QBRT: Bias and Rising Threshold Algorithm with Q-Learning
Implementation of the Tower of Hanoi

Ryo Ogino(✉), Masao Kubo, and Hiroshi Sato

National Defense Academy, 1-10-20, Hashirimizu, Yokosuka, Kanagawa, Japan
em60010@nda.ac.jp

Abstract. In multi-agent reinforcement learning, the problems of non-stationarity of the environment and scalability have long been recognized. As a first step toward solving these problems, this paper proposes a learning model, the BRT Algorithm with Q-Learning (hereafter, QBRT), based on the Bias and Rising Threshold (hereafter, BRT) algorithm, which can solve best-of-n problems where the number of options n is greater than 2 (hereafter, best-of-n problems (n >> 2)). This model is characterized by the fact that all of the agents that make up the herd agree in advance on what action the herd will take next. We thought that the problem of non-stationarity could be ameliorated to some extent by having all agents follow the same policy. On the other hand, the time it takes for agents to reach an agreement with each other generally tends to increase as the number of agents increases. In contrast, if BRT is used as a base, the time required for agreement could be kept almost constant even if the number of agents increases.We will validate the problem with an experiment using Tower of Hanoi by Multiagent (hereafter THM), a best-of-n problem (n >> 2) based on the classic puzzle "Tower of Hanoi", which is a flock coordination problem.

Keywords: Multi-agent · Reinforcement learning · Best-of-n problem · Tower of Hanoi

1 Introduction

Among machine learning, reinforcement learning is being applied and studied for various tasks because of its ability to maximize earned rewards in the future. Here, the subject of reinforcement learning is called the agent. There are multi-agent systems in which multiple agents interact in the same environment and learn how to solve a task simultaneously. In recent years, there have been many attempts to extend single-agent reinforcement learning algorithms for multiple agents [1]. However, direct implementation of single-agent reinforcement learning algorithms on multiple agents is known to cause problems such as environment non-stationarity and scalability [1]. Non-stationarity is the problem that even if one agent takes certain all agents is the joint action, it will not converge to the optimal solution because different rewards are given depending on the

Y. Kambayashi et al. (Eds.): AICON 2022, LNICST 477, pp. 45–59, 2023.
https://doi.org/10.1007/978-3-031-29126-5_4

actions taken by other agents and the environment is not stationary [2]. Therefore, previous studies have proposed solutions to non-stationarity, such as a method that combines the Q tables of all agents [3] and a method that differentiates between the rate of increase and decrease of Q values [4]. However, these studies have only been implemented with a small number of agents (2–4) and have not been validated with a larger number of agents. On the other hand, the scalability problem is the problem that the size of the action space increases exponentially with the number of agents when the action space of all agents is in the joint action space of all agents [5]. Therefore, scalability solutions have been proposed, including those based on local observations [6] and those based on deep reinforcement learning [7]. Wang et al.. Also point out that there is a trade-off between the non-stationarity problem and the scalability problem [2], and an algorithm that can solve both of these problems is needed.

Here, we introduce the Bias and Rising Threshold (hereafter, BRT) algorithm [8] to the agent as a solution to the problems of non-stationarity and scalability. The BRT algorithm was proposed by Phung et al.. to enable a swarm of robots to handle a large number of options (n >> 2) in a best-of-n problem. Experiments have shown that this algorithm is able to select an alternative that is generally constant in time without being significantly affected by the number of agents in the herd. On the other hand, the BRT algorithm does not have a mechanism to take advantage of past experience and is an approximately random search, which can be inefficient in some cases. Therefore, we propose a multi-agent learning model that combines reinforcement learning and the BRT algorithm, in which all the constituent agents agree in advance on what action the herd will take next. We thought that the problem of non-stationarity could be ameliorated to some extent by having all agents follow the same policy. On the other hand, the time it takes for agents to reach an agreement with each other generally tends to increase as the number of agents increases. In contrast, if BRT is used as a base, the time required for agreement could be kept almost constant even if the number of agents increases. As a first step, we propose a learning BRT algorithm, BRT Algorithm with Q-Learning (hereafter QBRT), which can solve best-of-n problems (hereafter best-of-n problems (n >> 2)) where the number of options n is greater than 2.

Here, as a best-of-n problem (n >> 2), we experimented with Tower of Hanoi by Multiagent; THM, based on the classic puzzle Tower of Hanoi. If a conventional herd of BRT agents with random behavior is applied to THM as it is, it is not easy to solve the puzzle because the state of the tower changes each time due to the herd's behavior. On the other hand, when the learning BRT algorithm is applied to THM, the learning results in solving the puzzle much faster than the traditional BRT algorithm. QBRT, an application of the BRT algorithm, was able to achieve a herd that could promptly choose an appropriate option from a large number of options depending on the environment, even for a herd with a large number of agents. QBRT can be a stepping stone to reinforcement learning models that can address non-stationarity and scalability issues. In the future, we hope to see applications that handle more options with a larger number of agents, such as automated delivery and search and rescue at disaster sites.

Subsequent parts of this section are organized as follows. In Sect. 2, a brief introduction to related research is given. In Sect. 3, a learning BRT algorithm is proposed, In Sect. 4, the effectiveness of the proposed method is verified by computer experiments. Finally, in Sect. 5, we present our conclusions.

2 Related Studies

We propose a learning BRT algorithm, QBRT, which can solve the best-of-n problem ($n >> 2$). As a related study, this chapter begins with a brief introduction to previous research on environment non-stationarity and scalability issues in Sect. 2.1. The BRT algorithm is described next in Sect. 2.2. Section 2.3 describes the best-of-n problem addressed in this study. Finally, we describe Q-learning, which was used for the learning rules of the learning BRT algorithm in Sect. 2.4.

2.1 Prior Work on Environment Non-stationarity and Scalability Issues

According to a review by Canese et al. [1], the issues that must be considered when extending from single-agent reinforcement learning algorithms to multi-agent scenarios are the non-stationarity of the environment and scalability issues. Non-stationarity of the environment is the problem that even if one agent takes certain same action, it will not converge to the optimal solution because different rewards are given by the actions taken by other agents and the environment will not be stationary [2]. On the other hand, the scalability problem is the problem that the size of the action space increases exponentially with the number of agents when the action space of all agents is in the joint action space of all agents [5]. The following is a brief review of previous studies on these issues.

Studies on non-stationarity include those by Matta et al. [3] and Matignon et al. [4]. Matta et al. used a "centralized aggregation center" to combine the Q-value tables of all agents and create a global Q-value table containing the highest and lowest Q-values, representing the most highly rated iterations [3]. Matignon et al. propose an optimistic agent that makes the rate of decrease in Q values smaller than the rate of increase when Q values decrease due to learning [4]. This reduces the possibility of being punished for the bad behavior of other agents, even if they have made the best options. On the other hand, these studies have only been implemented with a small number of agents (2–4) and have not been validated with a larger number of agents.

As a study of scalability issues, Kar et al. proposed QD-learning, a distributed version of Q-learning, in which the size of the action space is limited under the assumption that each agent knows only its local actions and rewards and that the inter-agent communication network is weakly connected QD-learning, a distributed version of Q-learning, has been proposed [6]. On the other hand, a method to approximate Q values using deep reinforcement learning (DRL) has also been proposed [7].

2.2 BRT Algorithm

The Bias and Rising Threshold (hereafter, BRT) algorithm [8] is a herd decision-making framework that can handle a large number of options ($M \geq 2$) in a best-of-n problem. A swarm $A = \{A_i : i = 1, ..., N\}$ consisting of N agents is given an action set $O = \{o_j : j = 1, ..., M\}(M \geq 2)$. If agent A_i satisfies Eq. (1), the option $O_i(t + 1)$ to be selected at the next time continues to be the currently selected option $O_i(t)$; otherwise, $O_i(t + 1)$ is stochastically changed to an option other than $O_i(t)$.

$$\frac{n(O_i(t))}{N} \geq \theta_i + \tau \cdot c_i(t) \cdot (t - t_{i,last}(t)) \tag{1}$$

where $n(O_i(t))/N$ is the percentage of agents in the entire population who choose the same option as themselves, θ_i is the bias value of the individual attribute ($0 < \theta_i < 1$), and τ is a constant corresponding to the assumed value of the increase in supporters. $t_{i,last}(t)$ is the time when agent A_i last changed its option, and $(t - t_{i,last}(t))$ is the time it continues to choose the same option, represented by Eq. (2).

$$t_{i,last}(t) = \begin{cases} t & O_i(t) \neq O_i(t-1) \\ t_{i,last}(t-1) & otherwise \end{cases} \tag{2}$$

$c(t)$ is an evaluation function that is zero when the option expressed in Eq. (3) is feasible.

$$c_i(t) = c(t) = \begin{cases} 0 \ \forall i, O_i \equiv o_{goal} \\ 1 \quad otherwise \end{cases} \tag{3}$$

o_{goal} is a suitable option $\left(o_{goal} \in O\right)$ to look for. Each agent does not know the o_{goal} in advance and decides to continue or change its option, taking into account the overall trend of the swarm. The state in which all agents select the same option is called the consensus state; otherwise, it is called the non-consensus state. Various behaviors can occur depending on the distribution of this individual attribute θ_i. Phung et al. report that using the distribution in Eq. (4), if the agreed-upon option is not o_{goal}, then over time, Eq. (1) is no longer satisfied and the state of agreement is broken, instantly causing the herd to agree again on a different option, and this creates behavior that repeats until agreement on o_{goal} is reached.

$$n(\theta_i) = \begin{cases} 0 & \theta_i \leq 0 \\ k_1 N \theta_i^2 & 0 < \theta_i < \frac{1}{M} \\ k_1 N \left(\theta_i - \frac{2}{M}\right)^2 & \frac{1}{M} < \theta_i < \frac{2}{M} \\ 0 & \frac{2}{M} \leq \theta_i \end{cases} \tag{4}$$

where $k_1 = (3M^3)/2$ is the normalization term.

This algorithm can be used in swarms with a large number of agents or with variable numbers of agents since experiments have shown that the number of agents, N, has little effect on the time required for consensus building. Because the BRT algorithm has the property that the swarm can agree on a single option even as the number of agents increases, a learning algorithm based on the BRT algorithm can be an algorithm that addresses the problem of non-stationarity and scalability in the environment.

2.3 Best-of-n Problem

The best-of-n problem is one of the problems related to swarm decision making. The problem is that the swarm chooses the best option to meet the current needs of the swarm from the n options available to it. There are still few studies dealing with the best-of-n problem with n >> 2 according to Valentini et al. [9]. Here we treat THM as a best-of-n problem with n >> 2 best-of-n problem, which is an application of the classical puzzle Tower of Hanoi on the theme of swarm coordination problems.

2.4 Q Learning

Q learning [8] is a type of reinforcement learning technique in which the Q value, the future value of a state/action pair, is empirically acquired through trial and error. The Q value is the total amount of reward that a given action in a given state can obtain in the future and is updated by Eq. (5). The estimated Q value can be used to select actions that will yield higher value in the future.

$$Q(s_t, o_t) \leftarrow (1-\alpha) \cdot Q(s_t, o_t) + \alpha\left(r_t + \gamma \cdot \max\left(Q\left(s_{t+1}, o'\right)\right)\right) \tag{5}$$

where s_t and a_t are the states (the swarm board in the case of THM) and behavior at time t, and r_t is the reward earned. $max(Q(s_{t+1}, o'))$ is the maximum Q value in the transition destination state, and this term is used to learn an action sequence that considers future rewards to be earned. α And γ are parameters for learning, called the learning rate and discount rate, respectively. A flowchart of Q learning is shown in Fig. 1.

Here, ε-greedy was employed for the agent's action selection. The agent randomly chooses an action with probability ε and chooses the action with the largest Q value with probability $(1-\varepsilon)$.

3 Proposed Method

Because the BRT algorithm introduced in Sect. 2.2 is a trial-and-error search, even in environments that are new to the swarm, the swarm can find the optimal action options over time, as long as the action options are appropriate to the environment. However, they are unable to make use of this experience and thus cannot effectively adapt to similar environments. On the other hand, since the BRT algorithm agrees in advance which option to select and the number of agents has little impact on the consensus-building time, it is an algorithm that has a high potential to address the issues of non-stationarity and scalability of the environment.

Therefore, in this chapter, we propose QBRT, a learning BRT algorithm that introduces Q learning into the BRT agent and allows it to choose the best option more efficiently.

3.1 QBRT

The BRT algorithm introduced in Sect. 2.2 always randomly selects the next action $O_i(t+1)$ from the remaining set of actions $(O - O_i(t))$ when agent A_i changes its option.

$$O_i(t+1) \in O - O_i(t) \tag{6}$$

This may result in inefficient searches in some environments that have already been experienced. Therefore, QBRT adds an element of reinforcement learning to the part of the BRT agent's behavior that is changed to improve the efficiency of the search. Here, Q learning was chosen as the reinforcement learning method to be added. Here, agent A_i changes its options according to Eq. (7) instead of Eq. (6).

$$\begin{cases} O_i(t+1) \in O - O_i(t) & probability\,(1-\varepsilon_i) \\ O_i(t+1) = \underset{o \in O(s_{t+1})}{\text{argmax}}\, Q_i(s_{t+1}, o) & probability\,\varepsilon_i \end{cases} \tag{7}$$

where s_t is the state of the environment observed by the swarm at time t, and $\underset{o \in O(s_{t+1})}{argmax}\, Q_i(s_{t+1}, o)$ is the maximum Q value of agent A_i in the transition destination environment. Here, the Q value of agent A_i is stored for each combination of observed environment state $S = s_l (l = 0 \sim L-1)$ and option $O = o_m (m = 0 \sim M-1)$ as in Eq. (8). L is the number of possible states of the environment.

$$Q_{i_{s_l, o_m}} = Q_i(s_l, o_m) \tag{8}$$

Agent A_i acts only when the swarm's option reaches a consensus state, and updates the Q value according to Eq. (4) in Sect. 2.2, regardless of whether the option is changed or not (Fig. 2(10)). On the other hand, if the swarm does not consent, it does not act and re-selects (Eq. (9)). The reward r_i given in process (9) in Fig. 2 is set by the designer to an appropriate reward for the goal state (final board in THM). This allows each agent in the swarm to act and learn only when it makes the same options as the other agents, thus acting and learning cohesively as a swarm. As a result, if enough exploration and learning takes place, the swarm can obtain Q values through trial-and-error iterations with the environment to obtain an optimal series of options.

$$\begin{cases} execution & \exists j, \forall i, O_i(t) = o_j \\ reselection & otherwise \end{cases} \tag{9}$$

A conceptual diagram of the proposed method is shown in Fig. 2. In the traditional BRT algorithm, when changing options, the only change is random, whereas the learning BRT algorithm adds a Q learning component to each agent (Fig. 2(6), (9), (10)).

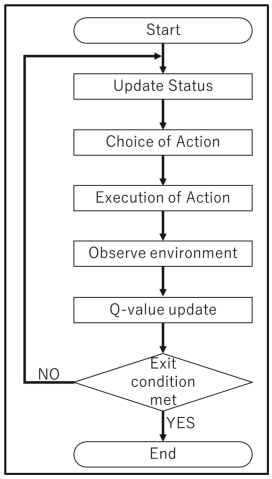

Fig. 1. Flowchart of Q learning

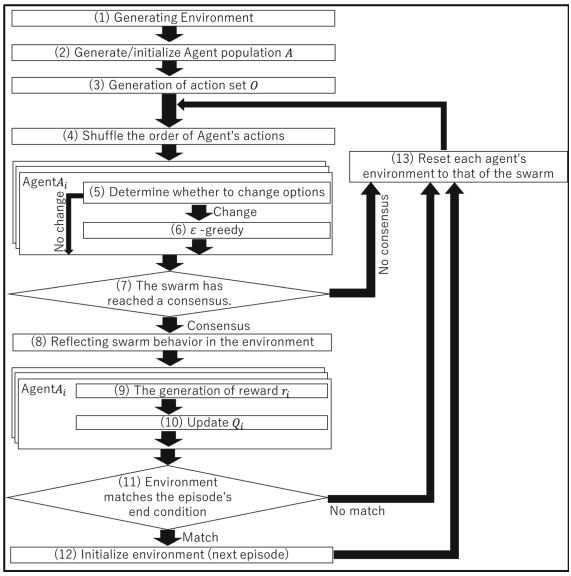

Fig. 2. Conceptual diagram of QBRT

3.2 Behavior of Swarms Implementing QBRT

The agent implementing the QBRT proposed here observes the state of the environment (the THM board) and presents the options it wishes to execute to the swarm as a whole. Then, only when all the agents' presentations agree on one option, i.e., when the swarm has reached a consensus state, does the swarm actually act and the agents learn. Therefore, each agent acts and learns by incorporating the actions of other agents, so that the entire swarm behaves like a single agent, making the algorithm capable of dealing with the non-stationarity of the environment. In addition, because it is based on the BRT algorithm, the time required to reach a consensus state does not change significantly even when the number of agents increases, so it can generally be said to address scalability issues.

In Sect. 4, as a first step to verify the above, THM, as a cooperative problem for swarms, is run on a swarm that implements QBRT.

4 Experiments

This chapter first describes the environment of the experiment, Tower of Hanoi by Multiagent (hereafter, THM). Next, we will verify the effectiveness of the QBRT by conducting computer experiments playing THM with each of the conventional BRT algorithm and QBRT, which adds Q learning to the BRT algorithm.

4.1 Experimental Environment

4.1.1 Tower of Hanoi

A best-of-n problem with n >> 2 and requiring coordinated swarm behavior is used below as an application of the Tower of Hanoi, a type of classical puzzle. In this section, before describing the assignment, we will first briefly describe the tower of Hanoi. The puzzle $T(P, D)$ consists of a set of multiple piles $P \in \{P_0, P_1, \ldots, P_{p-1}\}$ and a set of multiple disks of different sizes $D \in \{D_0, D_1, \ldots, D_{d-1}\}$, as shown in Fig. 3. Here, the board that is first given to the player is called the initial board, and the board that is the condition for clearing the puzzle is called the final board. There are also three standard rules,

1. Only one disc at the top of any pile may be moved per operation.
2. Do not place a larger disk on top of a smaller disk.
3. Disks shall not be placed anywhere other than the stakes.

The difficulty level of the Tower of Hanoi can be adjusted by changing the number of piles $p = |P|$, the number of disks $d = |D|$, and the initial and final board surfaces, and it is generally known that the larger p, the lower the difficulty level and the larger d, the higher the difficulty level.

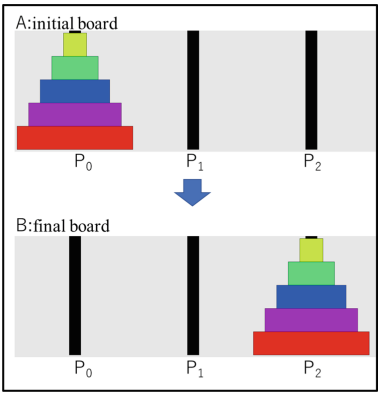

Fig. 3. Tower of Hanoi: Example of $T(3, 5)$

4.1.2 THM: The Tower of Hanoi Multi-agent

Tower of Hanoi is essentially a puzzle game played by a single player; to make it playable by a large number of agents, the following changes were made. In the following, this is referred to as THM (Tower of Hanoi by Multiagent). First, THM prepares a pair of Hanoi towers (Fig. 4(1)). Each agent in the swarm then observes the current board (Fig. 4(2)) and decides which disc to move next and where to move it (Fig. 4(3)). This is the move of agent A_i. Next, the moves of all agents are counted, and if all agents agree on a move,

the board reflects the move accordingly as a "swarm consensus" (Fig. 4(4)). The disc moves to select a move on a new board (Fig. 4(5)). On the other hand, if the hands of all agents do not match, the board of the Tower of Hanoi is not changed. However, time elapses, and again all agents re-determine their moves (left-pointing arrows in Fig. 4(3)).

Since we want to prototype a concise BRT agent that performs Q learning, we evaluated the board only when the board changed. If the BRT agent's action is in a non-consensus state, the board is not evaluated.

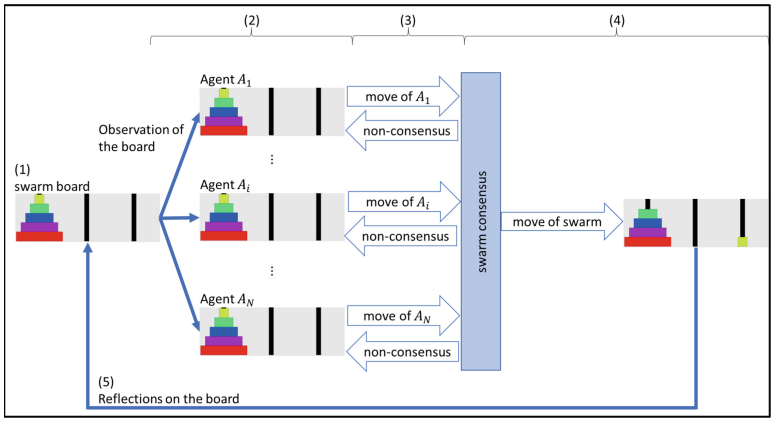

Fig. 4. Conceptual diagram of the execution of the Tower of Hanoi as a swarm

4.1.3 THM Settings

In the experiments conducted in this chapter, the number of piles is p and the number of disks is d. In THM, the state of reinforcement learning corresponds to the board, the state set S is $\left\{s_0, \ldots, s_{p^d-1}\right\}$ and the action set O is $\left\{o_0, \ldots, o_{p\cdot(p-1)-1}\right\}$. The initial board was unified with all disks at pile P_0 on the left side as shown in Fig. 3A, and the final board was unified with all disks at pile P_2 on the right side as shown in Fig. 3B. The total number of possible states $|S|$ is p^d and the number of elements of the action set $|O|$ is $p \cdot (p-1)$.

Figure 5 shows an example of the state transition of the THM board S observed by the swarm for $p = 3, d = 5$, and option O. If option o_1 (move the disk from pile P_0 to pile P_2), which can move the disk in state s_0 of the board, is executed, the state transitions from s_0 to s_1. If an option that cannot move the disk in state s_0 (e.g., o_2 (move the disk from pile P_1 to pile P_0)) is executed, the state is treated as a transition from s_0 to s_0. However, the agent shall not know in advance whether any of the options can move the disk in a given state.

The series of trials from the initial board to the final board is referred to as an episode in the following. When the Tower of Hanoi moves to the final board, the board is initialized and the next episode begins.

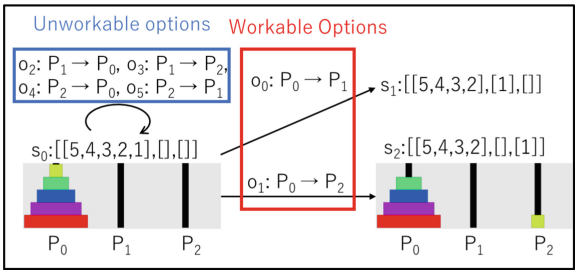

Fig. 5. Examples of THM state transitions and options

4.2 Experiment at THM with $p = 3, d = 3$

In this section, the experiment is performed on a THM with $p = 3$ and $d = 3$. In this case, the minimum number of moves required to move all disks from pile P_0 to pile P_2 is seven moves, following the basic rule described in Sect. 4.1.1.

4.2.1 Performing THM with the Conventional BRT Algorithm

To compare with QBRT, we first run the THM with the conventional BRT algorithm.

The number of agents N of the BRT algorithm treated in this section is set to 30. Also, the number of options O, M, is $p \cdot (p-1)$ (disks from P_0 to P_1, …, from P_0 to P_{p-1}, …, from P_{p-1} to P_0, …, from P_{p-1} to P_{p-2}).

According to these conditions, the results of 40 episodes of THM run by the conventional BRT algorithm are shown in Fig. 6.

A run with the conventional BRT algorithm, which agrees on nearly random options, resulted in a maximum of 1394 moves, a minimum of 21 moves, and an average of 302.1 moves. The number of moves required to complete an episode was significantly greater than the minimum number of moves is seven, and there was no tendency for the number of moves to shorten as the episodes were repeated. The average number of non-consensus moves per move, which is the time it takes to reach an agreement, was 532.1.

4.2.2 Performing THM with QBRT

The setting of the number of options M in the QBRT is the same as in Sect. 4.2.1. However, each agent has its Q_i value, learning rate α_i, discount rate γ_i, and random learning rate ε_i, since it selects itself according to Eq. (6) proposed in Sect. 3. Each agent acts collectively as a swarm using the BRT algorithm, but because no two individuals are the same in the real world, each agent receives the results differently. To reproduce and verify this, we conducted experiments with swarms in which the parameters α_i, γ_i,

and ε_i were set randomly and the agents differed individually, and with swarms in which all the parameters were unified to simplify the experiment and no individual differences were observed. Here, α_i, γ_i, and ε_i are assumed to be randomly determined by a Gaussian distribution with mean 0.5 and variance 0.5/3, with each value greater than 0 and less than 1 and a bell-shaped distribution. The rewards for each Q learning were set as shown in Table 1. The "workable options" in the table are those options that can move the disks. On the other hand, "unworkable options" are those options that cannot move the disk. It also gives a large positive reward to the swarm if it chooses the "option to reach the final state" and it is carried out.

Table 1. Setting Rewards for Q-learning at THM with $p = 3, d = 3$

Options	Workable options	unworkable options	Option to reach the final state
Rewards	0	0	1000

Furthermore, agent A_i in QBRT has variables as shown in Table.2

Table 2. Variables and initial values of agent A_i in QBRT

Variable name	Description	Initial value
τ	Constant indicating predicted increase in support	$0.01/M$
t_{last}	Time of last option change	0
O	Possible options	$O \in \{o_0, ..., o_{p\cdot(p-1)-1}\}$
Q_i	Q-table for Q-learning	0
α_i	Q-learning learning rate	Randomly determined by Gaussian distribution with mean 0.5 and variance 0.5/3
γ_i	Q-learning discount rate	
ε_i	Probability of randomly choosing an option in ε-greedy	

The Hanoi tower was run for 40 episodes with four QBRT combinations of individual differences in swarms with $N = 30$ and 100 agents, according to the above conditions. These are summarized in Fig. 7.

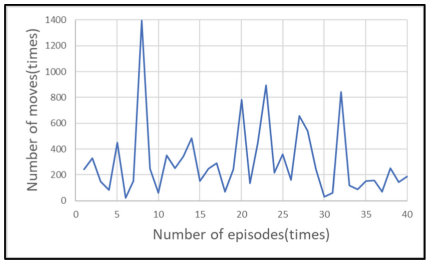

Fig. 6. Results of running conventional BRT algorithm on THM with $p = 3, d = 3$

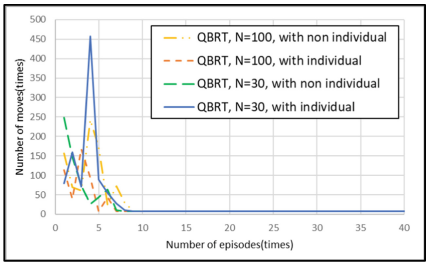

Fig. 7. Results of running the QBRT algorithm on THM with $p = 3, d = 3$ (Color figure online)

All QBRT swarms are now able to finish an episode in 7 moves, the shortest number of moves within 10 episodes. In all conditions, the number of moves was significantly reduced from the average of runs with the conventional BRT algorithm. The average number of non-consensus moves per swarm consensus, which is the time it takes to reach consensus, is 500.9 for the swarm with $N = 100$ non-individual differences (yellow dashed line in Fig. 7), 505.4 for the swarm with $N = 100$ individual differences (red dashed line in Fig. 7), 513.4 for the swarm with $N = 30$ non-individual differences (green dashed line in Fig. 7), 539.4 for the swarm with $N = 30$ individual differences (the blue straight line in Fig. 7), which is about the same as the number of swarms executed by the conventional BRT or slightly reduced. This is presumably because learning has made it easier for the agent to select certain options.

These results suggest that the same trend would be true even if the number of agents is further increased, and thus QBRT was able to improve the non-stationarity and scalability problems to some extent regardless of the number of agents.

4.3 Experiment at THM with $p = 3, d = 5$

In this section, the experiment is performed on a THM with $p = 3$ and $d = 3$. In this case, the minimum number of moves required to move all disks from pile P_0 to pile P_2 is 31 moves, following the basic rule described in Sect. 2.5.1.

4.3.1 Performing THM with the Conventional BRT Algorithm

To compare with QBRT, we first run the THM with the conventional BRT algorithm. The settings for the number of agents N and the number of options O, M, are the same as in Sect. 4.2.1.

According to these conditions, the results of 40 episodes of THM run by the conventional BRT algorithm are shown in Fig. 8.

A run with the conventional BRT algorithm, which agrees on nearly random choices, resulted in a maximum of 13372 moves, a minimum of 579 moves, and an average of 4142.2 moves. The number of moves required to complete an episode was significantly greater than the minimum number of moves, 31, and as with Sect. 4.2.1, there was no tendency for the number of moves to shorten over the course of the episode.

4.3.2 Performing THM with QBRT

The variable settings for the number of options M and agent A_i in the QBRT are the same as in Sect. 4.2.2. The number of agents $N = 30$ and the rewards for each Q learning were set as shown in Table 3. The "workable options" in the table are those options that can move the disks. On the other hand, "unworkable options" are those options that cannot move the disk. It also gives a large positive reward to the swarm if it chooses the "option to reach the final state" and it is carried out. Also, experiment with negative rewards for swarms with the intention of speeding up learning.

Table 3. Setting Rewards for Q-learning at THM with $p = 3, d = 5$

	Options	Workable options	unworkable options	Option to reach the final state
Rewards	No Negative Reward	0	0	1000
	Negative Reward	−0.1	−100000	1000

According to these conditions, 250 episodes of the Tower of Hanoi were run by QBRT with four different combinations of individual differences and negative rewards. These are summarized in Fig. 9.

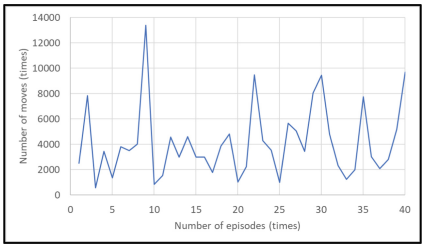

Fig. 8. Results of running conventional BRT algorithm on THM with $p = 3, d = 5$

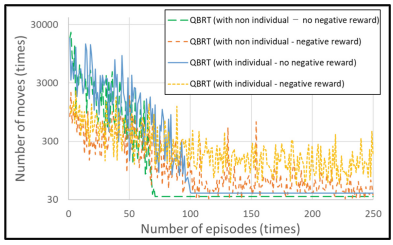

Fig. 9. Results of running the QBRT algorithm on THM with $p = 3, d = 5$ (Color figure online)

In the QBRT with non-individual differences and no negative reward (Fig. 9 green dashed line), it took 34 moves in about 70 episodes from the beginning of the experiment, and in the QBRT with individual differences and no negative reward (Fig. 9 yellow dashed line), it took 39 moves in about 100 episodes from the beginning of the experiment. Episodes could be completed. In the QBRT with non-individual differences and with negative rewards (Fig. 9 red dashed line), the shortest number of moves, 31, appeared in the first 115 episodes of the experiment, and since then the episode has been completed in the shortest number of moves several times. Q learning also allowed for a

significant reduction from the average of runs using the conventional BRT algorithm in all conditions, provided that sufficient learning had progressed.

In the runs with swarms with no negative rewards (Fig. 9 green dashed line and red dashed line), learning stopped when the shortest number of moves was not found, but this is thought to be because once some better series are found, positive rewards continue to be given to that series, which prevents further search for better series. On the other hand, in the runs with swarms with negative reward (Fig. 9 dashed yellow line and solid blue line), the search in the initial phase progressed more efficiently than those with no negative reward, so we can expect the learning to proceed even faster and more consistently if the negative reward is adjusted.

This experiment confirmed that QBRT is able to solve the puzzle even as the THM difficulty level increases. It was also more efficient than the BRT algorithm in solving the cooperative problem where all agents make the same move. Since QBRT is based on the BRT algorithm, which has the property that the time required for agreement does not change significantly as the number of agents increases, the learning speed is not expected to change significantly even if it is run with a larger number of agents. Therefore, we were able to implement a learning algorithm that solves the problems of non-stationarity and scalability in the cooperative problem. On the other hand, the issue of competition is unknown. Therefore, it is necessary to verify in the future whether QBRT is effective against the contention problem and whether it works without problems even when executed with a large number of agents.

5 Conclusion

Here, we proposed QBRT as a reinforcement learning model that can deal with the non-stationarity of the environment and scalability issues by applying the BRT algorithm, which has the property that the time required for agreement does not change significantly even when the number of agents increases. Experimental results with THM show that QBRT solves puzzles more quickly than the BRT algorithm. Learning could also be advanced by increasing the number of agents and the difficulty level of the THM. Therefore, it can be said that we were able to implement a learning algorithm that improves non-stationarity and scalability issues to some extent in the cooperative problem. In the future, it is necessary to verify whether QBRT is also effective for competing problems where the number of agents increases or agents move differently.

References

1. Canese, L., et al.: Multi-agent reinforcement learning: a review of challenges and applications. Appl. Sci. **11**(11), 4948 (2021)
2. Wang, Y., Damani, M., Wang, P., Cao, Y., Sartoretti, G.: Distributed reinforcement learning for robot teams: a review. arXiv preprint arXiv:2204.03516 (2022)
3. Matta, M., et al.: Q-RTS: a real-time swarm intelligence based on multi-agent Q-learning. Electron. Lett. **55**(10), 589–591 (2019)
4. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 64–69. IEEE (2007)

5. Qu, G., Lin, Y., Wierman, A., Li, N.: Scalable multi-agent reinforcement learning for networked systems with average reward. Adv. Neural. Inf. Process. Syst. **33**, 2074–2086 (2020)
6. Kar, S., Moura, J.M., Poor, H.V.: QD-learning: a collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations. IEEE Trans. Signal Process. **61**(7), 1848–1862 (2013)
7. Palmer, G., Tuyls, K., Bloembergen, D., Savani, R.: Lenient multi-agent deep reinforcement learning. arXiv preprint arXiv:1707.04402 (2017)
8. Phung, N.H., Kubo, M., Sato, H.: El Farol Bar problem by agreement algorithm based on trial and error behavior at the macro lever. In: Proceedings of the 22nd Asia Pacific Symposium on Intelligent and Evolutionary Systems (2018)
9. Valentini, G., Ferrante, E., Dorigo, M.: The best-of-n problem in robot swarms: formalization, state of the art, and novel perspectives. Front. Robot. AI **4**, 9 (2017)