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

Yung-Ping Fang
Department of Information Management
National University of Kaohsiung,
No. 700, Kaohsiung University Road, Kaohsiung, 811 Taiwan
sangis228@gmail.com

I-Hsien Ting
Department of Information Management
National University of Kaohsiung,
No. 700, Kaohsiung University Road, Kaohsiung, 811 Taiwan
iting@nuk.edu.tw
TEL:+886-7-5919751

Abstract—**Reinforcement learning is an unsupervised machine learning method in the field of Artificial Intelligence and offers high performance in simulating the thinking ability of a human. However, it requires a trial-and-error process to achieve this goal. In the research field of game AIs, it is a good approach that can give the non-player-characters (NPCs) in digital games more human-like qualities. In this paper, we try to build a Tank-battle computer game and use the methodology of reinforcement learning for the NPCs (the tanks). The goal of this paper is to make this game become more interesting due to the enhanced interactions with the more intelligent NPCs.**

Index Terms—**artificial intelligence, reinforcement learning**

I. Introduction

Artificial Intelligence plays an important role in modern computer games, as a well-designed AI allows games to become more entertaining and challenging. Therefore, how to give Non-Player-Characters (NPCs) more human-like thinking and abilities and also give players more fun from the interaction with NPCs, are very important considerations in the area of game AI.

Currently, there are a range of different types of digital games (shown in TABLE 1[3] below).

Although there are so many types of games, the moving path of NPCs is usually a critical factor. If we want the NPCs to behave like human beings, the first thing is that they should move to the meeting point. Therefore, it is of interest to find the best moving path and policy for NPCs. In this paper, the algorithm of reinforcement learning will be used as the main technique to solve the above problem.

Reinforcement Learning is one of the unsupervised machine learning methods in the area of artificial intelligence. The methodology has been developed based on the concept of "trial-and-error", and the result of each "trial-and-error" action will be saved as a "delay reward". The ultimate goal of reinforcement learning is to give the machines human-like thinking and abilities.

TABLE I. The categories of digital games

Genre	Examples
Classic board	Chess, Checkers
Adventure	Bonji's Adventures in Calabria
Team Sports	RoboCup Soccer
Real-time Individual	Bilestoad ,Space Invaders
Real-time God	SimCity
Discrete Strategy	Freeciv
Real-time Strategy	Wargus

Although the methodology of reinforcement learning can provide the AI agents with a good ability to act as human beings, there are few applications that have applied it in the computer game context. In this paper, we attempt to add reinforcement learning to the NPCs of a tank-battle game. The main design of the game AI is only a simple random selection meant to select all possible paths in the game space. Players can always hide in a corner, waiting for the passing of the NPCs. The players can then attack and defeat NPCs accordingly.

Therefore, the purpose of this paper is to improve the wisdom of the NPCs when they are walking in the game. The NPCs will learn which places are dangerous and which may hide player characters through reinforcement learning. By doing this, the NPCs will not be easily defeated by silly maneuvers and the game can be a better experience.

However, the major problem of reinforcement learning is it needs a lot of time for the AI learn the best solution. In order to deal with this problem, the concept of fuzzy logic will be used in this paper to improve the capability for reinforcement learning.

The paper is organized as follows. In Section 2, the related works and literature regarding reinforcement learning and game AI will be reviewed. In Section 3, we will describe the research methodology and process of the paper with the experimental process and framework discussed in section 4. In Section 5, the results of the experiment and related analyses will be included. This paper is concluded in section 6, and suggestions of future researches also provided.

II. RELATED WORKS

Game AI is one of the most important AI issues, for the following two reasons: the first is that AI enhanced computer games are more entertaining for people than traditional games are and thus the players will be attracted to, and play the games, continuously; the second reason is that the solutions of the game AI are represented as functions or algorithms to resolve many AI problems and . therefore, if good answers can be discovered by game AI, they can help us to find good methods for solving questions in the real world [4].

In previous studies, most game AIs were designed to focus on board games (such as Checkers, Poker and Puzzles, etc.). However, the subject of applying game AI to different types of digital games has been discussed in more and more recent studies [3].

TABLE II presents some current related research covering digital games, although different types of machine learning methodology have been used in these studies. However, there are still some unresolved problems in these studies and we expected they could perhaps be solved by using the algorithm of reinforcement learning.

TABLE II. Related game AI researches

Method	Research	Author
Case-Based Reasoning	Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game	David W. Aha et al.
Genetic Algorithm	Improving Adaptive Game AI With Evolutionary Learning	Marc Ponsen and Pieter Spronck
Neural Network	Evolving Game NPCs Based on Concurrent Evolutionary Neural Networks	XiangHua Jin, et al.

Reinforcement learning is a machine learning methodology typically formulated as Markov Decision Processes [9]. In the previous research three different types of approaches have been used in reinforcement learning: LMS (Least Mean Squares), ADP (Adaptive Dynamic Programming) and TD (Temporal Difference Learning) [8].

LMS is designed to calculate the distance between the final state and all states (or for some cases, compute the probability of achieving the final state). Then, it will refresh all values during every step. This method has some drawbacks, e.g. slow convergence, so it needs a large amount of time to find the best result [9][11].

ADP is a method that uses a 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, since the model changes with each observation this method will converge faster than when using LMS; however it still has some shortcomings in applying it to a large state space [11].

TD, just like the name, is intended to find the difference between one state and a later state. It means that we will change the utility value to adapt to a later state's expected value. By refreshing each of the values and results, it will then find the best policy to achieve the goal [9]. In reinforcement learning, the TD method would possibly be the best method so far. The TD method is based on the following formula: 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 [11].

All we have talked about above though are cases in a known environment, so, they are not suitable methods for applying to an unknown environment. Because most real environments are unknown environments, Q-Learning [2] therefore would be a good solution to solve the problems faced in an unknown environment. For the ADP method, Q-Learning means to compute the Q-value (a value that varies according to the actions and states). The Q-value is determined by translating the original model to a new model through considering the expected best value in all actions space. The formula for the Q-learning based TD method is Eq. (1):

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma \max Q(a',s') - Q(a,s)) \quad (1)$$

The difference between Q-learning and the original formulation is in the use of th o of Q-value and expecting maximum value from experience [8]. 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 [4], and there are few cases where it has been applied to complex computer games, such as real-time games [6][12][13]. Thus, our purpose in this paper is to use these methods as applied to a tank-battle like computer game, and make implementations to demonstrate the research results.

There is another popular algorithm of reinforcement learning known as the SARSA (State-Action-Reward-State-Action) algorithm. The formula for the SARSA algorithm is Eq. (2)

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma Q(a',s') - Q(a,s)). \quad (2)$$

The major difference between the Q-Learning and SARSA algorithms is that the SARSA algorithm is a on-policy algorithm, and the Q-Learning is a off-policy algorithm. Thus, the SARSA algorithm is unnecessary to calculate the maximum Q-value of the next state.

The process of reinforcement learning is shown in Fig. 1. At first, the agent will explore the environment and then carry out a rational action. After the action, every action will affect the environment, and related reward will be generated to judge this action. This process forms a reinforcement learning circle. After performing many reinforcement learning circles, the agent will learn the best policy in handling the unknown environment.

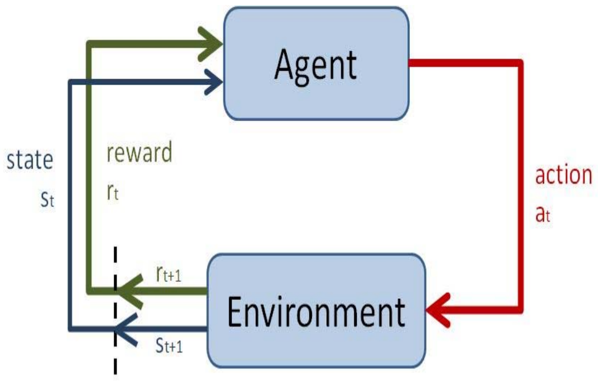

Figure 1. Reinforcement learning

The disadvantage of reinforcement learning is that the algorithm needs to try many times to get enough rewards for deciding suitable policy. Currently, some researchers have devoted themselves to dealing with this problem, such as by means of hierarchical reinforcement learning (HRL) [7]. In this paper, we will try to apply fuzzy logic in reinforcement learning to solve this problem. Fuzzy logic emerged as a consequence of 1965's fuzzy set theory by Zadeh, L.[5]. It has being applied to many fields of research, such as computer science, financial engineering, management, control, etc.

III. Research Methodology

The pilot study of this paper is shown in Fig. 2. In this paper, we firstly need to build a game as the experimental environment. Therefore, a game editor engine will be used as a base to make a tank-battle game. Then, the game AI that uses reinforcement learning algorithm will be applied to the original NPCs (tanks) in the game. In the paper, two experiments will be carried out based on different experimental methods; one where the NPC tank moves from point to point, and another where the NPC tank moves in 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.

Fig. 3 shows the research methodology of this paper and the meaning of each component of the figure in explained below.

(1) Tank game: we use a game engine to build a tank game.
(2) Reinforcement learning: Applying the algorithm of reinforcement learning to NPC tanks.
(3) Fuzzy Logic: The concepts of fuzzy logic and fuzzy sets will be used to improve the performance of reinforcement learning.
(4) Experiment and Analysis: According to the results found, we can then adjust the value of the parameters to find the best fuzzy function.

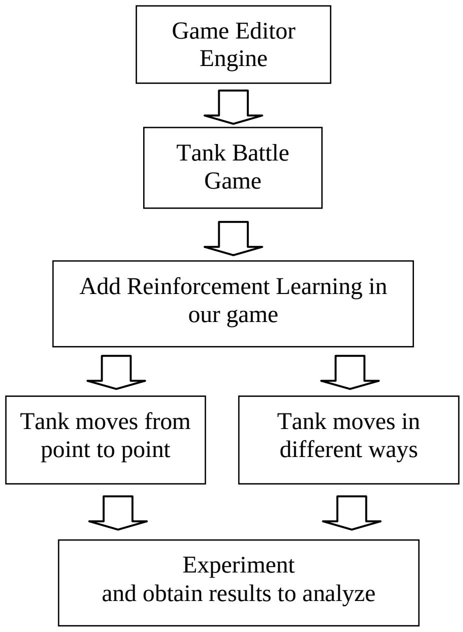
Game Editor Engine
Tank Battle Game
Add Reinforcement Learning in our game
Tank moves from point to point
Tank moves in different ways
Experiment and obtain results to analyze

Figure 2. Pilot study

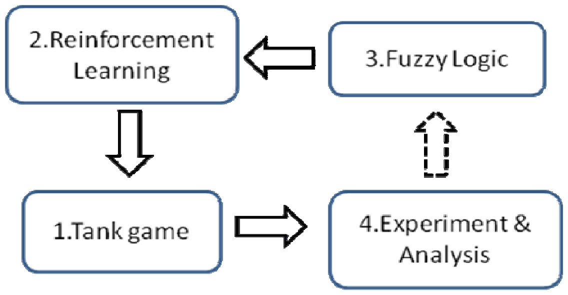

Figure 3. The Research methodology

IV. Experiment Design

In order to implement the reinforcement learning algorithm based game AI, the world editor of Warcraft III [1] has been selected to create a game to simulate a tank-battle game (see Fig. 4). At first, the whole map will be divided into areas from left to right. The NPC tank will move randomly towards the left or right side of the map.

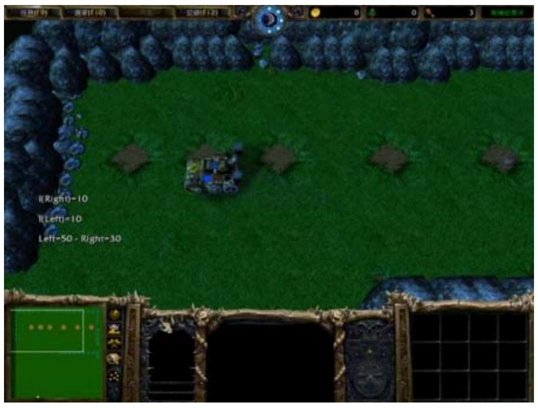

Figure 4. The game screenshot

In the game, the player will fire at several fixed areas (see Fig. 5). 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 formula is Eq. (4).

$$f(x)=(10-A)B \quad (4)$$

Where f(x) is the probability of moving, A is the number of times the tank died, and B is a random value. The tank will not move to an area, if it has been killed more than 10 times there. However, this experiment only deals with the linear situation, and it should be a two-dimensional space in a true game map. Moreover, it is not true reinforcement learning as it only considers previous experiences and has not gone on to make an ”*Exploration*” process.

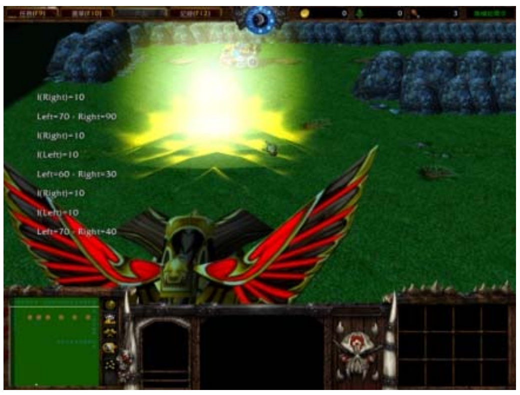
Figure 5. The attacking screenshot of the player

Therefore, we will make some revisions to the game, by dividing the map into a two-dimensional 5x3 space (see Fig. 6). In this map, the tank will start from a beginning point and move to the ending point (see Fig. 7). 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 formula for this is Eq. (3):

$$f(x_i)=f(x_i)+0.2(R(x_i)+f(x_j)-f(x_i)) \quad (3)$$

In this formula, $f(x_i)$ is the probability to move previously and $f(x_j)$ is the probability to move now. R(x) is the 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.

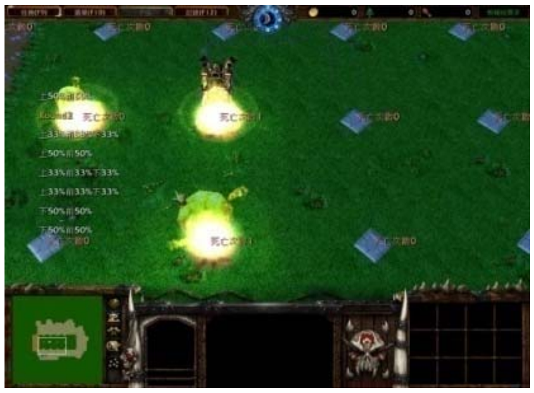
Figure 6. A 5x3 map and the attack point

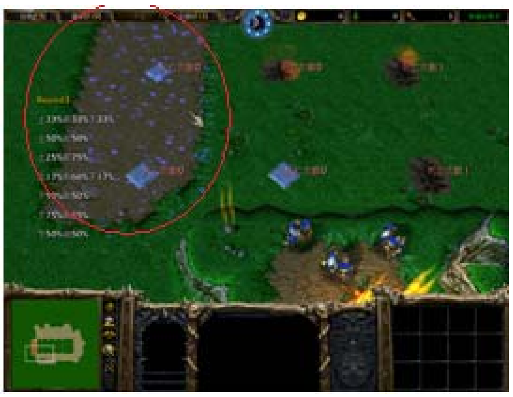
Figure 7. The ending points

However, the result still does not fulfill the *“Expand*” characteristic of reinforcement learning. A reinforcement learning based AI should unceasingly attempt the exploration of different areas, even if an area had been attacked, as a player will not only attack the same areas, they may also change their attack target to other areas. In this situation, the previous dangerous area will become a safe one. Thus, we need the ”positive” reward to reconsider the previous dangerous area to improve the game AI.

In the next experiment, we expect to create a game like Fig. 8. The players can shoot NPC tanks and NPC tanks try to move to some goal points. Furthermore, there are some walls that can the NPC can use to defend itself from attack by the players.

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 path, but not only from one area to another. It is therefore necessary to create a table to record all possible paths for a map. When a tank has been shooting down in this path, the path will be given a negative reward. Simultaneously, other paths will be given a positive reward. TABLE III shows an example of how 6 possible paths on a 3x3 map are recorded, and how the detailed steps of each path are recorded as well.

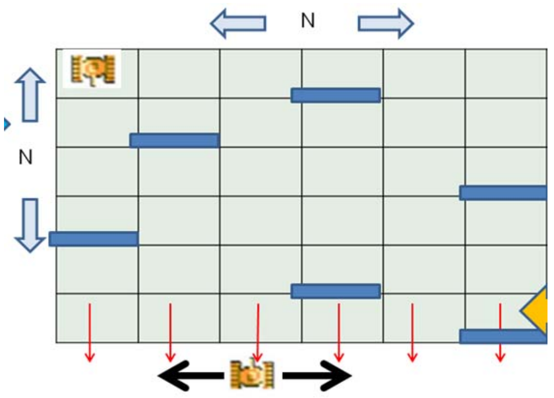

Figure 8. The concept graph of the tank game

TABLE III. The possible ways

	Step 1	Step 2	Step 3	Step 4
policy a	left	left	down	down
policy b	left	down	Left	down
policy c	left	down	down	left
policy d	down	left	Left	down
policy e	down	left	down	left
policy f	down	down	Left	left

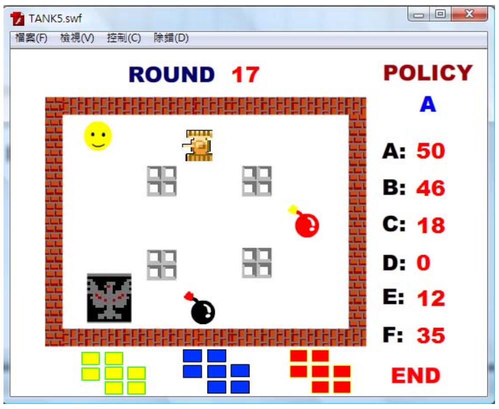

Figure 9. screenshot of experiment 3

In Fig. 9, the flash technique is used to create a game for tank-battle game simulation. The game is designed as a 3*3 map (see Fig. 9). There is a starting point in the upper right and ending point in the lower left of the map. In this game, the tank can only move left or down. So there are six policies for reinforcement learning in this game (see table III). In this game, some bombs will be located in the map and will try to attack the tank. If a tank is hit by any one of the bombs, the tank will be destroyed and it is the end of this turn of the experiment.

The reward function of this experiment is shown in TABLE IV. The values of reward function are set by our previous work: try-error to find those appropriate values.

TABLE IV. The reward function of the reinforcement learning algorithm

State s	R(s)
Tank hits bomb，this policy:	-10
Tank hits bomb，another policy:	+5
Tank successful moves to end point this policy:	+20
Tank successful moves to end point another policy:	-10

The experiment can be divided into two different parts. The first one is fixed reward, and the other is fuzzy reward. In the second part of the experiment, we use fuzzy reward "x*R(s)" to substitute for the original reward "R(s)". The variable x stands for the degree of danger. It is calculated by the fuzzy function M (s) (see Fig. 10 and Eq. (5).).

$$\mu(x) = \begin{cases} 1, & \text{If } x>=10 \\ \dfrac{10-x}{10}, & \text{If } 0<x<10 \\ 0, & \text{If } x=0 \end{cases} \qquad (5)$$

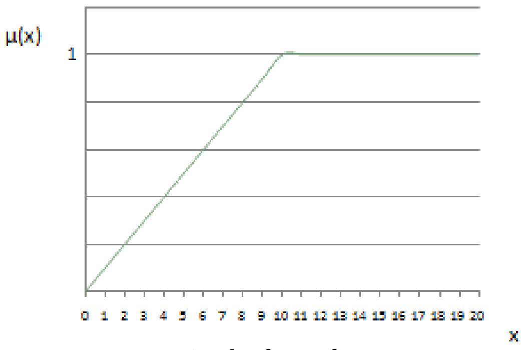

Figure 10. The fuzzy function

We use a fuzzy function to represent the degree of hazard in a 'danger value' of Xi. This value of Xi begins from 0. When NPC tank hits a bomb, the danger variable of the policy is defined as Xi=Xi+1. If tanks hits a bomb twice in a row, then the variable will be set as Xi=Xi+2. Furthermore, if the tank hits a bomb three times in a row, then the variable is Xi=Xi+3 and the maximum value is +3. The meaning of this function is that if the tank hits the bomb under the same policy, it indicates that this policy is very dangerous and the danger variable accumulation speed is higher. On the other hand, the other j (j≠i) policy the danger variables are Xj=Xj-1 which have an equal accumulation effect, and the maximum value is -3.

TABLE V shows an example of the fuzzy based reinforcement learning function. In the example, the tank hits the bomb in 1, 2, 3, 6, 8, 9 rounds.

TABLE V. An example of the fuzzy based reinforcement learning function

Round	1	2	3	4	5	6	7	8	9
Result	+	+	+	-	-	+	-	+	+
Change value	1	2	3	2	0	1	0	1	2
Total X	1	3	6	4	4	5	5	6	8

V. Experiment Result

The first experiment results for this paper are shown in figure 11. The y-axis of the table is number of steps that the tank moved, and the x-axis is the number of simulation times. For example, in the first simulation, the NPC tank was attacked after four steps moving but in the second 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 beat. The result indicates that the tank already has some learning ability. However, in the experiment, the tank is only moving by one path from the right or left side of the map, it is therefore necessary to consider the second experiment in this paper.

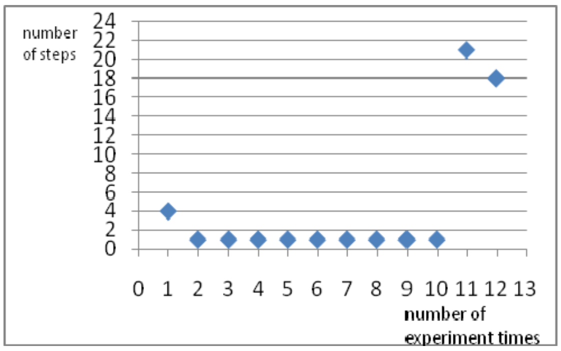

Figure 11. The result of experiment 1

The result of the second experiment is shown in figure 12. In this experiment, the tank was moving from a fixed starting point, and would then move randomly left, down or up until it was attacked or reached the ending point. In this experiment, we ran the simulation about twenty times. 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 survived. The results show that more times the simulation is run, the higher probability that the tank can move to the ending point. Furthermore, the more times the simulation is run, the more steps the tank can move before it is attacked or achieved the ending point.

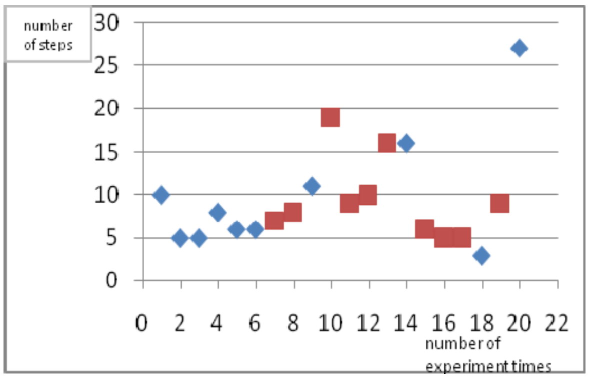

Figure 12. The result of experiment 2

The results of the two parts of experiment 3 are shown in TABLE VI. There are ten different times of the experiment (ID1~10). E1 is the experiment with a fixed reward, and E2 is the experiment with a fuzzy reward. The value in the table is total turns when there is only one policy. In figure 13, the y-axis is the number of turns when there is only one policy, and the x-axis is the total number of simulation times. It is shown that the fuzzy reward can reduce the execution time and the performance of the reinforcement learning with fuzzy reward can be clearly improved.

TABLE VI. The performance evaluation

ID	E1	E2
1	58	29
2	52	28
3	54	24
4	59	25
5	55	26
6	52	32
7	56	27
8	54	28
9	47	30
10	59	23
mean	54.6	25.6667
success rate	0.56	0.67

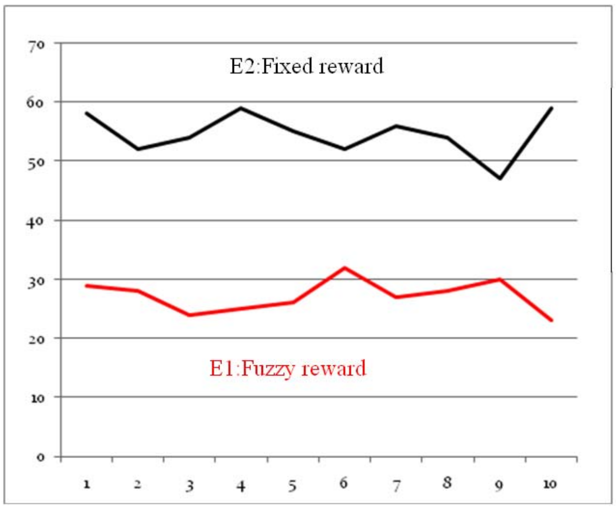

Figure 13. The result of experiment 3

VI. Conclusions & Future research

In this paper, the reinforcement learning method has been applied with the NPCs in order to improve a game AI. This method will allow the NPCs to become more human, and players can have more fun through playing the game. The experiment was divided into two parts: in the first part the tank was moving from point to point; but 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 indicates a good ability in acting like a real human. In other words, the reinforcement learning isn’t intended to produce a perfect AI that makes no any mistakes, but instead tries to be a reasonable AI which can act like a human. Under this principle, reinforcement learning presents a great performance in the area of game AI.

We had successfully used fuzzy logic in the reinforcement learning by changing fixed reward to fuzzy reward. The performance evaluation of the paper shows that the execution time can be reduced and the performance of reinforcement learning can be improved.

In the future, we expect to find some better ways 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 which are also big challenges for game AI.

REFERENCES

[1] Blizzard, “world editor of warcraft III”, [Available at http://classic.battle.net/war3/faq/worldeditor.shtml](Access date: 20 May 2009).

[2] C. J. C. H. Watkins, “Learning from Delayed Rewards.”, PhD thesis, King’s College, Cambridge, 1989.

[3] D. W. Aha, M. Molineaux and M. Ponsen, “Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game”, In Proceedings of International Conference on Case-Based Reasoning, Chicago, USA , 23-26 August 2005, pp. 5-20.

[4] I. Ghory, “Reinforcement learning in board games.”, Technical Report of Department of CS , University of Bristol, 2004

[5] L. Zadeh, "Fuzzy sets", *Information and Control*, 8: 338-353, 1965

[6] M. McPartland and M. Gallagher, “Creating a Multi-Purpose First Person Shooter Bot with Reinforcement Learning” In Proceedings of IEEE Computational Intelligence and Games, Perth, Australia, 15 - 18 December 2008, pp. 143-150.

[7] M. Ponson, P. Spronck and K. Tuyls, “Hierarchical Reinforcement Learning with Deictic.” In Proceedings of the BNAIC, 2006

[8] P. Melenchuk, E. Wong, W. Yuen, K. Wong, “CPSC 533 Reinforcement Learning”, [Available at http:// pages.cpsc.ucalgary.ca/~jacob/Courses/Winter2000/CPSC 533/Pages/CPSC-533-CourseOutline.html](Access date: 15 May 2009).

[9] R. Sutton and A. Barto, Reinforcement learning :an introduction, Mass, Cambridge, 1988

[10] S. Epstein, “Games & Puzzles”, [Available at http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/Games] (Access date: 21 May 2009).

[11] S. Russell and P. Norvig, Artificial Intelligence A Modern Approach, Prentice Hall, New Jersey, 2003.

[12] S. Wender and I. Watson, “Using Reinforcement Learning for City Site Selection in the Turn-Based Strategy Game Civilization IV”, In Proceedings of IEEE Computational Intelligence and Games, Perth, Australia , 15 – 18 December 2008, pp. 372-377.

[13] Y. Björnsson, V. Hafsteinsson, A. Jóhannsson and E. Jónsson, “Efficient Use of Reinforcerment Learning in A Computer Game” In Proceedings of International Journal of Intelligent Games & Simulation, 2008.