Applying Reinforcement Learning for Game AI in a Tank-Battle Game

Yung-Ping Fang
Department of Information Management
National University of Kaohsiung
Kaohsiung, Taiwan
sangis228@gmail.com

I-Hsien Ting
Department of Information Management
National University of Kaohsiung
Kaohsiung, Taiwan
iting@nuk.edu.tw

Abstract—Reinforcement learning is an unsupervised machine learning method in the area of Artificial Intelligence. It presents well performance in simulation of the thinking ability of human. However, it needs a trial-and-error process to achieve the goal. In the research field of game AI, it is a good approach to allow the non-player-characters (NPCs) of digital games to become more humanity. In this paper, we try to build a Tank-battle computer game and use the methodology of reinforcement learning for the NPCs (tanks). The goal of this paper is to make this game become more interesting from the enhanced interactions with these intelligent NPCs.

Keywords—artificial intelligence, reinforcement learning

I. INTRODUCTION

Artificial Intelligence plays an important role in modern computer games. The games can be improved to become more amusing through well designed artificial intelligence, and players can also obtain better challenge experience from playing. Therefore, how to let Non-Player-Characters (NPCs) have human-like thinking ability, and players can get fun from the interaction with NPCs, is one of the most importance subjects in game AI nowadays.

Reinforcement Learning is one of the unsupervised machine learning methods in the area of artificial intelligence. The methodology is developed based on the concept of “trial-and-error”, and the result of each “trial-and-error” will be saved as a “delay reward”. The goal of reinforcement learning is to allow the machines have human-like thinking ability.

Although the methodology of reinforcement learning can provide good ability to let the AI agents act as human beings, there are few applications that applied it in computer games. In this paper, we attempt to add the reinforcement learning to the NPCs of a tank-battle game. This main idea of the game AI is only a simple random selection mean to select all possible paths in the state space. Players can always hide in a corner, wait for the passing of the NPCs and shoot it, and to defeat NPCs accordingly. Therefore, the purpose of this paper is to improve the wisdom of the NPCs when they are walking. The NPCs will learn which places is dangerous and the hiding places of players through reinforcement learning.

The NPCs will not to be defeated silly and it then can make the game to have more fun.

The paper is organized as follows. In Section 2, the related works and literatures of reinforcement learning will be reviewed. In Section 3, we will describe the research methodology and process of the paper. Then, the experiment process and framework will be discussed in section 4. In Section 5, the experiment results and related analysis will be included. This paper is concluded in section 6, and the suggestions of future researches will be provided as well.

II. RELATED WORKS

Game AI is one of the important AI issues due to the following two reasons. The first one is that the AI enhanced digital games are funnier for human than traditional games. Thus, the players will be attracted and to play the games again and again. The second reason is that the solutions of game AI are represented as functions or algorithms to solve the AI problems in games. Therefore, if good answers can be discovered for game AI, they can help us to find good methods for solving questions in the real world [4].

In previous researches, most of game AIs are designed to focus on board games (such as Checkers, Poker and Puzzles, etc.). However, the topics of applying game AI is different types of digital games have been discussed in more and more researches nowadays [3].

Reinforcement learning is a machine learning methodology, and it is typically formulated as Markov Decision Processes [4]. In previous researches, it always apply three type of methods, they are LMS (Least Mean Squares), ADP (Adaptive Dynamic Programming) and TD (Temporal Difference Learning) [7].

LMS method is designed to compute the distance between the final state and all state (or for some cases, compute the probability of achieving the final state). Then, it will refresh all values during every step. This method have some drawbacks, e.g. slow convergence, so it needs large amount of time to find the best result [8][9].

ADP is a method to use the dynamic programming based skill. It uses a policy iteration algorithm to calculate a value, and then an estimated model will be created to fit. Furthermore, the model will be changed by each observation.
This method will be converged faster than LMS, but it still have some shortcomings in apply to large state space [8].

TD method, just like the name, is meaning to find the differ from a state to a later state. It means that we will change the utility value to adapt to later state expect value. By refreshing each values and results, it will then find the best policy to the goal[9]. In reinforcement learning, TD method would possible be the best method so far. The formulation of TD method is 

\[
U^π(s) ← U^π(s) + \alpha (R(s) + \gamma U^π(s') - U^π(s))
\]

In the formulation, \(U^π(*)\) means the utility of state under the policy \(π\), \(s\) means the current state, and \(s'\) means the next state. \(α\) is a learning rate in the formulation [8].

All we talked previously are cases in a known environment. However, they are not suitable methods when applying to an unknown environment. Due to most of real environments are unknown environment, Q-Learning [10] therefore would be a good solution to solve the problem when under the unknown environment. For ADP method, Q-Learning means to compute the Q-value (a value that according to the actions and states). The Q-value is computed from translating the original model to a new model through considering the expected best value in all actions space. The formulation of Q-learning based TD method formation is

\[
Q(a,s) ← Q(a,s) + \alpha (R(s) + \gamma \max_a Q(a',s') - Q(a,s))
\]

The difference between Q-learning and original formulation is the using of Q-value and expect maximum value from experience [7]. Related applications of applying Q-learning in game AI have been developed for a long period of time. However, the major type of the applications are board games [5], and there are little cases that applied to complex computer games, such as real-time games [1][6][11]. Thus, our purpose of this paper is using these methods to apply to a tank-battle like computer game, and make implementations to demonstrate the research results.

III. RESEARCH METHODOLOGY

The research methodology of this paper is shown in figure 1. In this paper, we firstly need to build a game as the experiment environment. Therefore, we use a game editor engine as a base to make a tank-battle game. Then, game AI is necessary for the NPCs (tanks) in the game. The reinforcement learning method is used as the AI to be added to the original NPCs. In the paper, two experiments will be held based on two different experiment methods. One is that the NPC tank moves by point to point. Another is that the NPC tank moves by different ways. After the experiments, the results will be recorded and discussed. In the end, we have some conclusions about the experiments and the reinforcement learning method.

IV. EXPERIMENT DESIGN

In order to implement the reinforcement learning based game AI, the world editor of Warcraft III [2] has been selected to create a game to simulate a tank-battle game (see figure 2). At first, the whole map will be divided into areas from left to right. The NPC tank will move randomly toward the left or right side of the map.

In the game, the player will fire at several fixed areas. Once the tank is hit by the player in a particular area, the probability of moving to the area will be reduced next time. The formulation is \(f(x)=(10-A)B\), \(f(x)\) is the probability of moving, \(A\) is the number of times which died, and \(B\) is a random value. The tank will not move to the area, if it has been killed over 10 times in the area. However, this experiment only deal with the line situation, and it should be a two-dimensional space in a true game map. Moreover, it is not a reinforcement learning due to it only considered of
previous experiences and has not gone to make "Exploration" process.

So we make some revisions to the game, the map is divided into a two-dimensional 5x3 space (see figure 4). In the map, the tank will start from a beginning point and move to the ending point (see figure 5). When the tank has been attacked, the result will be saved as a negative reward. The probability of moving to this area is updated by using the TD method. The formulation is \( f(x_i) = f(x_i) + 0.2(R(x_i) + f(x_j) - f(x_i)) \). In the formulation, \( f(x_i) \) is the probability to move previously and \( f(x_j) \) is the probability to move now. \( R(x) \) is reward, and the parameter 0.2 is our learning parameter. After approximately 15 rounds, this tank will not move to the area where it will be attacked.

However, the result still not fulfills the “Expand” characteristic of reinforcement learning. A reinforcement learning AI should unceasing to attempt the exploration of different areas, even if the area had been attacked. Due to player will not only attack the same areas and they may also change their attack target to other areas. Under this situation, the previous dangerous area will turn to a safe one. Thus, we need the "positive" reward to reconsider the previous dangerous area to improve the game AI.

In this experiment, the NPC only moves from one area to another. In fact, the moving path of a tank should be considered as one way, but not only from one area to another. A table is therefore necessary to be created to record all possible ways for a map. When a tank has been shooting down in this way, the way will be given a negative reward. Simultaneously, other ways will be given a positive reward. Table 1 shows an example of possible 6 ways in a 3x3 map are recorded, and the detail steps of each way are recorded as well.

<table>
<thead>
<tr>
<th>Way 1</th>
<th>Way 2</th>
<th>Way 3</th>
<th>Way 4</th>
<th>Way 5</th>
<th>Way 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>left</td>
<td>left</td>
<td>down</td>
<td>left</td>
<td>down</td>
</tr>
<tr>
<td>left</td>
<td>down</td>
<td>down</td>
<td>left</td>
<td>down</td>
<td>left</td>
</tr>
<tr>
<td>left</td>
<td>down</td>
<td>left</td>
<td>down</td>
<td>left</td>
<td>left</td>
</tr>
<tr>
<td>down</td>
<td>left</td>
<td>Left</td>
<td>down</td>
<td>Left</td>
<td>Left</td>
</tr>
<tr>
<td>down</td>
<td>left</td>
<td>down</td>
<td>left</td>
<td>down</td>
<td>left</td>
</tr>
<tr>
<td>down</td>
<td>down</td>
<td>Left</td>
<td>left</td>
<td>left</td>
<td>left</td>
</tr>
</tbody>
</table>

V. EXPERIMENT RESULTS

The first experiment result of this paper is shown in figure 6. The y-axle of the table is number of steps that the tank moved, and the x-axle is the number of simulation times. For example, in the first time of the simulations, the NPC tank was attacked after four steps moving. In the second time, the tank was attacked after only one step moving.

In the figure, when the learning factor becomes 0 (about 10 times), the tank will learn how to survive and it is difficult to be beaten. The result indicates that the tank already has the learning ability. However, in the experiment, the tank is only moving by one way from the right or left side of the map, and therefore the second experiment was held in this paper.
The result of the second experiment is shown in figure 7. In the experiment, the tank was moving from a fixed beginning point, and it will then move randomly to left, down or up point until the tank was attacked or achieved the ending point. In this experiment, we run about twenty times of simulation. The blue points in the figure mean that the tank has been attacked before achieving the ending point. The red points mean that the tank achieved the ending point and survive. The result shows that more times the simulation run, the higher probability that the tank can move to the ending point. Furthermore, the more times the simulation run, the more steps of the tank moved before it has been attacked or achieved the ending point.

![Figure 7. The result of experiment 2](image)

**VI. CONCLUSIONS & FUTURE RESEARCH**

In this paper, the reinforcement learning method has been applied in the NPCs to achieve the concept of game AI. This method will allow the NPCs become more humanity, and players can have more fun through playing the game. The experiment can be divided into two parts. In the first part of the experiment, the tank was moving by point to point, and the tank can move by different ways in the second part.

In game AI, the reinforcement learning method doesn’t show better performance than other machine learning methods. However, it performs good ability in acting like a real human. In other words, the reinforcement learning doesn’t intend to be a perfect AI without any mistakes, but try to be a reasonable AI which is acting like human. Under this principle, reinforcement learning presents prominent performance in the area of game AI.

However, there are still some shortcomings of reinforcement learning. For example, it needs many attempts to process rewards for achieving the accepted goal. In the future, we are expecting to find some better way to deal with this problem. Furthermore, there are some research issues we will focus in the future. For example, reinforcement learning for multi-agents (NPCs) games or real-time games, they are also big challenges for game AI.

**REFERENCES**


