the agents is equal to or greater than the total TSTs, then according to the Eq. (3), all agents reach their MPP. The problem becomes challenging when the global reward, to be distributed among agents, is less than the total TST of agents. In this case, the problem becomes a bankruptcy problem according to the definition of bankruptcy, and the performance power of each agent follows Eq. (3). In this paper, we examine the second case.

To distribute global rewards among agents, concerning Eq. (5) and Eq. (6), each agent is considered as a player. In addition, the global reward that must be distributed among the agents is regarded as a finite asset (E), the agents' demands are considered as distribution vector D, the agents' received reward from the critic is considered as the player's share, and each agents' request is considered as the player's request. Table 1 indicates how the MCA is mapped to a bankruptcy game.

In the bankruptcy game, each player (agent) uses a mixed strategy σ_i to find its share of the global reward.

If we assume that the p method (rule) exists as $Rule = \{\text{rule}_1, \text{rule}_2, \ldots, \text{rule}_p\}$ to solve the bankruptcy problem, then

Table 1 Mapping the MCA into a bankruptcy game

MCA Problem	Bankruptcy Problem
Ag_i	$\mathrm{claimant}_i$
Global Reward	E
r_i	x_i
TST_i	d_i

according to Eq. (15), the agents' share of the global reward at time t is determined based on the bankruptcy game:

$$r_i^t = \sum_{a=1}^{p} \sigma_i \times \mathrm{rule}_a \quad , \; where \quad \sum_{i=1}^{K} r_i^t \leq \mathrm{Reward}^t \tag{15}$$

By determining the agents' reward at time t, i.e., r_i^t, the allocation vector at time t, i.e., $R^t(r_1^t, r_2^t, \ldots, r_K^t)$ is also determined. Each allocation vector at time t is an instance of a global reward distribution among the agents, which we consider it as an EG player and is represented by R_j^t.

4.2 Phase 2: EG to find the best reward distribution among agents

4.2.1 The Main Idea

Given the finite number of global reward distribution instances as players and the number of strategies, the proposed game in this section is a finite game. According to Nash equilibrium, each finite strategic game has at least one Nash equilibrium of mixed strategy [46]. Given that it is difficult to prove this and find Nash equilibria, there are alternative ways to find it, one of which is the EG [47].

In the second phase of our proposed method, to solve the MCA we use the EG. If we consider each instance of the global reward distribution among the agents, namely the vector $R_j^t\,(r_1^t, r_2^t, \ldots, r_K^t)$ as an answer for MCA, then the goal of the EG will be to find the best answer to solve MCA. Therefore, in the EG in question, we consider every R_j^t as a player.

If we assume that for each global reward, there are "ExistDisNo" number instances of distribution between agents, then the set of players in the EG will be the exact as the instances set of distributions:

$$Players = \{R_1^t, R_2^t, \ldots, R_{\mathrm{ExistDisNo}}^t\}$$

Each player's strategy will be distributing the global reward among the agents, which based on Q-learning, causes the agents to take action in the environment, consider the current state, and be placed in the next state.

```
Algorithm MCASolving

Inputs:    Vector of agents Reward  $U\_BestMCA\ (R_0^U, R_1^U, R_2^U, \ldots, R_{RewNo}^U)$
Outputs :  Valued Vector $V\_BestMCA\ (R_0^V, R_1^V, R_2^V, \ldots, R_{RewNo}^V)$

1.    i=0
2.    for Reward=0 to MaxReward do
3.        $R_i^V$ =EvoMCA(Reward)
4.        Update (BestMCA, $R_i^V$)
5.        i++
6.    end for
```

Each action of an agent causes the environment to generate a partial reward/punishment and return the resultant of these partial rewards/punishments to the MAS in the form of a global reward, which is the payoff for each player's game. The normal/strategic form of this game is indicated below:

$$N = \{1, 2, \ldots, ExistDisNo\}$$
$$A_i = \mathrm{Act}_i^t$$
$$U_i = En(Mas(R_i))$$

The main algorithm we offer to solve the MCA is the "MCASolving" algorithm, which is described below.

In the "MCASolving" algorithm, we are looking for the best distribution for each reward that the environment returns to the MAS. This distribution must be such that it results in the best performance of the MAS, to obtain the highest reward. The algorithm's input is a vector of unvalued distributions, i.e.,R_i^U, which we represent with "U_BestMCA" in the algorithm. After running the algorithm, each distribution will contain the best possible distribution for the corresponding reward based on the EG. The V_BestMCA vector contains these distributions. For example, R_i^V represents the best reward distribution among agents, given the ith reward. To simplify the problem, we assume that the number of rewards that the environment returns to the critic is finite and bounded to a minimum of 0 and a maximum of "MaxReward". In lines 2 to 5 for each of these rewards the EvoMCA function is called to find the best distribution based on the EG, and the "V_BestMCA" vector is updated accordingly.

4.2.2 EG to Solve MCA

The *EvoMCA* function, which is the core contribution of this paper, finds the best distribution for each reward based on the EG.

In this function, the reward that must be distributed among the agents is considered as input, i.e., *Reward*. The output of this function is the $R^W\left(r_1^W, r_2^W, r_3^W, \ldots, r_K^W\right)$ vector , which contains the best distribution instance of *Reward* among the agents base on the EG.

```
Algorithm   EvoMCA

Inputs:    Reward                          // The reward should be distributed between agents
Outputs :  $R^W(r_1^W, r_2^W, r_3^W, \ldots, r_K^W)$   //The Best Distribution as winner player

1.  g=0
2.  Players= CreatePlayers (k,p,Reward,ExistST)
3.  Randomly generate the initialize V population as $\mathbb{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_V\}$
4.  while not termination do
5.      for each population $\mathcal{P}_i$ randomly selected from $\mathbb{P}$ do
6.          $\mathcal{P}_i' = \emptyset$
7.          for j=1 to $|\mathcal{P}_i|/2$ do
8.              player1=randomlySelect($\mathcal{P}_i$)
9.              player2=randomlySelect($\mathcal{P}_i$)
10.             winner=performGame(player1,player2)
11.             replica=replicate(winner)
12.             if random( )<= $P_m$ then
13.                 replica=mutate(winner)
14.             end if
15.             $\mathcal{P}_i = \mathcal{P}_i - \{player1, player2\}$
16.             $\mathcal{P}_i' = \mathcal{P}_i' \cup \{winner, replica\}$
17.         end for
18.         $\mathcal{P}_i = \mathcal{P}_i'$
19.     end for
20.     g=g+1
21. end while
22. return $R^W(r_1^W, r_2^W, r_3^W, \ldots, r_K^W)$
```

In this function, we must first have all the players as a population. These players, which are identified with Players in line 2, will be created using the "CreatePlayers" function. In other words, all instances of *Reward* distribution among agents are generated by the critic as players using the "CreatePlayers" function.

After this step, all the players (different distribution instances of *Reward*) are separated into the V populations that the set of these populations is indicated by $\mathbb{P}$.

EG takes place between lines 4 to 21 to find the best player (the best distribution instance of Reward). In fact, the *EvoMCA* algorithm implements the process of iterative evolution to find the dominant strategy, as the optimal global solution, between generations.

Theoretically, the process of evolution considers infinite population and infinite time, which is not possible in practice. For this reason, the implementation of the algorithm must be limited and stopped at one point. Two different static and dynamic approaches can be considered for this purpose.

In the static approach, the algorithm ends when the number of G generations reaches a specific number. This number is determined experimentally. In the dynamic approach, the algorithm terminates when the dominant strategy does not improve more than x% of the performance among the y generation,

```
Algorithm  CreatePlayers

Inputs:  Reawrd      // The value of Reward that should be distributed between Agents
         ExistIns    //The number of distribution instances
         k           // The number of agents in MAS
         p           // The number of pure strategies
Outputs: Players ($R_1, R_2, \ldots R_{ExistST}$)
1.  Players=∅
2.  for i=1 to ExistIns do
3.  |     $R_i$ =Distribution (k,p,Reward)
4.  |     Players=Players∪ $R_i$
5.  end for
6.  return(Players)
```

or the dominant strategy is more than z% of the population. Here, too, the values of x, y, and z are determined experimentally.

Line 4 examines the condition of termination of the algorithm based on the mentioned approaches. In this paper, we use the static method.

In line 5, a population such as $\mathcal{P}_i$, is selected from the separated populations (i.e., $\mathbb{P}$) for evolutionary analysis, randomly.

In the EG and comparison of two populations with different strategies, the population with the dominant strategy replaces the population with the defeated strategy.

To hold the dominant population, $\mathcal{P}_i^{'}$ is used. The following repetitive operations are performed between lines 7 to 17:

1. Two players are randomly selected.
2. The two players are compared using $performGame$, and the dominant player is selected based on a better strategy.
3. population replacement is performed
4. The mutation is performed if necessary.
5. Both players are eliminated from the $\mathcal{P}_i$ population.
6. $\mathcal{P}_i^{'}$ as the dominant population is updated using the dominant and mutant player (if any).

In line 18, the dominant population is renamed $\mathcal{P}_i$ for the next round. After reaching the endpoint and leaving the iteration loop, there will be a dominant population with a better strategy based on a static or dynamic approach. In fact, at the end of the repetition loop, the population of players (distribution) as $R^W\left(r_1^W, r_2^W, r_3^W, \ldots, r_K^W\right)$ is the dominant population, which will be the output of the function.

The *EvoMCA* function uses the *CreatePlayers* function to generate players and the *performGame* function to find the dominant player, which are described below. First, we describe the *CreatePlayers* function.

Algorithm Distribution

Inputs : k // The number of Agents in MAS
p// The number of Pure strategies
Reward // The value of Reward that should be distributed between Agents

Outputs:
Valued Vector R($r_1, r_2, r_3, \ldots, r_k$) // a distribution of Reward Between Agents

```
1.  R=$Zero(r_1, r_2, \ldots, r_k)$
2.  New(D)=0;
3.  for i=1 to k do
4.      for j=1 to p do
5.          $r_i = r_i + \sigma(i,j) \times rule_j$
6.      end for
7.      if (New(D) + $r_i$)<=Reward then
8.          begin
9.              New(D)=New(D)+ $r_i$
10.             UpDate(R, $r_i$)   //Add the $r_i$ into R
11.         end
12.     else if
13.         begin
14.             $r_i = 0$
15.             UpDate(R, $r_i$)
16.         end
17.     end if
18. end for
19. return (R)
```

In this paper, our players are different instances of the global reward distribution among agents. This function is the producer of these players. Initially, an empty set of Players in line 1 is considered. This set is updated in a loop between lines 2 and 5 with the created distribution instances, i.e., players. This function uses another function in line 3, which is called *Distribution* to create a distribution instance (player) and build the players set. In Distribution function, a distribution instance is created for each Reward. This instance of distribution is considered as a player. A bankruptcy game is used to create this player instance.

Each distribution instance (player) uses a mixed strategy to assign rewards according to pure strategies. If we consider the number of pure strategies equal to p, then the strategy that each player chooses is obtained from lines 3 to 6.

Given that the MCA has become a bankruptcy problem, line 7 examines the feasibility of reward assignment to each agent, and if the bankruptcy condition is met, i.e., $\sum_{i=1}^{K} r_i \leq E$, which is mapped to (New(D) + r_i)<=Reward in here, the reward is assigned to the agent otherwise the reward is not assigned to the agent (lines 7 to 17).

Algorithm performGame

Inputs: Player1 , Player2 // Two different Distribution of Reward

Outputs: Winner Of The Game

```
1. if En(Mas(Player1))>En(Mas(Player2)) then
2.     return (Player1)
3. end if
4. if En(Mas(Player1)) < En(Mas(Player2)) then
5.     return(Player2)
6. end if
7. if En(Mas(Player1)) = En(Mas(Player2)) then
8.     return Randomly(Player1,Player2)
9. end if
```

Given that the R vector contains the amount of rewards should be assigned to agents. Therefore after determining the amount of allocated reward to each agent, this vector is updated (lines 10 and 15).

At the end of the algorithm, the R vector is returned, which now contains all the rewards assigned to the agents according to the bankruptcy game and global reward.

Another function, which is used by the *EvoMCA* function is the *performGame* function. As mentioned earlier, according to Eq. (1), each distribution instance causes the different performance of agents in the environment. As a result of this performance in the environment, the environment returns a global reward that is the resultant of these partial rewards/punishments of agents. According to Eq. (1), if $\text{Act}^t\ (a_1^t, a_2^t, \ldots, a_K^t)$ is the vector of the performance of agents in the environment at time t due to the reward vector R, then the function En (.) represents the received reward from the environment by a MAS at time t + 1 and will have:

$$\text{Reward}^{t+1} = En(Mas(R^t)) \qquad (16)$$

In the performGame function, both players are compared based on the amount of rewards they receive from the environment according to the Eq. (16), and the dominant player will be the player who generates the most rewards.

5 Simulation

To simulate the proposed method, MAS that contained ten agents was considered. Five strategies P, AP, CEA, CEL, and Talmud were used as pure strategies. The operating environment was a MsP, in a way that the goal of the agents in the MAS was to put the parts in the right place. Each agent was responsible for solving a part of the puzzle based on their Q table. To evaluate the proposed method, the following criteria, which were introduced in [10, 30], were used and three methods of dynamic, ranking, and history-based,

based on these criteria, were compared with the proposed method. These criteria were: group learning rate, confidence, expertness, certainty, efficiency, and correctness, which are described below.

5.1 Group learning rate

If we denote the LR_i as the learning rate of each agent, then the group learning rate will be the average learning rate of the agents, which is indicated by LR and is calculated based on Eq. (17).

$$\mathrm{LR}^t = \left(\frac{1}{n}\right) \sum_{i=1}^{N} \mathrm{LR}_i^t \tag{17}$$

Eq. (18) is used to calculate the learning rate of each agent.

$$\mathrm{LR}_i^t = \frac{|Learnt(S_i^t)|}{|S|} \tag{18}$$

The learning rate of each agent is defined according to Eq. (19) as the rate of learner states.

$$\text{Learnt}\left(S_i^t\right) = \left\{\forall a_i^t : feasible\left(a_i^t, s_i^t\right) \rightarrow f_{a_i}^{\text{suggested}} = f_{a_i}^{\text{real}}\right\} \tag{19}$$

In Eq. (18) $|Learnt(S_i^t)|$ is the number of states that the agent learns. Learning is considered as the highest value of Q in all states. In addition, the correct action is selected using the greedy method. | S | Indicates the number of states available for each agent. Figure 1 shows the results of comparing the proposed method with existing methods based on the group learning rate criterion. There are different types of agents in the MAS. These agents include agents with different functions. Agents can be categorized based on their performance in the environment and whether they cause the reward or punishment. As mentioned earlier, there are two approaches to solving the MCA. These approaches are fair approach and efficient enhancement approach. We are following the second approach. In this approach, the best method to reward assignment is the method that causes the highest reward in the next round. Therefore, in this approach, agents with better performance may be given more attention and receive more rewards; in return, agents with weak performance may receive fewer rewards.

In the proposed method, in each generation of the population, the dominant population is selected as the reward assignment method. Therefore, in the first generations, when the best instance of reward assignment is not found, the reward assignment is based on the dominant population, which is not the best instance of reward assignment. This increases the likelihood of reward assignment to poorly performing agents, resulting in the group learning rate will be decreased. Therefore, considering that the best instance of reward assignment has not been found in the first episodes, it is observed that the learning rate

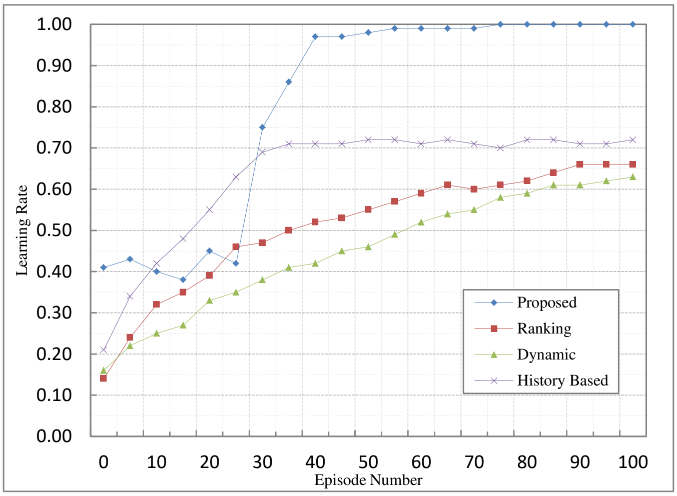

Fig. 1 Comparison of four MCA methods based on the group learning ratio criterion

of the proposed method is low and fluctuating. The reason for the fluctuation of the learning rate is that in each episode, the reward assignment instance is used, which is not necessarily better or worse than the previous instance, but only the reward assignment method is used, which constitutes the majority of the methods. This process continues until the best reward allocation instance is found. After the best reward assignment instance, based on the proposed method (EG), is found, the performance of the MAS improves, and as a result, the learning rate increases. In other reward assignment methods, because the reward assignment method is fixed, so the best reward assignment method may never be found. As a result, their learning rate converges to an intermediate value and remains constant at this point. This is absolutely clear in the history-based method, which has the best performance among other methods, based on this criterion.

5.2 Confidence

The next criterion, which the proposed method is compared with other methods, is confidence. This criterion is extracted when Q-Table is being completed. This criterion is the difference between the second largest value of Q-Table from its largest value. The greater the difference means, the more inclined the agent is to choose the appropriate action. If [q(1),q(2),q(3),. . . ,q(|A|-1),q(|A|)] are the values in Q-Table, which are arranged in ascending order, then the confidence of each agent, based on Eq. (20), is obtained as follow:

$$\mathrm{Cnf}\left(Ag_i^t\right) = q\left(|A|\right) - q\left(|A|-1\right) \tag{20}$$

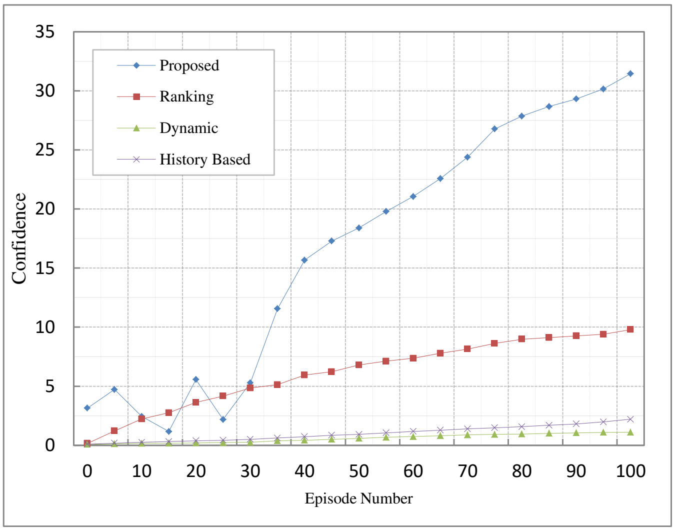

Fig. 2 Comparison of four MCA methods based on the confidence criterion

Figure 2 indicates the comparison of the proposed methods with other methods based on this criterion. This criterion is specified based on the Q-Table of agents. Due to the fact that in the initial episodes, the values of Q-Table are often zero, so the value of this criterion is low in all methods. The higher the reward received by the agent, the higher the correct values in Q-Table, and consequently, the value of this criterion also increases. Due to the fact that different reward assignment methods, according to Eq. (1), will receive different global rewards, so the value of this criterion will be increased by reward assignment methods that will result in higher global rewards by the MAS. Given this, the best method of reward assignment is the method that causes the highest global reward. In the proposed method, we seek to find the best instance of reward assignment. In the proposed method, the best instance of reward assignment as EG players will be found as the dominant population after a few episodes. Therefore, until the best instance of reward assignment is found, other methods are used to assign rewards between agents, which are the dominant method in the population. These methods result in different global rewards, and as a result, the amount of rewards they distribute among agents are varies. Each method may receive a higher or lower global reward than the previous episode. Figure 2 indicates that until finding the best instance of reward assignment, there are no fixed state values. After finding the best reward assignment instance, the global reward increases. As a result the difference between the highest and the second highest value of Q-Table and consequently, the value of this criterion also increases. In other reward assignment methods, because they do not necessarily use the best method of assignment, it is possible to assign an inappropriate rewards to agents; as a result, it causes to obtain a less global reward. Among other methods of

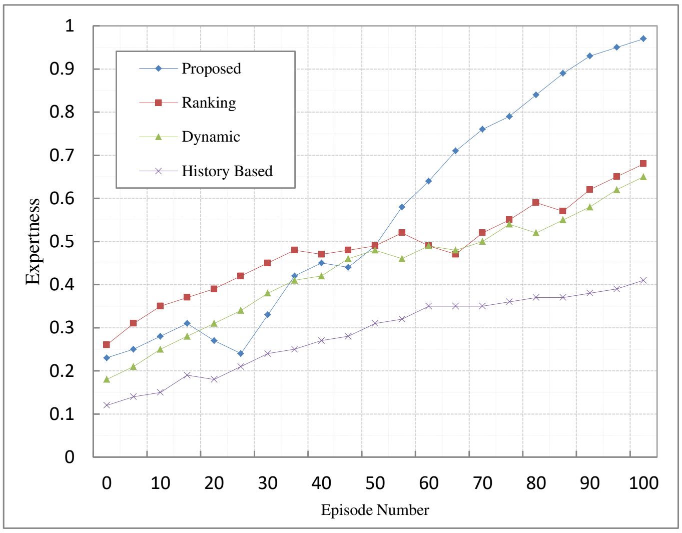

Fig. 3 Comparison of four MCA methods based on the expertness criterion.

reward assignment based on this criterion, the best performance is the ranking method, which operates based on the knowledge of agents. Although this method tries to give priority to agents with higher knowledge to obtain higher global rewards, but this method is not necessarily the best method of reward assignment and therefore has worse performance than the proposed method.

5.3 Expertness

The following criterion for evaluating the proposed methods is expertness. This criterion indicates the difference between the number of times the agent receives a reward (selecting the right action) and the number of times it receives a punishment (choosing the wrong agent). Eq. (21) states this criterion:

$$Expertness = N_r - N_p \tag{21}$$

N_r : The number of times the agent receives a reward N_p: The number of times the agent receives a punishment

Figure 3 indicates the comparison of the proposed method with existing methods based on this criterion. In the proposed method, the most crucial goal is to find the best instance of reward assignment among the agents. The best reward assignment instance increases the learning rate of agents and, as a result, their better performance in the environment and obtaining the highest global reward by the MAS. In addition, the amount of fines received due to the improper performance of agents will be reduced. Considering that finding the best instance of reward assignment requires the passage of time and the dominance of the population, which has the best performance among the populations. Hence, until finding this instance of reward assignment, the proposed method uses the instances that are dominant and not the best method. This makes the proposed method work normally until it reaches the point of finding the best reward allocation method. Finding the best instance to reward assignment will improve agent learning and result in fewer mistakes, resulting in more rewards for agents and fewer punishments.

If it is possible, the ranking method, history-based method, and dynamic method, assigns reward to all agents and do not differentiate between them. Therefore, it is possible that low-performance agents may be rewarded. Poor performance of these agents leads to fines in the environment, which results in an increase in global fines and, consequently, the distribution of fines among the agents. Among these methods, the ranking method, because it operates based on knowledge of agents and prioritizes agents with higher knowledge, has a better performance than the history based method and dynamic methods. However, the ranking method does not seek the best method of assignment.

5.4 Certainty

The fourth criterion for evaluating the proposed methods is certainty. This criterion is calculated based on Eq. (22) and compares the value of Q in action "a" and the state "s" with other values of the state "s" based on Eq. (22).

$$\operatorname{Cert}\left(s_i^t, a_i^t\right) = \frac{\exp\left(\frac{Q(s_i^t, a_i^t)}{T}\right)}{\sum_{a_i^t \in A} \exp\left(\frac{Q(s_i^t, a_i^t)}{T}\right)} \quad (22)$$

Eq. (23) specifies the value of T for each episode.

$$T = Max\left\{\frac{T_0}{1 + \log(\text{episode})}, T_{\min}\right\} \quad (23)$$

Based on [10], we set T_0 to 10 and $T_{\min}$ to one for our experiments. Figure 4 shows a comparison of the proposed methods with other methods based on this criterion. This criterion is calculated based on the values in Q-Table. Due to the fact that in the first episodes, most of the values in Q-Table are zero, so, as seen in figure 4, in the early episodes, the values of this criterion are low. In addition, the correct or incorrect action of the agents will cause changes in this criterion. In the proposed method, the goal is to find the best method to assign rewards. The best method to assign a reward is the method that results in the best performance of the agents and thus the highest global reward. This makes the agents act the best action in the proposed method, and as a result, the value of this criteria increases. This increase occurs from the middle episodes and at the same time as finding the best instance of reward assignment. In the initial episodes, the performance of MAS is based on this criterion and is low. The reason for this is that in the early episodes, based on instances that make up the dominant population and are not the best instance of reward assignment, reward assignment occurs. Among other MCA solution methods, the ranking method has the best performance. The reason for this is that the ranking method prioritizes the reward assignment to agents with higher

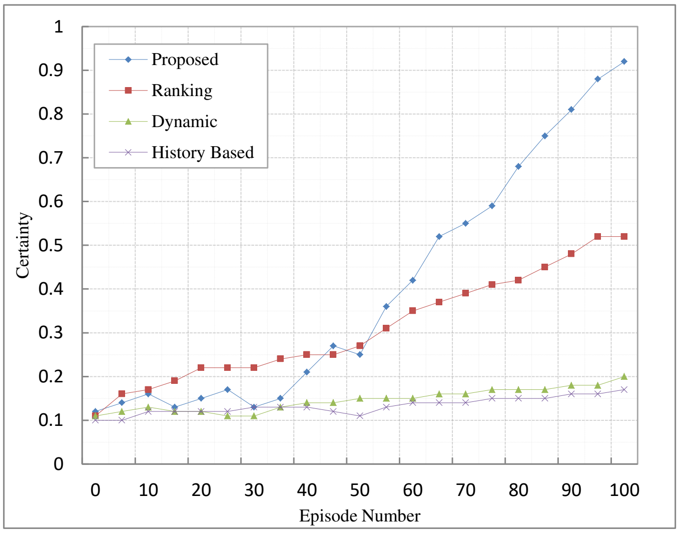

Fig. 4 Comparison of four MCA methods based on the certainty criterion.

knowledge. Due to higher knowledge, these agents have a higher learning rate than the history-based method and dynamic method.

5.5 Efficiency

Efficiency, which is defined by Eq. (24), is the fifth criterion for evaluating methods. This criterion indicates how many times a non-zero reward is assigned to the agent. The process of assigning the reward to agents by the critic must be done in a conservative way to guide the agent to the goal. In other words, if the reward assignment to any agent causes that agent will be misled, then the agent will lose the goal or spend more time to reach the goal. Therefore, any non-zero attribution to the agent by the critic means that the critic accepts this risk and judges the agent's choice of action.

$$\text{Eff} = \sum_{i=1}^{F} I\left(r_i^t \neq 0\right) \quad (24)$$

$$I\left(x\right) = \begin{cases} 1, & x : \text{True} \\ 0, & x : \text{False} \end{cases}$$

$$F : \text{the number of feedbacks}$$

The results of comparing the methods based on this criterion are shown in Figure 5. The ranking method has a better performance in comparison to all methods. Due to the fact that in the ranking method and two other methods (history based method and dynamic method), it is an attempt to assign reward to all agents (even if it will be low) and this reward has a non-zero value, Therefore, unlike the proposed method, which may not assign reward to a large

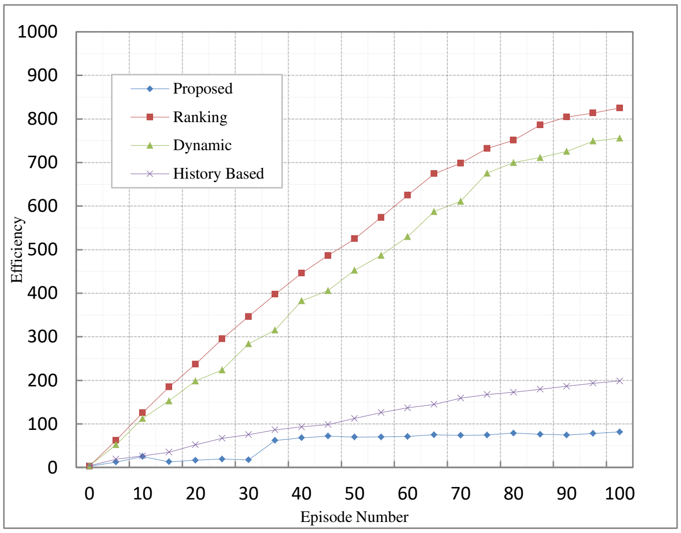

Fig. 5 Comparison of four MCA methods based on the EfficiencyP3 criterion.

number of agents, these methods have better performance than the proposed method. In the proposed method, the most essential issue is to find the best instance of reward assignment, which leads to the highest global reward by MAS. In this method, the found instance may assign a zero value to one or more agents. Therefore, considering that in the proposed method, it is possible to assign a zero value to many agents, so in this view, it performs worse than other reward assignment methods that try to reward all agents. In other words, because in ranking methods, dynamic method, and history-based method, the goal of reward assignment is a fair distribution of reward, so it is attempt to assign rewards to all agents based on their participation. In contrast, in the proposed method, the goal is to increase the efficiency of MAS, so in the proposed method, the best instance of reward assignment maybe does not assign a reward to one or more agents.

5.6 Correctness

The sixth criterion to comparison between the proposed methods and the existing methods is correctness. Correctness can be expressed based on various criteria. The most flexible definition of Correctness is based on the threshold value. In this case, if the difference between the assigned reward and the actual reward is less than the threshold, this assignment is considered as the correct assignment; otherwise, this assignment is incorrect. Eq. (25) states this.

$$\mathrm{Corr} = \sum_{i=1}^{F} I(\left(r_i^t - r_{i\ \mathrm{cor}}^t\right) < T) \tag{25}$$

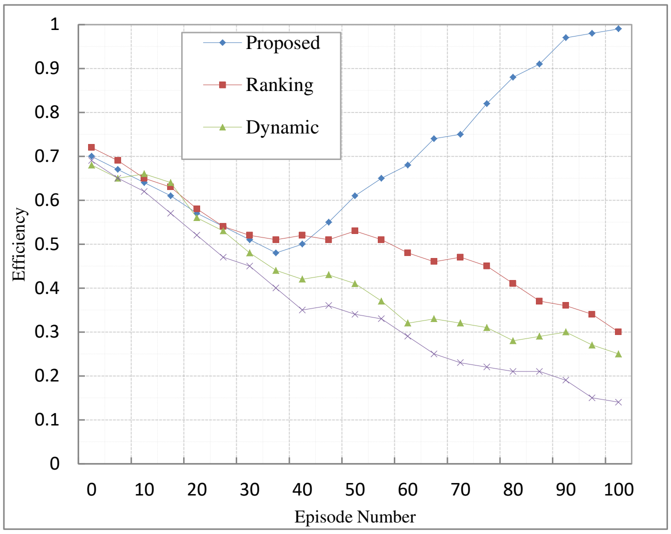

Fig. 6 Comparison of four MCA methods based on the Correctness criterion.

$$I(x) = \begin{cases} 1, & x : \text{True} \\ 0, & x : \text{False} \end{cases}$$

$$r_i^t : \text{ The done assignment}$$

$$r_{i\ \text{cor}}^t : \text{ The actual assignment}$$

$$F : \text{ The Number of Feedbacks}$$

$$T : Threshold$$

Figure 6 indicates the comparison of the proposed method with other methods based on this criterion. In MAS, agents with a higher learning rate have better action, resulting in higher scores. With this in mind, the proposed method seeks to assign rewards to these types of agents in order to obtain higher rewards. In the lower episodes, because the best instance of reward assignment has not yet been found, incorrect rewards may be assigned to agents. This means that low-performing agents, which cause low-score, may receive high rewards that make MAS perform poorly based on this criterion. This is quite obvious in the first episodes. Because the best instance of reward assignment seeks to earn the best reward from the environment, so the best instance of reward assignment makes the correct assignment to the agents. As a result, after finding this instance, MAS performs better based on this criterion. In other methods of reward assignment, because without considering the performance of agents, rewards are assigned to all agents, so the probability of incorrect reward assignment to agents will be high. As a result, based on this criterion, they have poorer performance than the proposed method. Among these methods, the ranking method has a better performance. The reason for this is that in the ranking method, the reward assignment to the agents is done based on their knowledge, and as a result, the higher knowledge agents, causes higher scores, receive more rewards, and as a result, the correctness in this method is more heightened. However, in this method, less knowledgeable agents who cause fines are also rewarded, which makes them perform worse than the proposed method based on this criterion.

6 Conclusions

MASs are widely used for modeling and implementing complex systems. One of the most disputable problems about MAS is their learning, which often takes the form of RL. MARL is a multi-part process from which one of the parts is the MCA. In this paper, we present a two-step method for solving the MCA by converting the TST to the MPP, meaning that the performance of each agent is based on the reward it receives. The existence of this constraint caused us to turn the MCA into a bankruptcy game in the first stage.

In the bankruptcy game, there is a finite asset that must be distributed among the claimants. The main point in the bankruptcy game is that the amount of this asset is less than the total claim of the claimants. In this paper, by mapping the agents to the claimants, the MPP of the agent to the claimant's claim, and the global reward that must be distributed among the agents to the finite asset, we turned the MCA into a bankruptcy game.

Another challenge that we faced in this research was that to solve MCA as a bankruptcy problem, a large number of answer instances can be found, so finding the best answer instance was not an easy task. To find the best instance answer among the answers, we used the EG. In the EG, which is derived from Darwin's theory of evolution, the population (s) that have well performance will survive; in contrast, the population(s) that have weak performance will be wiped.

To turn the MCA into an EG, which has now become a bankruptcy problem, answer instances of bankruptcy problem have been considered as EG players, in a way that each of them plays based on a mixed strategy. Each instance of the answers for the MCA will cause a different performance of agents and, as a result receive different rewards from the environment, which we considered in this paper as the players' payoff.

As a result of this game, the best instance of the answers remains as the dominant population, and the rest of the instance answers disappear as the defeated population. As a result, the critic will be able to choose the best mode of distribution of rewards between agents for each global reward that it receives from the environment. Therefore, the proposed method will always find the best answer among the available answers.

To evaluate the proposed method, six criteria including group learning rate, confidence, expertness, efficiency, certainty, and correctness were used. The proposed method was compared with three methods include: ranking methods, history based method and dynamic method. Compared to the other methods and according to the considered criteria, the proposed method performed better. This is because, according to the global reward received through distribution, the proposed method is looking for the best instance of reward assignment. The only criterion that the proposed method had a poorer performance than the other methods was the efficiency criterion. The reason for this is that this criterion depends on the number of times the agent(s) receive a non-zero reward. In the proposed method, as it is intended to increase the efficiency of the MAS, the agents that have poor performances may not be rewarded at all. So in the proposed method, the number of agents that do not receive rewards is much less than the methods that seek fair distribution, in which the goal is to reward all agents, even those who perform poorly.

References

[1] Ferber J, Weiss G. Multi-agent systems: an introduction to distributed artificial intelligence. vol. 1. Addison-Wesley Reading; 1999.

[2] Krishnan S, Boroujerdian B, Fu W, Faust A, Reddi VJ. Air Learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation. Machine Learning. 2021;110(9):2501–2540.

[3] Asadi M, Fathy M, Mahini H, Rahmani AM. An Evolutionary Game Approach to Safety-Aware Speed Recommendation in Fog/Cloud-Based Intelligent Transportation Systems. IEEE Transactions on Intelligent Transportation Systems. 2021;.

[4] Gazafroudi AS, Prieto-Castrillo F, Pinto T, Corchado JM. Organization-based multi-agent system of local electricity market: bottom-up approach. In: International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer; 2017. p. 281–283.

[5] Powers R, Shoham Y, Vu T. A general criterion and an algorithmic framework for learning in multi-agent systems. Machine Learning. 2007;67(1):45–76.

[6] Pesce E, Montana G. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. Machine Learning. 2020;109(9):1727–1747.

[7] Kilinc O, Montana G. Reinforcement learning for robotic manipulation using simulated locomotion demonstrations. Machine Learning. 2021;p. 1–22.

[8] Adams S, Cody T, Beling PA. A survey of inverse reinforcement learning. Artificial Intelligence Review. 2022;p. 1–40.

[9] Cowen-Rivers AI, Palenicek D, Moens V, Abdullah MA, Sootla A, Wang J, et al. Samba: Safe model-based & active reinforcement learning. Machine Learning. 2022;p. 1–31.

[10] Rahaie Z, Beigy H. Critic learning in multi agent credit assignment problem. Journal of Intelligent & Fuzzy Systems. 2016;30(6):3465–3480.

[11] Mannion P, Devlin S, Duggan J, Howley E. Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning. The Knowledge Engineering Review. 2018;33.

[12] Mannion P, Devlin S, Duggan J, Howley E. Multi-agent credit assignment in stochastic resource management games. The Knowledge Engineering Review. 2017;32.

[13] Nguyen DT, Kumar A, Lau HC. Credit assignment for collective multiagent RL with global rewards. Advances in neural information processing systems. 2018;31.

[14] Shao J, Zhang H, Jiang Y, He S, Ji X. Credit assignment with meta-policy gradient for multi-agent reinforcement learning. arXiv preprint arXiv:210212957. 2021;.

[15] Likmeta A, Metelli AM, Ramponi G, Tirinzoni A, Giuliani M, Restelli M. Dealing with multiple experts and non-stationarity in inverse reinforcement learning: an application to real-life problems. Machine Learning. 2021;110(9):2541–2576.

[16] O'Neill B. A problem of rights arbitration from the Talmud. Mathematical social sciences. 1982;2(4):345–371.

[17] Curiel IJ, Maschler M, Tijs SH. Bankruptcy games. Zeitschrift für operations research. 1987;31(5):A143–A159.

[18] Antonopoulos A. Bankruptcy problem in network sharing: fundamentals, applications and challenges. IEEE Wireless Communications. 2020;27(4):81–87.

[19] Weibull JW. Evolutionary game theory. MIT press; 1997.

[20] Sandholm WH. Evolutionary game theory. Complex Social and Behavioral Systems: Game Theory and Agent-Based Models. 2020;p. 573–608.

[21] Pearce JM. Animal learning and cognition: an introduction. Psychology press; 2013.

[22] Neftci EO, Averbeck BB. Reinforcement learning in artificial and biological systems. Nature Machine Intelligence. 2019;1(3):133–143.

[23] Xu Y, Liu P, Zhang X, Zha C, Tian Z. Formation Control and Obstacle Avoidance for Multi-agent Systems in Unknown Environment. In: 2019 IEEE International Conference on Unmanned Systems (ICUS). IEEE;

2019. p. 925–930.

[24] Dubenko Y, Dyshkant E, Gura D. Multi-Agent Reinforcement Learning for Robot Collaboration. In: Robotics, Machinery and Engineering Technology for Precision Agriculture. Springer; 2022. p. 607–623.

[25] Li T, Zhu K, Luong NC, Niyato D, Wu Q, Zhang Y, et al. Applications of Multi-Agent Reinforcement Learning in Future Internet: A Comprehensive Survey. IEEE Communications Surveys & Tutorials. 2022;.

[26] Zhang K, Yang Z, Başar T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of Reinforcement Learning and Control. 2021;p. 321–384.

[27] Liu Y, Luo Y, Zhong Y, Chen X, Liu Q, Peng J. Sequence modeling of temporal credit assignment for episodic reinforcement learning. arXiv preprint arXiv:190513420. 2019;.

[28] Gupta D, Mihucz G, Schlegel M, Kostas J, Thomas PS, White M. Structural Credit Assignment in Neural Networks using Reinforcement Learning. Advances in Neural Information Processing Systems. 2021;34.

[29] Mao W, Gratch J. The social credit assignment problem. In: International Workshop on Intelligent Virtual Agents. Springer; 2003. p. 39–47.

[30] Harati A, Ahmadabadi MN, Araabi BN. Knowledge-based multiagent credit assignment: A study on task type and critic information. IEEE systems journal. 2007;1(1):55–67.

[31] Skinner BF. The behavior of organisms: An experimental analysis. BF Skinner Foundation; 2019.

[32] Zhou M, Liu Z, Sui P, Li Y, Chung YY. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. Advances in Neural Information Processing Systems. 2020;33:11853–11864.

[33] Salimibeni M, Mohammadi A, Malekzadeh P, Plataniotis KN. Multi-Agent Reinforcement Learning via Adaptive Kalman Temporal Difference and Successor Representation. Sensors. 2022;22(4):1393.

[34] Devlin S, Yliniemi L, Kudenko D, Tumer K. Potential-based difference rewards for multiagent reinforcement learning. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems; 2014. p. 165–172.

[35] Wang J, Zhang Y, Kim TK, Gu Y. Shapley Q-value: A local reward approach to solve global reward games. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34; 2020. p. 7285–7292.

[36] Fu WT, Anderson JR. Solving the credit assignment problem: explicit and implicit learning of action sequences with probabilistic outcomes. Psychological research. 2008;72(3):321–330.

[37] Rahaie Z, Beigy H. Expertness framework in multi-agent systems and its application in credit assignment problem. Intelligent Data Analysis. 2014;18(3):511–528.

[38] Chen SA, Tangkaratt V, Lin HT, Sugiyama M. Active deep Q-learning with demonstration. Machine Learning. 2020;109(9):1699–1725.

[39] Tian J, Yu Y, Li T, Zhou Y, Li J, Wang X, et al. A cooperative game model with bankruptcy theory for water allocation: a case study in China Tarim River Basin. Environmental Science and Pollution Research. 2022;29(2):2353–2364.

[40] Abedin SF, Hong CS. Bankruptcy Game based Computational Resource Scaling in Mobile Edge Computing. 2018 Korean Network Operations and Management. 2018;.

[41] Thomson W. How to divide when there isn't enough. 62. Cambridge University Press; 2019.

[42] Maynard Smith J. Evolutionary game theory. In: Vito Volterra symposium on mathematical models in biology. Springer; 1980. p. 73–81.

[43] Garay J, Móri TF. Best Reply Player Against Mixed Evolutionarily Stable Strategy User. Bulletin of Mathematical Biology. 2022;84(1):1–21.

[44] Hofbauer J, Sigmund K. Evolutionary game dynamics. Bulletin of the American mathematical society. 2003;40(4):479–519.

[45] Křivan V, Galanthay TE, Cressman R. Beyond replicator dynamics: From frequency to density dependent models of evolutionary games. Journal of theoretical biology. 2018;455:232–248.

[46] Nash J. Non-cooperative games. Annals of mathematics. 1951;p. 286–295.

[47] Jiang AX, Leyton-Brown K. A Tutorial on the Proof of the Existence of Nash Equilibria. University of British Columbia Technical Report TR-2007-25 pdf. 2009;14.