information

MDPI

Article

Fuzzy Reinforcement Learning and Curriculum Transfer Learning for Micromanagement in Multi-Robot Confrontation

Chunyang Hu [1] and Meng Xu [2,*]

[1] School of Computer Engineering, Hubei University of Arts and Science, Xiangyang 441053, China; huchunyang@hbuas.edu.cn
[2] School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China
* Correspondence: menghsu@mail.nwpu.edu.cn

Received: 12 October 2019; Accepted: 30 October 2019; Published: 2 November 2019

Abstract: Multi-Robot Confrontation on physics-based simulators is a complex and time-consuming task, but simulators are required to evaluate the performance of the advanced algorithms. Recently, a few advanced algorithms have been able to produce considerably complex levels in the context of the robot confrontation system when the agents are facing multiple opponents. Meanwhile, the current confrontation decision-making system suffers from difficulties in optimization and generalization. In this paper, a fuzzy reinforcement learning (RL) and the curriculum transfer learning are applied to the micromanagement for robot confrontation system. Firstly, an improved Q-learning in the semi-Markov decision-making process is designed to train the agent and an efficient RL model is defined to avoid the curse of dimensionality. Secondly, a multi-agent RL algorithm with parameter sharing is proposed to train the agents. We use a neural network with adaptive momentum acceleration as a function approximator to estimate the state-action function. Then, a method of fuzzy logic is used to regulate the learning rate of RL. Thirdly, a curriculum transfer learning method is used to extend the RL model to more difficult scenarios, which ensures the generalization of the decision-making system. The experimental results show that the proposed method is effective.

Keywords: multi-robot confrontation; fuzzy reinforcement learning; curriculum transfer learning; neural network

1. Introduction

1.1. The Robot Confrontation System

The aim of Artificial Intelligence (AI) is to develop a computer program that can realize human-level intelligence, self-consciousness, and knowledge application. Multi-Agent Systems (MAS) have recently become popular as an important means for the study of confrontational decision-making, strategic behavior in electricity markets [1], and so on. As an effective tool for AI research, the simulation platform allows the agent to rely on the predefined algorithm to perform various kinds of actions in a certain scenario, which plays a role in replacing the real physical environment [2]. These simulations can not only be used as substitutes for the physical environment that can touch but can also be set to some scenarios that cannot exist in real-life depending on our imagination [3]. Recently, computer games have been used for AI research, which helps the agent to grow since its birth, including the Atari video games [4], the imperfect information game and so on [5]. The Multi-Robot Confrontation [6], as a platform to imitate real battlefields, provides convenience for military command, situation assessment, and intelligent decision-making, and is also an effective platform to develop AI applications.

In this paper, we focus on the robot confrontation system to explore an effective learning method for agent control. In the robot confrontation system, the condition for the agent is similar to a decision-making (RTS) game [7]. Real-time strategic games are different from taking turns to play in board games because it runs in real-time and requires continuous decision-making for agents. RTS game provides a physical-based simulation environment to study the control of agents with different learning levels, such as StarCraft [8], Dota 2, and Namco. Reinforcement learning (RL) [9] is an effective machine learning method and the goal of reinforcement learning for an agent is to learn an optimal action strategy and obtain optimal rewards. This year, it has attracted extensive attention from scholars [10–12].

1.2. Machine Learning Algorithms

Machine learning algorithms have been applied in many fields, such as electricity price forecasting [13]. Generally, machine learning algorithms can be divided into three categories: supervised learning, unsupervised learning and reinforcement learning. Reinforcement learning includes value-based RL method [14], policy-based RL method [15], and the hybrid method named actor-critic [16]. Generally, the RL method is very suitable for sequential real-time scenarios when these scenarios are modeled as the Markov decision-making sequences or semi-Markov decision sequences. [17] uses a Q-learning method for evolving a few basic behaviors and learning to flight for bots in the FPS game of Counter-Strike. The author has carried out experiments on how bots can evolve its behavior in a more detailed model of the world using the knowledge learned from abstract models. For the RL method, the learning rate is important, because it affects the performance and convergence speed of the RL system. However, the performance of the fixed learning rate for RL methods often encounters bottlenecks. Reference [18] uses a neural network as a function approximator to estimate the action-value function for a group of units in StarCraft micromanagement. As an effective state representation, the neural network breaks down the dilemma caused by the large state-action space in the game scenarios. However, for the neural network, the overcorrection of the weights at one time may lead to the dilemma of learning instability, which will lead to low learning efficiency. The adaptive momentum [19] breaks down the dilemma between stability and efficiency.

Recently, deep learning has achieved record-breaking performance in a variety of complex scenarios, and which provides an opportunity of scaling to problems with infinite state spaces for the RL algorithms [20]. For StarCraft micromanagement, reference [21] uses the actor-critic algorithm and Multi-agent Bidirectionally-Coordinated Network (BiCNet) to control bot's behavior. They model the dependency of units by the BiCNet, which can handle different types of combats with arbitrary numbers of AI agents for both sides. However, deep learning requires relatively high computational power and cannot work on all platforms. As a powerful technology, transfer learning is very effective for expanding the Machine Learning model to another domain and avoiding many expensive data-labeling efforts. The relationship between transfer learning and other related machine learning techniques is discussed in [22].

1.3. Research Motivation in This Work

The main research contents of Multi-Robot Confrontation include situation assessment and strategy selection. This study focuses on strategy selection. The scenes of Multi-Robot Confrontation are usually modeled as a semi-Markov decision-making process, and there are some strategy selection models that are based on classical reinforcement learning. However, these existing models often have the following four dilemmas. Firstly, most of the reinforcement learning models that are used for the strategy selection are tabular reinforcement learning methods, and the performance of these methods becomes worse with the expansion of the state-action space of the learning agent. Secondly, the learning rate of classical reinforcement learning methods is a fixed value. The process of turning the learning rate is quite empirical and often costs a lot of time. Thirdly, for the learning model of the Multi-Robot Confrontation based on RL, most models will use the ε-greedy strategy. This strategy

chooses each action with the same probability, whether good or bad. Finally, the existing works suffer from generalization dilemma, which causes a learning gap between a known environment and the unknown environment.

In this work, we investigate the Multi-Robot Confrontation in the semi-Markov decision-making process. An improved Q-learning algorithm with softmax function is introduced to train an agent in the semi-Markov decision-making process. A defined state-action space will not be expanded to avoid the curse of dimensionality due to the scale of the scenario. A reward function of this RL model helps agents balance losses of our agents and opponents.

1.4. Contributions in This Work

The main contributions of this paper include four parts. Firstly, in order to address the first dilemma mentioned above, this study introduces a Multi-agent RL algorithm with parameter sharing to train agents. A neural network (NN) with adaptive momentum is used as a function approximator to estimate the state-action value function. This NN model not only accelerates the efficiency of decision-making as an effective state representation but also guarantees the stability of neural network learning. Secondly, this study introduces an adaptive method with the fuzzy method to adjust the learning rate for the RL model to tackle the second dilemma. This method has been proved to be effective in improving the performance of micromanagement in the experiments. Thirdly, in order to address the third dilemma, we use a Boltzmann distribution to describe a statistical probability that decides the selection of each action. Fourthly, this study introduces a curriculum transfer learning method to address the generalization dilemma. This method improves the performance of micromanagement in different scenarios instead of learning from scratch. Meanwhile, we set the decay function according to Newton's cooling law. The decay function can reduce the influence of interference information in prior experience on a new micromanagement scenario for curriculum transfer learning. As far as the method itself is concerned, the developed method can be applied not only to the robot confrontation system but also to other application scenarios, such as Starcraft II.

1.5. Paper Structure

The remainder of the paper is organized as follows: Section 2 is devoted to the background for the proposed technique, such as reinforcement learning, softmax function. Section 3 presents a learning model for a single agent and this model uses an improved Q-learning with the fuzzy method. A neural network with adaptive momentum and the proposed multi-agent RL algorithm with parameter sharing for Multi-Robot Confrontation are introduced in Section 4. Section 5 introduces a curriculum transfer learning to address the generalization issue. This method transfers the prior learning experience to different scenarios without starting from scratch. Experiments are detailed in Section 6, to illustrate the performance of the proposed learning model. Conclusions are drawn in the last section.

2. Background

2.1. Reinforcement Learning

The goal of reinforcement learning is to obtain an optimal strategy π in which the agent selects action A under state S, which is given by $\pi(S) = A$. The architecture for reinforcement learning is shown in Figure 1. s_t is the current state of the agent and s_{t+1} is the next state. a_t is the current action that the agent takes. r_t is the current reward for the process of $s_t \rightarrow s_{t+1}$. γ is the discount factor. $V(s)$ is the state value function for the state s. $T^a_{ss'}$ is the transition probability from the state s to the next state s'. $R^a_{ss'}$ is the current reward that is obtained from the environment from the state s to the next state s'. α is the learning rate and its value range is $(0, 1)$.

The agent which is in state s_t selects action a_t until the final state is reached, and the cumulative reward obtained in this process is shown in Equation (1).

$$R(S_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = r_t + \gamma R(S_{t+1}) = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{1}$$

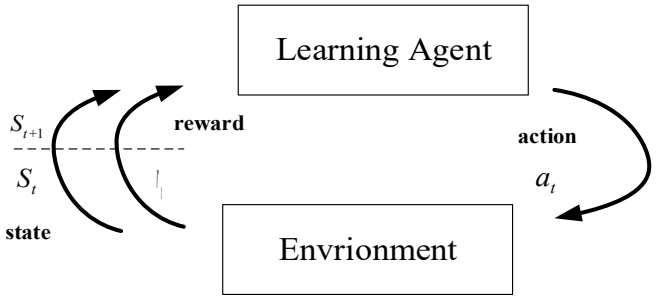


Figure 1. Reinforcement learning architecture.

For the reinforcement learning algorithm that uses the future P-step average rewards, the mathematical expression for the value function $V^{\pi}(s_t)$ is shown in Equation (2).

$$V^{\pi}(s_t) = \lim_{p \to \infty} \left(\frac{1}{p} \sum_{t=0}^{t=p} r_t \right) \tag{2}$$

When an agent takes a_t a strategy π, the value function represents the expectation of the cumulative reward obtained by the agent.

The value function $V^{\pi}(s)$ of an agent which is in the state s is given by:

$$\begin{aligned} V^{\pi}(s) &= \mathrm{E}^{\pi}\{R_t | s_t = s\} = \mathrm{E}^{\pi}\left\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \middle| s_t = s \right\} \\ &= \sum_a \pi(s,a) \sum_{s'} T^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s') \right] \end{aligned} \tag{3}$$

Q-Learning is a model-free reinforcement learning algorithm, and it is an off-policy reinforcement learning algorithm. The Q value $Q(s,a)$ is estimated via the Time Difference Method (TD Method) [21]:

$$Q_{t+1}(s_t, a) = (1 - \alpha) Q_t(s_t, a) + \alpha [r_t + \gamma \max Q_t(s_{t+1}, a')] \tag{4}$$

where the learning rate reflects the efficiency of an RL algorithm.

2.2. Softmax Function Based on Simulated Annealing

In order to control the randomness of action selection, a simulated annealing (SA) algorithm [23,24] is used to optimize softmax function. The softmax function is a method for balancing Exploration and exploitation [25] in the RL method, which chooses the action according to the average reward of each action, and the probability of the action a_t being chosen is higher if the average reward produced by the action is higher than the average reward produced by the other action.

The probability distribution of the action in the softmax algorithm is based on the Boltzmann distribution. The probability P_i of action a_i selected is given by,

$$\frac{e^{(a_i)}}{\sum_{k=0}^{K} e^{(a_k)}} \to P_i \tag{5}$$

where P_i represents the probability of choosing action a_i, and the total number of actions is K. The action selection policy based on Boltzmann distribution is used to ensure the randomness of the action selection, and the simulated annealing algorithm is added.

In this method, the probability of the action a_i being selected is given by,

$$\frac{e^{(Q(s_t,a_i)/T_t)}}{\sum_{k=1}^{K} e^{(Q(s_t,a_k)/T_t)}} \rightarrow P(a_i|s_t) \tag{6}$$

where T_t is the temperature parameter. The smaller the temperature parameter is $T_{\min}$, the bigger the probability of action with the high average reward being chosen is $T_{\max}$. The temperature value turned by the simulated annealing is given by,

$$\begin{cases} T_{t+1} = \eta(T_t - T_{\min}) + T_{\min} \\ T_0 = T_{\max} \end{cases} \tag{7}$$

where η is the annealing factor, and the value range is $0 \leq \eta \leq 1$.

3. An RL Model for a Single Agent

3.1. An Improved Q-Learning Method in Semi-Markov Decision Processes

For Markov dynamic systems, the stochastic control problems are modeled as Semi-Markov decision processes (SMDPs) [26,27]. The time cost for the RL system to transit from one state to the next state is defined as the sojourn time. The robot confrontation process is regarded as an SMDP, and the agent may take a serial of the same actions before transiting into the next state. The RL method can diminish the uncertainty resulting from modeling purposely, compared with the dynamic modeling method.

If our agent defeats the opponents with a better probability, the output of the agent is required to be more stable in the robot confrontation system for micromanagement. The classical Q-learning method solves the dilemma of exploration and exploitation using the ε-greedy algorithm, which makes the agent have a certain probability to explore new actions [28]. However, the probability of each action being selected is the same when the ε-greedy algorithm is used, so the action that can produce better rewards is not easy to choose. To tackle the problem of the greedy algorithm for micromanagement in the robot confrontation system, we design an improved Q-learning with the softmax function, which can make our agent explore more actions in the early stage of the learning process and exploit previous experience in the later stage of the learning process. For the learning process of SMDP, in each epoch, the next state is transited from the current state using the same actions after T learning cycles. $\{r_{t+i} \big| i = 0, 1, 2, \ldots, T-1\}$ is the real-time reward. Equations (8)–(10) gives the updating method of the state-action function.

$$\gamma Q_{t+1+i}(s_t, a_{t+1+i}) + r_{t+i} \rightarrow Q_{t+i}(s_t, a_{t+j}), 1 \leq i < T-1 \tag{8}$$

$$\begin{aligned} &\gamma \textstyle\sum_a \pi(a|s_{t+1}) Q(s_{t+1}, a) + r_{t+T-1} \rightarrow Q_{t+T-1}(s_t, a_{t+T-1}) \\ &\Rightarrow \gamma \textstyle\sum_a \frac{\exp\{Q(s_{t+1},a)/T_t\}}{\sum_a \exp\{Q(s_{t+1},a)/T_t\}} Q(s_{t+1}, a) + r_{t+T-1} \rightarrow Q_{t+T-1}(s_t, a_{t+T-1}) \end{aligned} \tag{9}$$

$$Q_t(s_t, a_t) + \alpha(\sum_{k=1}^{T-1} Q_{t+k} + r_t)/T \rightarrow Q_{t+T}(s_t, a_t) \tag{10}$$

where T_t is the temperature parameter.

The detailed step of this algorithm (SSAQ algorithm) is shown in Algorithm 1. The SSAQ algorithm is a way to solve the dilemma of exploration and exploitation, and this method can output more stable actions for micromanagement scenarios.

Algorithm 1: SSAQ algorithm for micromanagement

Definition
T_t: = Current temperature parameters
T_{t+1}: = Next temperature parameters
r_t: = Current reward
$T_{\min}$: = Minimum temperature parameter
$T_{\max}$: = Maximum temperature parameter
α: = Learning rate
γ: = Discount factor
K^t: = The number of actions in t time
$fun()$: = Updating state-action function in SMDP
Initialization
Initialize $s_t, a_t, s_{t+1}, a_{t+1}, r_t$
each value of Q matrix ← arbitrarily value;
Repeat (for each step)
Choose an initial state s_0
$t \leftarrow 0; i \leftarrow 0;$
Repeat (for each step of the episode)
$P(a_t|s_t) = \max_{ps:1\to K^t} P(a_k^{ps}|s_t) = \max_{ps:1\to K^t} \frac{\exp[Q(s_t,a_k^{ps})/T_t]}{\sum_{k=1}^{K^t} \exp[Q(s_t,a_k)/T_t]} \to a_t$
Observe s_{t+1}, after T learning cycles by the same actions a_t
Obtain reward $r_t, r_{t+1}, \ldots, r_{t+T-1}$
$P(a_{t+1}|s_{t+1}) = \max_{ps:1\to K^t} P(a_k^{ps}|s_{t+1}) = \max_{ps:1\to K^t} \frac{\exp[Q(s_{t+1},a_k^{ps})/T_t]}{\sum_{k=1}^{K^t} \exp[Q(s_{t+1},a_k)/T_t]} \to a_{t+1}$
$s_t \leftarrow s_{t+1}$
$t++$
$Q(s_t, a_t) \leftarrow f(Q(s_{t+1}, a_{t+1}), r_t, \ldots, r_{t+T-1}, \alpha, \gamma)$
until s is terminal.
Until Q matrix is convergence.

3.2. A Reinforcement Learning Method using a Fuzzy System

In order to reduce the learning cost, a learning method using a dynamic learning rate is a good solution [29]. For the reward function in the RL system, when the reward is positive after using the RL method, the influence of the positive feedback and a faster learning rate can be ensured by a high learning rate; on the contrary, a low learning rate can guide the RL method to a faster convergence. Therefore, there will be a relationship between the obtained reward and the value of the learning rate. Meanwhile, the reward is a fuzzy concept. For instance, a fixed value does not divide the "large positive number" and "large negative number". So, a fuzzy system is used to develop a dynamic learning rate to improve the performance of this learning system. The reward r is taken as the input of the fuzzy system and the learning rate α_F is taken as the output. We set the fuzzy description of r as "little negative number, the large negative number, zero, large positive number, little positive number", and their abbreviations are shown as "TN, LN, ZO, LP, TP". We set the fuzzy description of α_F as "little small, very small, medium, very large, little large", and their abbreviations are shown as "LS, VS, M, VL, LL". Figure 2 gives the corresponding fuzzy membership functions. Previous work has shown that the performance of the fuzzy system is affected by the shape of the membership functions [30]. In the fuzzy system, triangular membership function and trapezoidal membership function are simple and effective, so we choose these two membership functions.

In Figure 2, $\{r_i | i = 1, 2, \ldots, 6\}$ and $\left\{\alpha_i^F \middle| i = 1, 2, \ldots, 5\right\}$ are the division points that correspond to the input and output membership functions respectively. Five different input descriptions, "LP, TP, ZO, TN, LN" are $\left\{\mu_i^r(r) \middle| i = 1, \ldots, 5\right\}$, which correspond to each of these membership functions. Similarly, five fuzzy output descriptions, "VL, LL, M, LS, VS", correspond to the output membership

functions, expressed as $\left\{\mu_i^{\alpha_F}(\alpha)\middle|i=1,\ldots,5\right\}$. Figure 2 shows the curve for these membership functions. The degree of trust for the input shows a general mathematical symmetry. We take "LP" as an example, and its membership function is shown in Equation (11).

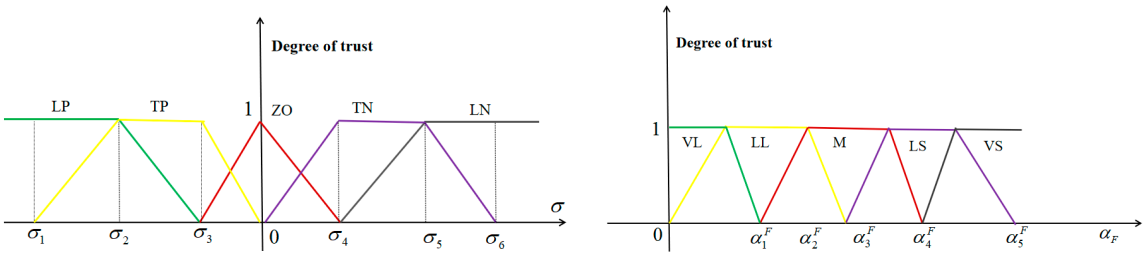


Figure 2. Curves for the Membership functions. (**a**) Reward/Input membership function. (**b**) Learning rate/Output membership function.

$$\mu_1^r(r)=\begin{cases}1, r<r_2\\(r_3-r)/(r_3-r_2), r_2<r<r_3\\0, r_3<r\end{cases} \tag{11}$$

The number of discrete points for the input domain is represented by N^r. We select five independent discrete points from the input domain, which are represented by $\mathbf{I}^r=\left\{I_j^r\middle|j=1,\ldots,N^r\right\}$. These five points have their own degrees of truth corresponding to five fuzzy input descriptions. Equation (12) gives the input degree for the truth discrete matrix $\mathbf{ID}^r=[ID_{ij}^r]_{5\times N^r}$.

$$ID_{ij}^r=\mu_i^r(I_j^r) \tag{12}$$

The number of discrete points for the output domain is M^{α_F}. We select five independent discrete points $\mathbf{O}^{\alpha_F}=\left\{O_j^{\alpha_F}\middle|j=1,\ldots,M^{\alpha_F}\right\}$ from the output domain. Equation (13) shows the corresponding output degree for the truth discrete matrix $\mathbf{OD}^{\alpha_F}=[OD_{ij}^{\alpha_F}]_{5\times M^{\alpha_F}}$.

$$OD_{ij}^{\alpha_F}=\mu_i^{\alpha_F}(O_j^{\alpha_F}) \tag{13}$$

Then, we design the fuzzy rules, as follows. "If reward is "LP" then the learning rate is "VL", If reward is "TP" then learning rate is "LL", if reward if "ZO" then learning rate is "M", if reward is "TN" then learning rate is "LS", if reward is "LN" then learning rate is "VS"". The fuzzy inference is represented by $\mathbf{RS}^{r\alpha_F}=[r_{ij}^{r\alpha_F}]_{N^r\times M^{\alpha_F}}$ and Equation (14) gives its mathematical expression.

$$r_{ij}^{r\alpha_F}=\bigvee_{k=1}^{5}(ID_{ki}^r\wedge OD_{kj}^r) \tag{14}$$

where "$\vee$" represents the operation of choosing the maximum value. "$\wedge$" represents the operation of choosing the minimum value. The operation of fuzzification transforms the reward r_0 into a fuzzy input vector $\mathbf{FI}^{r_0}=[FI_j^{r_0}]_{1\times N^r}$ if r_0 is measured. The degrees of the truth for r_0 corresponding to the five inputs are $\left\{\mu_i^{r_0}\middle|\mu_i^{r_0}=\mu_i^r(r_0);i=1,\ldots,5\right\}$. Equation (15) uses the weighted-average method to calculate $\mathbf{FI}^{r_0}$.

$$FI_j^{r_0}=\left\{\sum\nolimits_{i=1}^{5}(ID_{ij}^r\times\mu_i^{r_0})\right\}\Big/\sum\nolimits_{k=1}^{5}\mu_i^{r_0} \tag{15}$$

We use the "min-max compose" operation to calculate the fuzzy output vector $\mathbf{FO}^{r_0} = [FO_j^{r_0}]_{1\times M^{\alpha_F}}$ using Equations (14) and (15).

$$FO_j^{\alpha_{FS}} = \bigvee_{k=1}^{N^r} (rs_{kj} \wedge FI_k) \tag{16}$$

The de-fuzzifying operation uses a weighted average method to transform the fuzzy output into an output value and the weighted average method is shown in Equation (17).

$$\alpha_{FS} = \left\{\sum_{j=1}^{M^{\alpha}{}_F} (FO_j^{\alpha_{FS}} \times O_j^{\alpha})\right\} \Big/ \sum_{k=1}^{M^{\alpha}{}_F} FO_j^{\alpha_{FS}} \tag{17}$$

where the learning rate α_{FS} is the final result of the fuzzy system relative to r_0.

4. A Proposed Learning Model for Multi-Robot Confrontation

The classical Q-learning algorithm chooses actions by the look-up table method. However, with the increase in action space and state space, the look-up table method is obviously no longer suitable, and which results in low learning efficiency. In order to solve this problem, the RL method based on the neural network approximating the value function of Q learning is proposed [31]. In the random task scenarios, the state variable s_t is used as the input of the neural network and the Q value is used as the output of the neural network, which is also the estimation of the Q value of the neural network based on previous experience. This Q value is $Q_{current}$, and the action that was taken by the agent is the action corresponding to the maximum $Q_{current}$. After the agent takes action, the environment will give agent reward, and the state of the agent will be transferred to s_{t+1}. Similarly, the state s_{t+1} is input into the neural network, and the Q value of the corresponding state s_{t+1} is obtained, which is Q_{next}. Finally, Q_{next} and $Q_{current}$ is used to update the Q value using (4). The gradient descent method is used to update the gradient of the neural network. The loss function L_t of the neural network is shown in Equation (18).

$$\begin{aligned} L_t &= (Q\_current^* - Q\_current) \\ &= \{(1-\alpha)Q\_current + \alpha[r_t + \gamma \max Q\_next] - Q\_current\} \end{aligned} \tag{18}$$

where $Q\_current^*$ is the Q value after the state s_t is updated by (4), and Q_next is the Q value before the state s_t is updated.

4.1. Neural Network Model with Adaptive Momentum

Since the efficient action selection strategy should be considered for micromanagement scenarios, and our agent's experience usually has a limited subset of the large state space, it will be difficult to apply the conventional reinforcement learning to learn an optimal policy. To address this problem, a BP neural network is used to approximate the state-action values to improve the generalization of our RL model. An acceleration algorithm using adaptive momentum is considered to be used in the BP neural network to ensure the efficiency of action selection.

In addition to the output layer and the input layer, the input signal of any neuron j in a layer is represented by net_j. y_j is the output signal. y_j represents the output signal for the neuron i in the lower layer. Equation (19) gives the computing method for this output.

$$\begin{cases} net_j = \sum_i y_i \omega_{ji} \\ f(x) = \frac{\left(1-e^{-bx}\right)a}{1+e^{-bx}} \\ y_j = f(\theta_j + \sum_i y_i \omega_{ji}) \end{cases} \tag{19}$$

where the constant $a = 1.725$, $b = 0.566$, the threshold for a neuron k is represented by θ_k. In the output layer, y_k is the label output signal for the $k-$th neuron. net_k is the input signal. y_k and net_k are computed by Equation (20), if y_j is the output signal of the neuron $j-$th in the hidden layer next to the output layer.

$$\begin{cases} net_k = \sum_j y_j \omega_{kj} \\ y_k = f(\sum_j y_j \omega_{kj} + \theta_k) \end{cases} \tag{20}$$

In the $t-$th iteration for updating weight, an input value is $x_p(t)$. $O_{pk}(t)$, is the label output signal for the $k-$th neuron. $y_{pk}(t)$ is the actual output signal. Equation (21) gives the mean square error.

$$E_p(t) = \frac{\sum_k (O_{pk}(t) - y_{pk}(t))^2}{2} \tag{21}$$

The weight of this neural network is updated by the back-propagation method. The mean square error of this neural network is given by Equation (22) if there are M_n input values.

$$E_p(t) = \frac{\sum_p \sum_k (O_{pk}(t) - y_{pk}(t))^2}{2M_n} \tag{22}$$

The weights $\omega_{kj}(t)$ is updated according to the gradient direction of $E_p(t)$ to minimize the square error. The correction of $\omega_{kj}(t)$ is $\Delta_p \omega_{kj}(t)$, which is given by:

$$\begin{cases} \frac{\partial E_p(t)}{\partial \omega_{kj}(t)} = \left(\frac{\partial E_p(t)}{\partial net_k(t)}\right) \cdot \left(\frac{\partial net_u(t)}{\partial \omega_{kj}(t)}\right) \\ \frac{\partial net_u(t)}{\partial \omega_{kj}(t)} = \frac{\partial \sum_j y_{pj}(t) \omega_{kj}(t)}{\partial \omega_{kj}(t)} \\ \Delta_p \omega_{kj}(t) = \beta \Delta_p \omega_{kj}(t) - \lambda \frac{\partial E_p(t)}{\partial \omega_{kj}(t+1)} \end{cases} \tag{23}$$

where the learning rate is λ and the momentum constant is β that is 0.95.

If the learning rate of the neural network is too large, the weight correction $\Delta_p \omega_{kj}(t)$ will be too large, so the stability of the learning process will be affected. Therefore, we use an adaptive learning rate. $\alpha(p)$ represents the adaptive learning rate, and mean square error is $E_p(t)$. The adaptive learning rate satisfies Equation (24).

$$\alpha(p) = 1 - \exp(-E_p(t)) \tag{24}$$

If the mean square error increases, the learning rate increases, and the convergence rate of the neural network accelerate; on the contrary, the neural network tends to be stable.

We set $\delta_{pk}(t) = -\frac{\partial E_p(t)}{\partial net_k(t)}$, and (25) can be obtained.

$$\begin{cases} f(\theta_k(t) + net_k(t)) \\ \quad = \frac{\partial}{\partial net_k(t)} \left(\frac{2a}{1+\exp(-F(\theta_k(t)+net_k(t)))} \right) = (1 - y_{pk}(t)) y_{pk}(t) \\ \delta_{pk}(t) = -\left(\frac{\partial E_p(t)}{\partial net_k(t)}\right) \cdot \left(\frac{\partial y_{pk}(t)}{\partial net_k(t)}\right) = f(\theta_k(t) + net_k(t)) \cdot (O_{pk}(t) - y_{pk}(t)) \end{cases} \tag{25}$$

Therefore, Equation (26) can be obtained.

$$\begin{cases} \Delta_p \omega_{kj}(t) = \beta \Delta_p \omega_{kj}(t) - \delta_{pk}(t) y_{pk}(t) \\ \delta_{pk}(t) = (O_{pk}(t) - y_{pk}(t)) \frac{\partial}{\partial net_k(t)} \left(\frac{2a}{1+\exp(-F(\theta_k(t)+net_k(t)))} \right) \end{cases} \tag{26}$$

The updating for the weights is given by Equation (27).

$$\begin{cases} \delta_{pj} = \frac{\partial E_p}{\partial net_j} = \left(\frac{\partial E_p}{\partial y_{pj}}\right) \cdot \left(\frac{\partial y_{pj}}{\partial net_j}\right) = y_{pj}(y_{pj}-1)\left(\frac{\partial E_p}{\partial y_{pj}}\right) \\ \Delta_p \omega_{ji} = -\lambda \left(\frac{\partial E_p}{\partial net_j}\right) \cdot \left(\frac{\partial net_j}{\partial \omega_{ji}}\right) = \lambda y_{pj} \delta_{pj} \end{cases} \tag{27}$$

where,

$$\begin{aligned} -\frac{\partial E_p}{\partial y_{pj}} &= \sum_k \omega_{kj} \delta_{pk} = -\sum_k \left(\frac{\partial E_p}{\partial net_k}\right) \cdot \left(\frac{\partial net_k}{\partial y_{pj}}\right) \\ &= \sum_k \left(\frac{\partial \sum_m y_{pm} \omega_{km}}{\partial y_{pj}}\right) y_{pj}(1-y_{pj}) \left(\frac{\partial E_p}{\partial y_{pj}}\right) \end{aligned} \tag{28}$$

Then,

$$\delta_{pj} = -y_{pj}(y_{pj}-1) \cdot \left(\sum \delta_{pk} \omega_{kj}\right) \tag{29}$$

Finally, $\Delta_p \omega_{ji} = \lambda y_{pi} \delta_{pj}$ is the correction of the weights for the hidden layer.

4.2. Multi-Agent RL Algorithm Based on Decision-Making Neural Network with Parameter Sharing

In this paper, an accelerated BP neural network with adaptive momentum is used as an approximator of the state-action value function. In this study, we extend the SSAQ algorithm to multi-agent by sharing the parameters of the neural network. Agents behave differently because each one receives different states and actions in the environment. Therefore, it is feasible for multi-agent to use the same neural network via parameter sharing. The input of the neural network is the agent state set in this paper, and the output is the state-action value function of the corresponding state. In order to ensure the continuous actions of the agent, a softmax layer is added after the output layer of the neural network, and the softmax layer uses the softmax function to select the action for the agent. Meanwhile, the simulated annealing algorithm is added to the softmax function to adjust the temperature parameters. This kind of neural network is called a decision-making neural network (DMNN) in this paper. The neural network is used as the approximator for the SSAQ algorithm mentioned above, and the learning model for our agents is shown in Figure 3.

As shown in Figure 4, the decision-making neural network includes an input layer, a multilayer hidden layer, an output layer, and a softmax layer. The state s_t of the agent is inputted to the input layer of the decision-making neural network, and the Q value corresponding to the state s_t is outputted to the output layer of the decision-making neural network, which is recorded as

$$\left\{Q(s_t, a_t; \theta) \middle| i = 1, 2, \ldots, K^t\right\}.$$

where θ is the weight vector of the decision-making neural network. The Q values of all actions are entered into the softmax layer with a non-linear transformation function which is shown in Equation (30), and the action a_t is selected and outputted using the Max operation as shown in Equation (31).

$$\phi(x) = \exp(x)/T_i \tag{30}$$

$$\max\left\{\phi(x_1)/\sum_{i=1}^m \phi(x_i), \phi(x_2)/\sum_{i=1}^m \phi(x_i), \ldots, \phi(x_m)/\sum_{i=1}^m \phi(x_i)\right\} \tag{31}$$

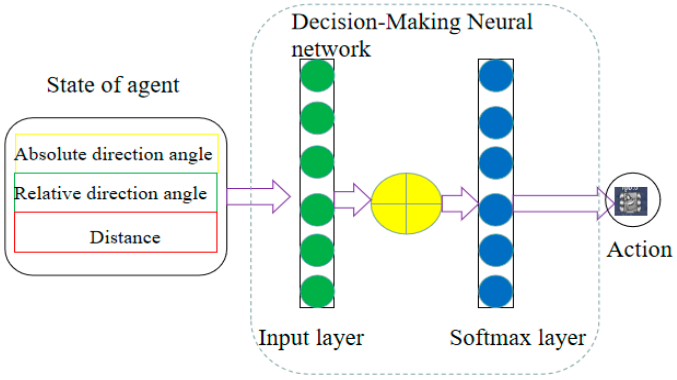


Figure 3. The learning model of agent architecture. The decision-making neural network is used as the approximator for the state-action function.

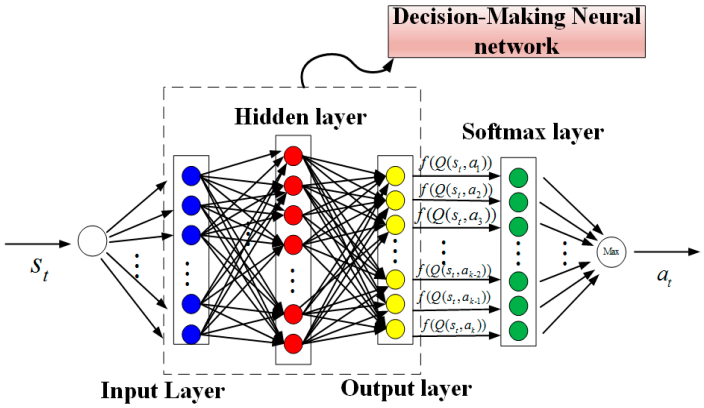


Figure 4. Decision-making neural network architecture.

The agent takes action a_t and the environment gives instant rewards $R(t)$. The RL based on state-action function stores the state-action function in the form of a table, but this method cannot solve the large-scale continuous state-space problem, and the powerful generalization ability of the neural network is used to approximate the value function $Q(s_t, a_t; \theta)$ for RL, which solves the problem of high dimensional continuous state space. The proposed method has a good generalization and can be used in other combat games, such as Starcraft II [18].

To update the weights of the neural network efficiently, the TD error of the RL method is used for the Loss function for the decision-making neural network. The TD error is shown in Equation (32). The back propagation algorithm is used to update the weights of neural networks. The Multi-agent SSAQ algorithm with network parameter sharing is given by Algorithm 2.

$$\begin{cases} Q_{next}(s_t, a_t, \theta_t) = Q(s_t, a_t, \theta_t)_{cur} + \\ \alpha[R(t) + \gamma Q(s_{t+1}, a_{t+1}, \theta_t) - Q(s_t, a_t, \theta_t)_{cur}] \\ \phi_t = Q_{next}(s_t, a_t, \theta_t) - Q(s_t, a_t, \theta_t)_{cur} \end{cases} \tag{32}$$

Algorithm 2: Multi-agent SSAQ algorithm

Definition
T_t: = Current temperature parameters
T_{t+1}: = Next temperature parameters
$T_{\min}$: = Minimum temperature parameter
$T_{\max}$: = Maximum temperature parameter
α_t: = Adaptive learning rate of the neural network
γ: = Discount factor
K^t: = The number of actions in t time
$fun()$: = Updating state-action function in SMDP
Initialization
Initialize $s_t, a_t, s_{t+1}, a_{t+1}, r_t$
Repeat (for each step)
Choose an initial state s_0
$t \leftarrow 0; i \leftarrow 0;$
Repeat (for each step of the episode)
$P(a_t|s_t) = \max_{ps:1\to K^t} P(a_k^{ps}|s_t) = \max_{ps:1\to K^t} \frac{\exp[Q(s_t,a_k^{ps})/T_t]}{\sum_{k=1}^{K^t} \exp[Q(s_t,a_k)/T_t]} \to a_t$
Observe s_{t+1}, after T learning cycles by the same actions a_t
Obtain reward $r_t, r_{t+1}, \ldots, r_{t+T-1}$
$P(a_{t+1}|s_{t+1}) = \max_{ps:1\to K^t} P(a_k^{ps}|s_{t+1}) = \max_{ps:1\to K^t} \frac{\exp[Q(s_{t+1},a_k^{ps})/T_t]}{\sum_{k=1}^{K^t} \exp[Q(s_{t+1},a_k)/T_t]} \to a_{t+1}$
Update TD error and weights:
$\phi_t \leftarrow \alpha[f(Q(s_{t+1}, a_{t+1}), r_t, \ldots, r_{t+T-1}, \alpha, \gamma) - Q(s_t, a_t, \theta_t)]$
$E_t \leftarrow \frac{1}{2}\phi_t^2; \alpha_t \leftarrow 1 - \exp(-E_t)$
$\theta_{t+1} \leftarrow \theta_t + \alpha_t \phi_t$
$t++$
until s is terminal.

5. Curriculum Transfer Learning

Curriculum Transfer Learning for Different Micromanagement Scenarios

It will cost a lot of time if the agent starts learning from scratch in a new environment. Many researchers focus on improving the learning performance by exploiting domain knowledge between some related tasks. The prior learning experience is exploited from the source task to the target task by the transfer learning to accelerate the learning rate. Therefore, we use the transfer learning method to take the well-trained model of source task as the prior experience to build the learning model to the target task.

In the curriculum transfer learning for micromanagement scenarios in the confrontation decision-making system, the mapping $\rho : \pi^*_{pasttime} \to \pi^*_{currenttime}$ represents the process that transfers the learning policy of the source task to the learning policy of the target task. In this paper, the state space and action space remain unchanged, as shown in Equation (33).

$$\begin{cases} \rho A : A_{last\ time} \to A_{current\ time} \\ \rho S : S_{last\ time} \to S_{current\ time} \end{cases} \tag{33}$$

Many interference information exists in the learning process. Therefore, we set up a decay function using the Newton law of cooling. The decay function enables agents to exploit the domain knowledge with a decreasing probability. A steady-state is achieved eventually. The threshold is ε. Equation (34) shows the mathematical relationship between the threshold ε and time t. The agent uses the domain

knowledge from the source task if the random number $\varepsilon < random$. Otherwise, the agent uses the conventional maximum Q value strategy to select an action.

$$\varepsilon(t) = \varepsilon(t_0) \cdot \exp\{-pt + pt_0\} \tag{34}$$

where the decay coefficient is p and the initial time is t_0. The probability of using the prior experience from the source task gradually decreases until a stable value is achieved.

If the target task is too difficult compared with the source task, an intermediate task is usually set up in curriculum transfer learning, and the agent can gain more experience by the learning model for the intermediate task. The curriculum transfer learning with an intermediate task and decay function for micromanagement scenarios is shown in Figure 5.

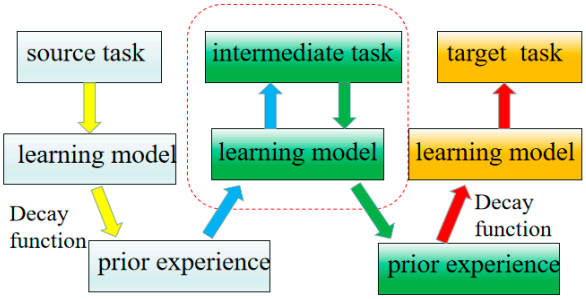


Figure 5. Curriculum transfer learning with an intermediate task and decay function.

The integral framework for the proposed learning model for micromanagement is shown in Figure 6. The proposed method has three parts: the decision-making neural network, the loss function that uses the TD error for the neural network and a fuzzy method. The state s_t of the agent is input into the neural network, and the neural network outputs the action a_t. So is the next state s_{t+1}. The reward $R(t)$ is obtained. The TD error is calculated as a loss function and a fuzzy method is used to adjust the learning rate of the RL method.

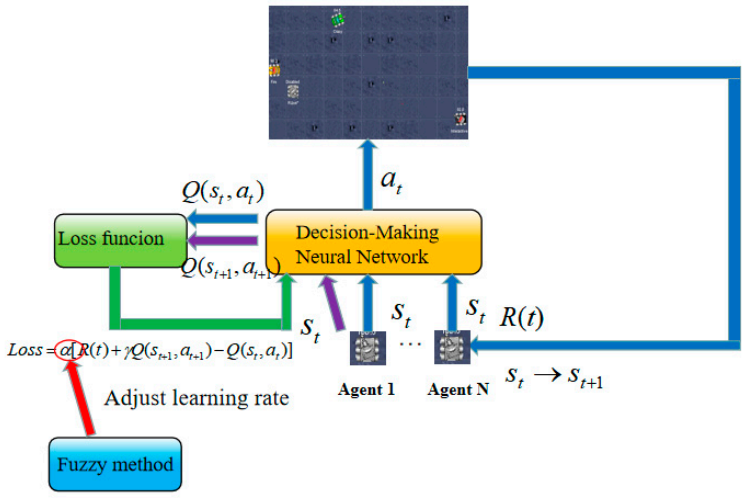


Figure 6. Framework for the proposed method for micromanagement.

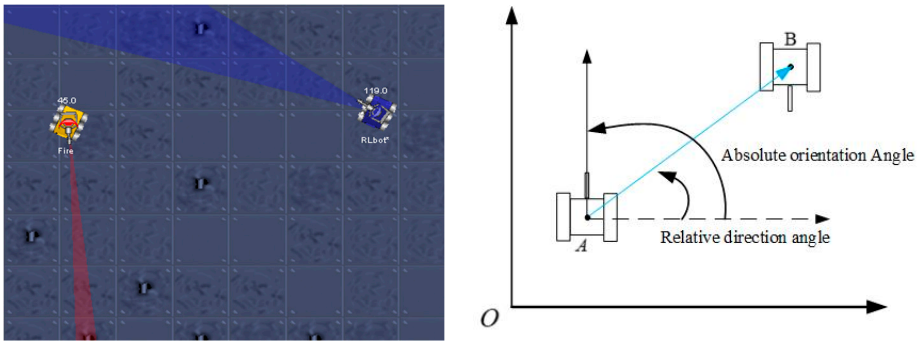
6. Experiment and Analysis

Generally, in the robotic systems that are based on reinforcement learning, higher learning rates enable robots to utilize the previous learning experience. The larger discount rate makes learning agents think more about long-term returns in the future. For the exploration and exploitation, the ε-greedy algorithm with a larger threshold allows the learning agent to more utilize prior experience. In this

work, the initial value for the threshold of the decay function is the same as that of the classical ε-greedy algorithm. In these experiments, the values for the learning rate, the discount factor, the Exploring rate, the Annealing factor, the Maximum temperature parameter, and the Minimum temperature parameter are given manually and empirically.

6.1. RL Model for a Confrontation Decision-Making System

Robocode [32,33] is an open-source platform, where the goal is to develop a robot to battle against other robots. The platform is shown in Figure 7. In this paper, the experiments are conducted in this platform and we consider several scenarios with the different enemies to test the generalization of the proposed method.



(**a**) (**b**)

Figure 7. Robot confrontation platform. (**a**) a platform for the tank battle system, (**b**) diagram of absolute and relative angles.

Effective state-action space definition of the RL model is still an open problem with no universal solution. An RL model for robot confrontation with inputs from the game engine is constructed, which ensures the size of the state-action space remaining unchanged to avoid the curse of dimensionality.

State-space: The combination of the relative direction angle, the absolute orientation angle, and the distance between the robots form the state space. The absolute direction angle and the relative direction angle are discretized into four kinds of states and the range of absolute direction angle is $0 \sim 360°$. We divide the distance between the robots into 20 discrete parts.

Action space: Movement and rotation are two movements for the robot in this platform. At each time step, each robot can move to arbitrary directions with arbitrary distances in the ground. Similar to other types of combat games, our robot can choose to attack their opponents with bullets of different energies. Forward, backward, clockwise rotation and anticlockwise rotation four kinds of different movements form the action space.

Reward function: If a robot fires bullets hit the enemy or is hit by bullets, the health point of this robot will change. We propose a reward function to help agent balance losses of our and opponents, as shown in:

$$\begin{aligned} &reward(t) = \\ &((E_t^m - E_{t-1}^m) - (E_t^e - E_{t-1}^e)) * \frac{\left|E_{t-1}^e - E_t^e\right|}{\left|E_t^m - E_{t-1}^m\right|} \end{aligned} \tag{35}$$

In this reward function, E_t^m represents the health point of our agent at t time, E_{t-1}^m represents the health point of our agent at the t−1 time, E_t^e represents the health point of the opponent at the t time and E_{t-1}^e represents the health point of the opponent at the t−1 time. If we lose fewer health points than the opponent loses, we will get a positive reward. The ratio of absolute health point changes between the two sides will encourage our agent to hurt the opponent more in battle. According to the experimental results, the RL model is effective at controlling our agent in the micromanagement scenarios.

6.2. Proposed RL Algorithm Test

In order to verify the effectiveness of the proposed SSAQ algorithm, a comparative experiment of robot migration is designed. The map for robot migration satisfies the binary tree structure. As shown in Figure 8, there are N layers on the map, and the number of endpoints is $2^N - 1$.

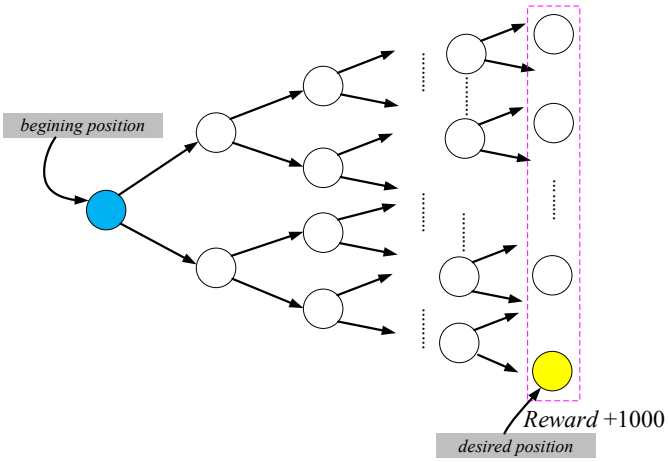


Figure 8. The experiment for robot migration.

The robot can get the reward corresponding to each endpoint by starting from the beginning and repeating the branch to the bottom endpoint. A total of $2^N - 1$ actions can be selected by the robot. The corresponding Q values of each endpoint are expressed as $Q_1 \sim Q_{2^N-1}$. When the robot reaches the endpoint $2^N - 1$, it gets a reward of +1000, and it does not get a reward reaching the rest of the endpoints. Q values can be obtained through experiments. This paper uses the SSAQ method to compare with the ε-greedy policy (Greedy-policy) and ε-greedy policy using the decay threshold (Dgreedy-policy) [34]. The settings of parameters for this experiment are listed in Table 1.

Table 1. Experimental parameters.

Parameter	Value
Learning rate α	0.3
Discount rate γ	0.9
Exploring rate ε	0.9
Annealing factor η	0.9
Maximum temperature parameter $T_{\max}$	0.1
Minimum temperature parameter $T_{\min}$	0.01
Layers of the map N	10

Through the observation during the experiments, it shows that Q_3, Q_7 and Q_{15} will change in the experiment, and we record these Q values. Therefore, the changes in three kinds of Q values of Q_3, Q_7 and Q_{15} are used in the experiment to compare three different strategies. As shown in Figure 9, the convergence time of the proposed method (SSAQ algorithm) is 214, 168 and 122 for Q_3, Q_7 and Q_{15}, respectively. The convergence time of the proposed method is the shortest and the convergence rate of it is the fastest compared with the other two methods. In addition, the Q-value curve of the proposed strategy is relatively smooth, which also proves that the proposed strategy is more stable and can achieve the target state quickly while ensuring the stability of learning performance.

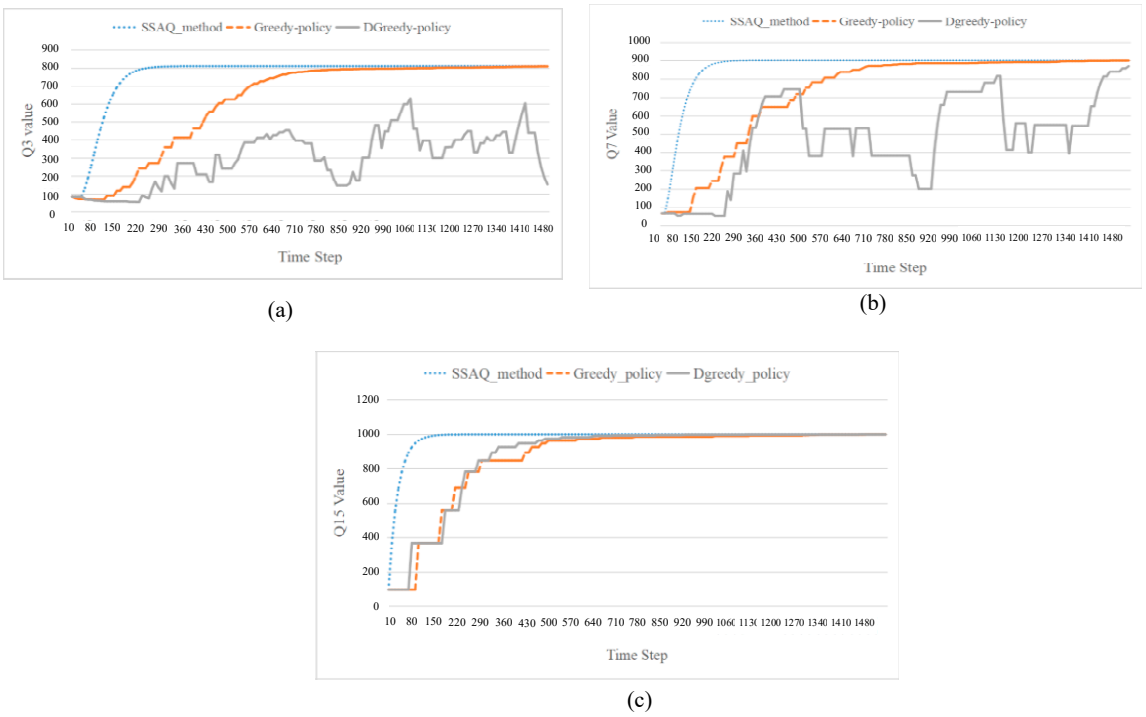


Figure 9. The comparison curve of Q value: (**a**) The change curve of Q_3; (**b**) The change curve of Q_7; (**c**) The change curve of Q_{15}.

6.3. Effect Test for Multi-Agent RL Based on DMNN and Fuzzy Method

The convergence and stability of the neural network are important factors affecting the performance of micromanagement for an agent. Hence, this paper proposes a decision-making neural network with adaptive momentum. In order to verify the effectiveness of the proposed Multi-agent RL algorithm based on DMNN, a team of agents trained by the neural network with adaptive momentum (NN with AWM) and another team of tank agents trained by the neural network without adaptive momentum (NN without AWM) are used to fight the team of tank agents trained by TD method separately. Each team has four tank agents. The settings of parameters for this experiment are listed in Table 2.

Table 2. Experimental parameters.

Parameter	Value
Learning rate of RL α	0.1
Discount rate γ	0.9
Exploring rate ε	0.95
Annealing factor η	0.9
Maximum temperature parameter $T_{\max}$	0.1
Minimum temperature parameter $T_{\min}$	0.01
Learning rate of Neural network α	0.9
Momentum constant β	0.95

We compare "NN with AWM" and "NN without AWM" from three different perspectives: total score ratio, defensive score and fluctuation value of score ratio. The number of rounds is 500, and the score is recorded every 10 rounds. Every 10 rounds are called a session. Figures 10–12 represent the resulting curve of this experiment, in which the horizontal axis represents the number of sessions and the vertical axis represents the score ratio or fluctuation value or defensive score. This paper calculates the first-order difference of the total score ratio, that is, $Score\ ratio_{t+1} - Score\ ratio_t$, and obtains the fluctuation value curve of the total score ratio, as shown in Figure 11. From the experimental results, it can be seen that these score ratios of the tank agents trained by the NN with the AWM method

are obviously higher than the NN without AWM method, and the fluctuation value is relatively small. Figure 12 shows the curve of the defensive score. Similar to Figure 10, the defensive score of the NN with the AWM method is more stable and higher than the NN without the AWM method. The experimental results show that the proposed Multi-agent RL algorithm can not only make the tank agent get higher scores in combat, but also get more stable scores. Hence, we can see that the proposed method has outstanding performances for micromanagement.

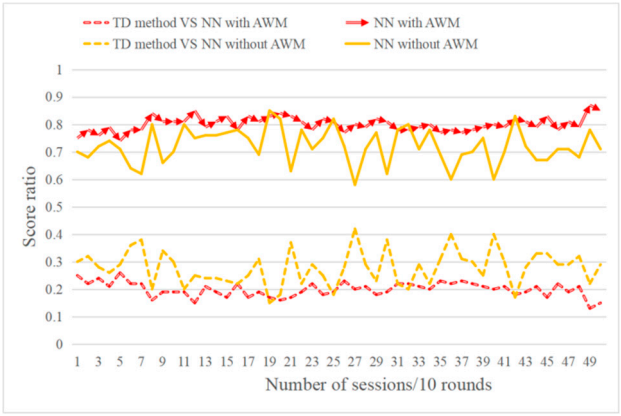


Figure 10. The comparison curve of the total score ratio.

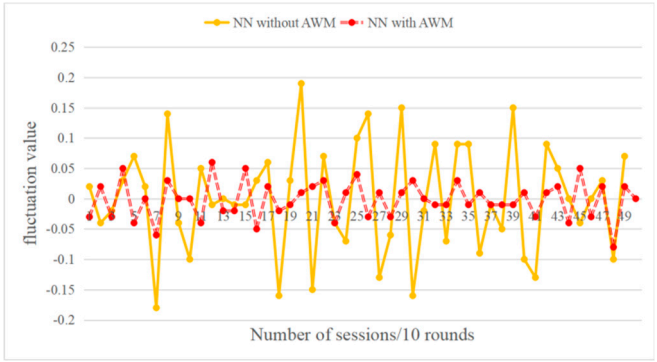
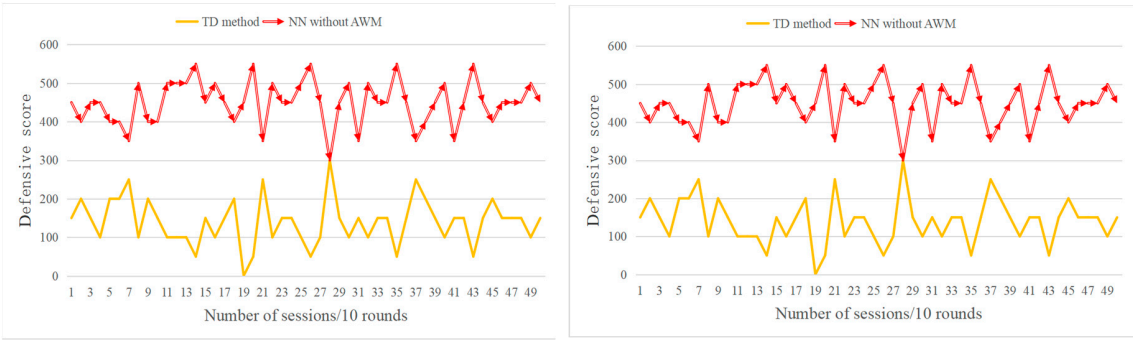


Figure 11. The curve of the fluctuation value of the total score ratio.



(**a**) (**b**)

Figure 12. The curve of the Defensive score. (**a**) The defensive score of neural network (NN) without adaptive momentum (AWM); (**b**) The defensive score of NN with AWM.

A fuzzy method is used to tuning the learning rate for the RL model. The input domain of the fuzzy method is set as $\{r_i|i = 1, 2, \ldots, 7\} = \{-30, -20, -10, 0, 10, 20, 30\}$, and the output domain of the fuzzy method is set as $\{\alpha_i|i = 1, 2, \ldots, 7\} = \{0.95, 0.80, 0.65, 0.50, 0.35, 0.20, 0.05\}$. The number of discrete points is 7. According to the fuzzy method used in this paper, the results of the fuzzy system

are shown in Figure 13. As shown in Figure 13, the fuzzy method converts the linear input into the smooth output, which is appropriate for micromanagement in the Multi-Robot Confrontation system. Then, the experimental results of the fuzzy-inspired learning model (FLM), learning model (LM) with learning rate being 0.2 (LM-0.2) [12] and learning model with learning rate being 0.95 (LM-0.95) are tested in the Robocode platform, as shown in Figure 14.

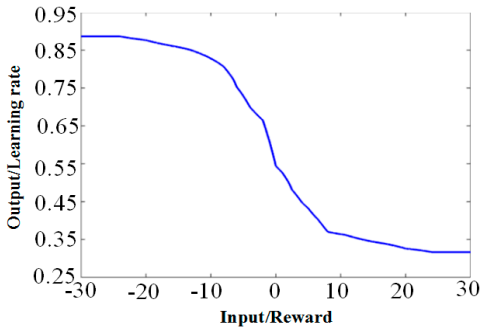


Figure 13. Results of the fuzzy method.

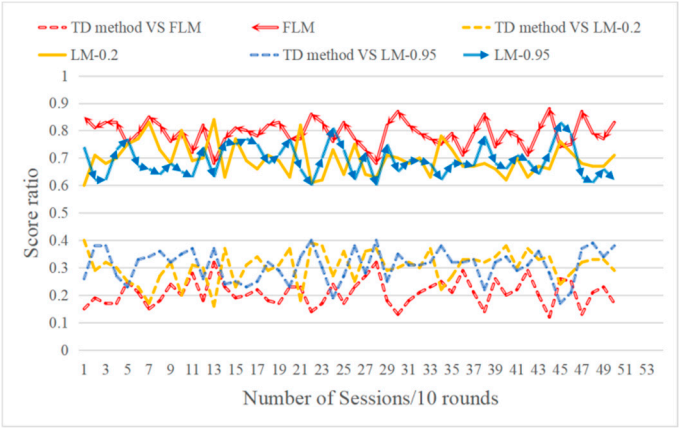


Figure 14. Comparison of learning results for different learning rates.

The three methods of FLM, LM-0.2, and LM-0.95 were respectively fought with the TD method. The solid line represents the score of three methods respectively, while the dotted lines represent the scores of TD methods fighting against three methods. From Figure 14, in FLM, the score ratio remains at around 0.8. In LM-0.2 and LM-0.95, the score ratio remains at around 0.7. According to the above results, it can be obtained that the fuzzy method can get a higher score in combat than the method of the fixed learning rate for micromanagement.

6.4. Effect Test for Curriculum Transfer Learning (TF)

Six tank agents trained by the TD method are formed, which is called "TD_team", and the multiple robots in formation will not attack each other. Then, the learning model proposed in this paper is used to fight the TD_team. Compared to the above experiments, our tank agents will face six opponents at the same time, as shown in Figure 15. In Figure 15, The agent in the red box is the enemy team, while the agent in the yellow box is our team.

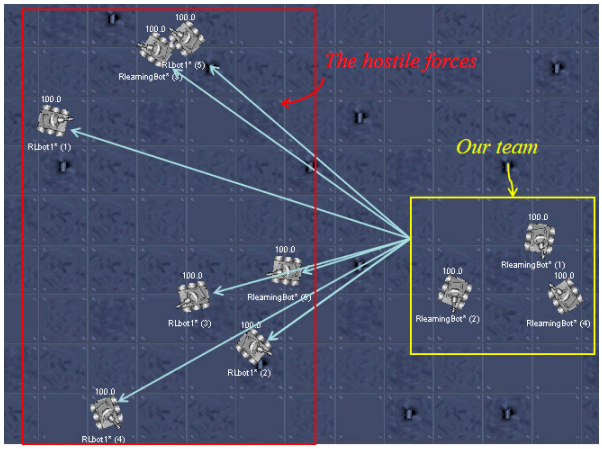


Figure 15. Simulation experiments for the TF method.

In order to prove the efficiency and practicality of the learning model with curriculum transfer learning (LM with TF), it is compared with the learning model without curriculum transfer learning (LM without TF) for 500 rounds.

The tank agent using curriculum transfer learning will use the prior experience gained in the above experiments. From Figure 16, transfer learning accelerates the learning speed of tank agents in the new micromanagement scenario and achieves higher scores than the method without transfer learning. From Figure 17, it is obvious that the defensive score of the LM with the TF method is higher, which means that the agent can defend the opponent's attack well when attacking opponents. Ten tank agents trained by the TD method were formed, and then the LM with the TF method and the LM without TF method were used to fight against them respectively. The task of fighting TD_team is regarded as the intermediate task of this task because it is more difficult than the original task. From Figure 18, the score ratio of the LM with the TF method is still higher than that of the LM without the TF method, which means the learning model with curriculum transfer learning has a strong and stable property for micromanagement scenarios in confrontation decision-making system.

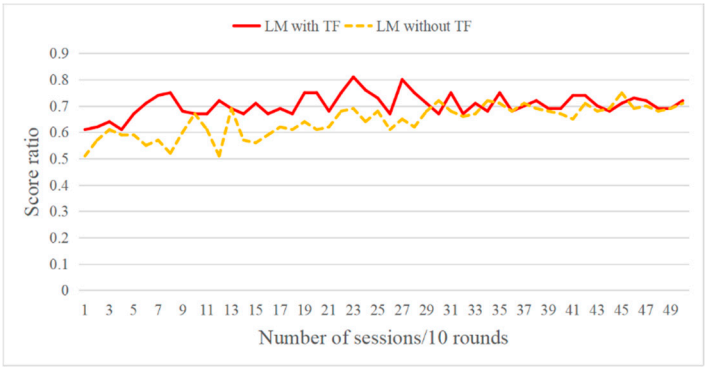
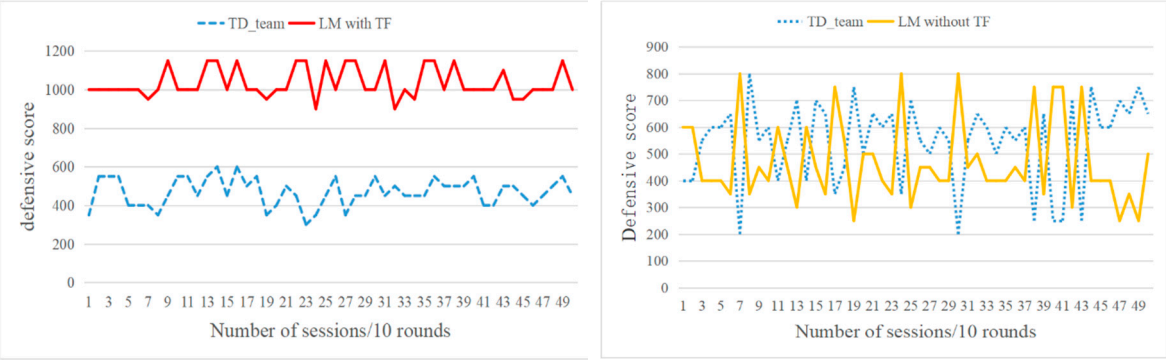


Figure 16. Comparisons of learning models in score ratio. One model uses transfer learning, the other does not use transfer learning.



(**a**) (**b**)

Figure 17. Comparisons of learning models in the defensive score. (**a**) The defensive score for LM with TF and TD_team; (**b**) Defensive score for LM without TF and TD_team.

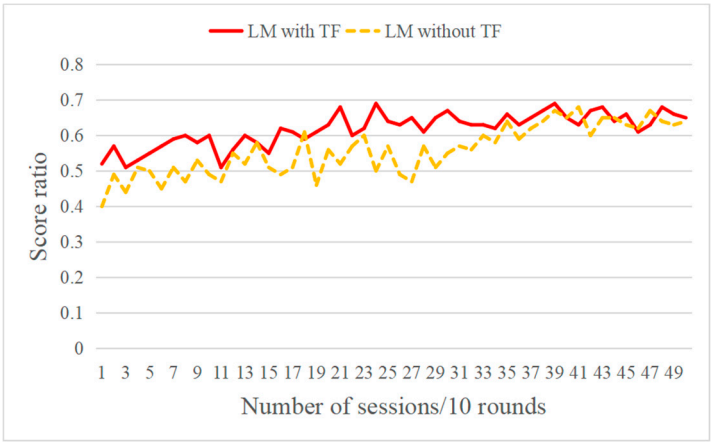


Figure 18. Comparisons of learning models in score ratio.

7. Conclusions

In this paper, the confrontation decision-making for micromanagement scenarios in SMDP is studied. This paper makes several contributions, including an improved Q-learning in SMDP (SSAQ algorithm) for confrontation decision-making, the fuzzy method for adjusting learning rate of the RL method, Multi-agent RL algorithm with parameter sharing, accelerated neural network for the representation of state (DMNN) and a curriculum transfer learning method with decay function. The RL model designed ensures the size of the state-action space remaining unchanged to avoid the curse of dimensionality and the reward function to help agent balance losses of our agents and opponents. The decision-making neural network uses adaptive momentum, which is used as an approximator to estimate the state-action function. The accelerated algorithm using adaptive momentum allows the decision-making neural network to react quickly in micromanagement scenarios. Finally, the curriculum transfer learning method with the decay function extends our model to other different scenarios. The proposed transfer learning method can still achieve better experimental results in more complex confrontation scenarios. The results of experiments demonstrate proposed methods can achieve an excellent and stable control for micromanagement in robot confrontation system. In the future, we will extend this model to the game scenario of Starcraft II. In addition, the behavior decomposition methods [35–38] will be studied to achieve more advanced Multi-agent confrontation strategies.

Author Contributions: C.H. and M.X. conceived the idea of the paper. C.H. and M.X. designed and performed the experiments; C.H. and M.X. analyzed the data; C.H. contributed reagents/materials/analysis tools; C.H. wrote and revised the paper.

Funding: This work is supported in part by the Natural Science Foundation of Hubei Province under Grant 2013CFC026, the Shaanxi Province Key Research and Development Program of China under Grant 2018GY-187.

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

References

1. Weron, R. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *Int. J. Forecast.* **2014**, *4*, 1030–1081. [CrossRef]
2. Ferreira, L.N.; Toledo, C.; Tanager, A. Generator of Feasible and Engaging Levels for Angry Birds. *IEEE Trans. Games* **2017**, *10*, 304–316. [CrossRef]
3. Hiller, J.; Reindl, L.M. A computer simulation platform for the estimation of measurement uncertainties in dimensional X-ray computed tomography. *Measurement* **2012**, *45*, 2166–2182. [CrossRef]
4. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
5. Moravik, M.; Schmid, M.; Burch, N.; Lisy, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; Bowling, M. Deepstack: Expert level artificial intelligence in heads-up no-limit poker. *Science* **2017**, *356*, 508–513. [CrossRef]
6. Yao, W.; Lu, H.; Zeng, Z.; Xiao, J.; Zheng, Z. Distributed Static and Dynamic Circumnavigation Control with Arbitrary Spacings for a Heterogeneous Multi-robot System. *J. Intell. Robot. Syst.* **2018**, *4*, 1–23. [CrossRef]
7. Ontanon, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; Preuss, M.A. Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Trans. Comput. Intell. AI Games* **2013**, *5*, 293–311. [CrossRef]
8. Synnaeve, G.; Bessiere, P. Multiscale Bayesian Modeling for RTS Games: An Application to StarCraft AI. *IEEE Trans. Comput. Intell. AI Games* **2016**, *8*, 338–350. [CrossRef]
9. Thrun, S.; Littman, M.L. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Netw.* **2005**, *16*, 285–286.
10. Shi, H.; Li, X.; Hwang, K.S.; Pan, W.; Xu, G. Decoupled Visual Servoing With Fuzzy Q-Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 241–252. [CrossRef]
11. Shi, H.; Lin, Z.; Zhang, S.; Li, X.; Hwang, K.S. An adaptive Decision-making Method with Fuzzy Bayesian Reinforcement Learning for Robot Soccer. *Inform. Sci.* **2018**, *436*, 268–281. [CrossRef]
12. Xu, M.; Shi, H.; Wang, Y. Play games using Reinforcement Learning and Artificial Neural Networks with Experience Replay. In Proceedings of the 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 6–8 June 2018; pp. 855–859.
13. Cincotti, S.; Gallo, G.; Ponta, L.; Raberto, M. Modeling and forecasting of electricity spot-prices: Computational intelligence vs. classical econometrics. *AI Commun.* **2014**, *3*, 301–314.
14. Geramifard, A.; Dann, C.; Klein, R.H.; Dabney, W.; How, J.P. RLPy. A value-function-based reinforcement learning framework for education and research. *J. Mach. Learn. Res.* **2015**, *16*, 1573–1578.
15. Modares, H.; Lewis, F.L.; Jiang, Z.P. Optimal Output-Feedback Control of Unknown Continuous-Time Linear Systems Using Off-policy Reinforcement Learning. *IEEE Trans. Cybern.* **2016**, *46*, 2401–2410. [CrossRef] [PubMed]
16. Konda, V. Actor-critic algorithms. *Siam J. Control Optim.* **2003**, *42*, 1143–1166. [CrossRef]
17. Patel, P.G.; Carver, N.; Rahimi, S. Tuning computer gaming agents using Q-learning. In Proceedings of the Computer Science and Information Systems (FedCSIS), Szczecin, Poland, 18–21 September 2011; pp. 581–588.
18. Shao, K.; Zhu, Y.; Zhao, D. StarCraft Micromanagement with Reinforcement Learning and Curriculum Transfer Learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *3*, 73–84. [CrossRef]
19. Xu, D.; Shao, H.; Zhang, H.A. New adaptive momentum algorithm for split-complex recurrent neural networks. *Neurocomputing* **2012**, *93*, 133–136. [CrossRef]
20. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
21. Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; Wang, J. Multi-agent bidirectionally-coordinated nets for learning to play StarCraft combat games. *arXiv* **2017**, arXiv:1703.10069.
22. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]

23. Gil, P.; Rez, M.; Mez, M.; Gómez, A.F.S. Building a reputation-based bootstrapping mechanism for newcomers in collaborative alert systems. *J. Comput. Syst. Sci.* **2014**, *80*, 571–590.
24. Bertsimas, D.; Tsitsiklis, J. Simulated Annealing. *Stat. Sci.* **1993**, *8*, 10–15. [CrossRef]
25. Shi, H.; Yang, S.; Hwang, K.; Chen, J.; Hu, M.; Zhang, H. A Sample Aggregation Approach to Experiences Replay of Dyna-Q Learning. *IEEE Access* **2018**, *6*, 37173–37184. [CrossRef]
26. Shi, H.; Xu, M.; Hwang, K. A Fuzzy Adaptive Approach to Decoupled Visual Servoing for a Wheeled Mobile Robot. *IEEE Trans. Fuzzy Syst.* **2019**. [CrossRef]
27. Shi, H.; Lin, Z.; Hwang, K.S.; Yang, S.; Chen, J. An Adaptive Strategy Selection Method with Reinforcement Learning for Robotic Soccer Games. *IEEE Access* **2018**, *6*, 8376–8386. [CrossRef]
28. Choi, S.Y.; Le, T.; Nguyen, Q.; Layek, M.A.; Lee, S.; Chung, T. Toward Self-Driving Bicycles Using State-of-the-Art Deep Reinforcement Learning Algorithms. *Symmetry* **2019**, *11*, 290. [CrossRef]
29. Zhao, G.; Tatsumi, S.; Sun, R. RTP-Q: A Reinforcement Learning System with Time Constraints Exploration Planning for Accelerating the Learning Rate. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **1999**, *82*, 2266–2273.
30. Basyigit, A.I.; Ulu, C.; Guzelkaya, M. A New Fuzzy Time Series Model Using Triangular and Trapezoidal Membership Functions. In Proceedings of the International Work-Conference On Time Series, Granada, Spain, 25–27 June 2014; pp. 25–27.
31. Zhou, Q.; Shi, P.; Xu, S.; Li, H. Observer-based adaptive neural network control for nonlinear stochastic systems with time delay. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *24*, 71–80. [CrossRef]
32. Harper, R. Evolving Robocode tanks for Evo Robocode. *Genet. Program. Evol. Mach.* **2014**, *15*, 403–431. [CrossRef]
33. Woolley, B.G.; Peterson, G.L. Unified Behavior Framework for Reactive Robot Control. *J. Intell. Robot. Syst.* **2009**, *55*, 155–176. [CrossRef]
34. Auer, P.; Cesabianchi, N.; Fischer, P. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* **2002**, *47*, 235–256. [CrossRef]
35. Shi, H.; Xu, M. A Multiple Attribute Decision-Making Approach to Reinforcement Learning. *IEEE Trans. Cogn. Dev. Syst.* **2019**. [CrossRef]
36. Shi, H.; Xu, M. A Data Classification Method Using Genetic Algorithm and K-Means Algorithm with Optimizing Initial Cluster Center. In Proceedings of the 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 18–20 August 2018; pp. 224–228.
37. Xu, M.; Shi, H.; Jiang, K.; Wang, L.; Li, X.A. Fuzzy Approach to Visual Servoing with A Bagging Method for Wheeled Mobile Robot. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation, Tianjin, China, 4–7 August 2019; pp. 444–449.
38. Shi, H.; Xu, M.; Hwang, K.; Cai, B.Y. Behavior Fusion for Deep Reinforcement Learning. *ISA Trans.* **2019**. [CrossRef] [PubMed]