Dynamic Difficulty Adjustment of Game AI by MCTS for the Game Pac-Man

Ya'nan Hao, Suoju He*, Junping Wang*, Xiao Liu, jiajian Yang, Wan Huang
International School, School of Software Engineering*, Beijing University of Posts and Telecommunications, Beijing, China, 100876
yananhao.bupt @gmail.com

***Abstract*—Dynamic Difficulty Adjustment (DDA) of Game AI aims at creating a satisfactory game experience by dynamically adjusting intelligence of game opponents. It can provide a level of challenge that is tailored to the player's personal ability. The Monte-Carlo Tree Search (MCTS) algorithm can be applied to generate intelligence of non-player characters (NPCs) in video games. And the performance of the NPCs controlled by MCTS can be adjusted by modulating the simulation time of MCTS. Hence the approach of DDA based on MCTS is proposed based on the application of MCTS. In this paper, the prey and predator game genre of Pac-Man is used as a test-bed, the process of creating DDA based on MCTS is demonstrated and the feasibility of this approach is validated. Furthermore, to increase the computational efficiency, an alternative approach of creating DDA based on knowledge from MCTS is also proposed and discussed.**

Keywords- Dynamic Difficulty Adjustment, MCTS, ANN, Pac-Man, Simulation Time, Performance Curve

I. INTRODUCTION

Video Games can be overwhelming when they are too hard and boring when they are too easy. A satisfactory game-experience is required to arouse a flow, a genuine feeling of fun, of the player, which results from a balance between the challenge and the player's personal ability. Dynamic Difficulty Adjustment (DDA), also known as dynamic game balancing (DGB), is the process of dynamically adjusting the level of difficulty (or challenge) the player faces in a video game in real-time according to the player's personal ability. As a result the game won't be neither too hard to frustrate the player nor too easy to lose the player's interest [1].

Difficulty adjustment provided by traditional games has two main drawbacks. Firstly, difficulty adjustment by setting pre-defined and static difficulty levels (easy, medium, hard, and insane) is not sufficient. It lacks the flexibility to match the game challenge to the skills of every player [2]. Secondly, game challenge adjusted by such approaches is not as satisfactory as expected. Players may feel "cheated" when they are defeated by an increased number of opponents rather than the augmented intelligence of them [3].

The approach we propose is expected to outperform the existing ways of difficulty adjustment. The essence DDA based on MCTS lies in that it is the nature of opponent's intelligence that is adjusted in the game. Players are more likely to enjoy the unpredictability or novelty created by the intelligence of opponents, rather than their increased number or power. More importantly, dynamic adjustment is proved to be effective by adjusting the simulation time of MCTS so that there is always a level of challenge that meet s the cability of each player.

II. DESCRIPTION OF GAME PAN-MCN

Pac-Man is a two-dimensional, multi-agent, grid-motion, predator/prey game, which is universally considered as one of the classics of the medium, virtually synonymous with video games [4]. Due to its linearity, simplicity as well as being one of the most representative test-beds of the video game genre, Pac-Man is used as a test-bed in our experiment.

In the original game, the goal of Pac-Man (the player side) is to eat all the pellets in a maze-shaped stage while avoiding being catched by four Ghosts (the NPC side). The game is over when either all pellets in the stage are eaten by Pac-Man or Ghosts manage to catch Pac-Man. However, in order to reduce complexity and to facilitate the experiment, some modifications are made:

- The original maze is replaced by a simplified 16*16 (measured by pellets) one and Power ups are removed, as shown in Figure1;
- Two Ghosts (identified as Ghost0 and Ghost1) instead of four are designed and their speeds are as same as Pac-Man so it is impossible for a single Ghost to finish the task of eating Pac-Man (cooperation is required);
- Ghosts win when they catch Pac-Man and Pac-Man must eat 45 pellets to win; If 55 steps have been finished and still neither Pac-Man nor Ghosts win, the result is considered as a Draw.

Figure 1. Pac-Man game

III. MCTS IN THE GAME PAC-MAN

Monte-Carlo Tree Search (MCTS) is am improved algorithm based on Monte-Carlo simulations. It is a best-first search algorithm which does not require a positional evaluation function [5].

In the experiment, the Ghosts controlled by MCTS are set to fight against the computer-simulated fixed strategy Pac-Man. For Ghosts at every decision point (the turning point in the maze), the MCTS simulation builds a search tree with all the legal moves. The possible legal moves consist of Right, Left, Up and Down. Illegal moves which may lead Ghosts or Pac-man to the wall or cause collisions between Ghosts are filtered out. The detailed simulation process is illustrated by Figure 2.

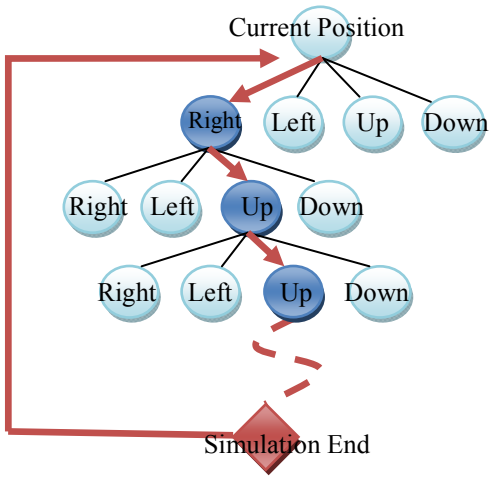

Figure 2. The MCTS simulation process of game Pac-Man

- Ghost0 expands the root node, which is the current position, by four legal moves and selects one leaf node;
- Ghost1 randomly selects next move;
- Pac-Man randomly selects next move;
- Previous steps are repeated until the end of the game is reached. Results of the simulation are returned to the selected leaf node;
- Simulations for the four leaf nodes are repeated as long as there is simulation time left. Win-rates of the four leaf nodes are calculated at last and the node with the highest win-rate is selected as the output move at the decision point.

IV. TO CREATE DDA BASED ON MCTS

This section demonstrates the process of creating DDA based on MCTS. Subsection A illustrates the reason to apply DDA by MCTS considering existing DDA approaches. Subsection B describes the process of generating the performance curve of NPCs controlled by MCTS. Subsection C gives the validation of the curve generated. And in Subsection D, the effectiveness of DDA based on MCTS is discussed.

A. *Why DDA based on MCTS?*

1) *Existing DDA*

Existing DDA is often realized by adjusting parameters such as number of enemies, frequency and power. However this mechanism can cause frustration or loss of interest of players because they may feel unfair or "cheated" when they are defeated by an opponent assisted with more associates or weapons rather than an opponent that is more intelligent [6].

FSM (Finite State Machine) can be utilized to generate opponent AI at different intelligence levels in some games. However difficulty adjustment by AI created by FSM still faces a significant problem — the resulted levels of challenge are too static to flexibly match the competence of every player since player abilities are not limited to a few levels. Such a problem also exits for the mechanism of adjusting parameters.

2) *Advantages of DDA based on MCTS*

DDA based on MCTS is expected to outperform the present ways of difficulty adjustment in terms of three points:

- It is the nature of intelligence of opponent that is adjusted in games. Players are set to fight against opponents with matched intelligence. They are fairly treated.
- It offers a way of continuous and dynamic adjustment of game challenge — by adjusting the simulation time of MCTS algorithm which is applied to generate opponent AI.
- DDA based MCTS is realized by comprehensive computation so that it requires little domain knowledge and human participation [3].

B. *Performance of NPCs controlled by MCTS with different Simulation Time*

The performance of opponent AI generated by the MCTS relies on the simulation time of MCTS. Normally the longer the simulation time, the more intelligent the NPCs are [7]. Therefore, it is proposed that the level of game challenge could be adjusted by modulating the simulation time of MCTS. The following steps are followed to get the statistical results in the experiment:

- Ghosts controlled by MCTS with different simulation times are set to fight against with Pan-Man with the ForwardOrRight strategy, which means Pac-Man can only choose to go forward or turn right at the corner.
- For each simulation time, a total of 250 gameplays are conducted. And win-rate of Ghosts is calculated for every 50 gameplays. So totally five win-rates result from one simulation time.
- Performance of Ghosts in each simulation time is represented by their average win-rates, calculated from the five resulted win-rates.

- Discrete points which indicate the relation between the simulation time and the performance could be spotted, as in Figure 3.

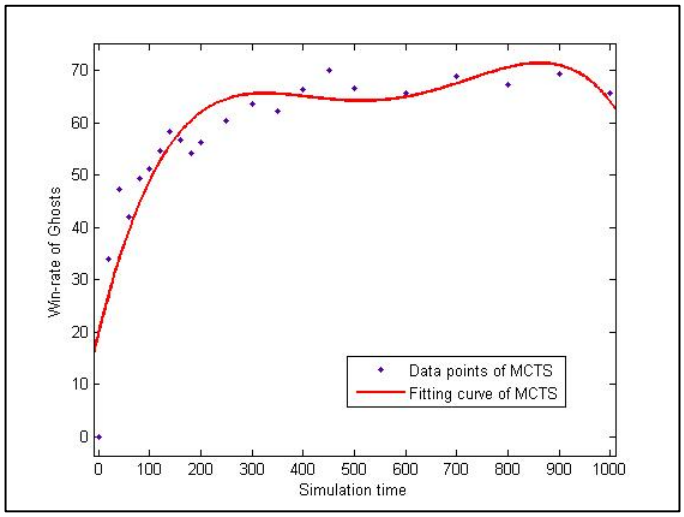

Figure 3. Performance of Ghosts controlled by MCTS with different simulation time

C. Determination and Validation of the performance curve

It can be observed a continuously increasing trend and a wide range of the win-rates of Ghosts against simulation time of MCTS. However, to realize effective DDA, a relatively precise regression function needs to be determined from the discrete points.

1) Determination

By the curve fitting tool in MATLAB, the polynomial fitting function and curve from the discrete points in Figure 3 is easy to be generated.

However to obtain a relatively high precision of the performance function for DDA, it should be cautious to choose the degree of polynomial fitting. A basic principle is to obtain a good demonstration of the data trend without losing too much certainty. Two measures are seriously considered — the R-square and the Confidence Bounds.

- R-square measures how successful the fit is in explaining the variation of the data; the higher degree of polynomial, the larger the value of R-square [8]. It can be seen from Figure 4 that the R-square has values that are quite close from the 4th degree to the 9th degree.

- Confidence Bounds demonstrates the reliability of the fitting curve. In contrary to the R-square, the Confidence Bounds will increase tremendously with the increase of the polynomial degree [9].

Therefore, to obtain a fitting curve well demonstrating the data variation without losing a certain level of reliability, we focused our attention on the 4th, 5th, and 6th degree polynomial whose R-square values are just a little less than the highest degree but confidence bounds are a lot smaller.

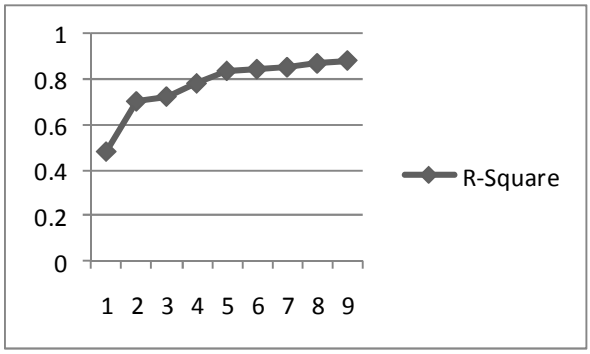

Figure 4. The R-square values of polynomial from 1st degree to 9th degree

2) Validation

For further validation, we select three crucial points on the 4th, 5th and 6th polynomial fitting curves respectively to examine their precision. Values of simulation time which has the win-rates 30%, 50% and 70% are figured out from the regression function and reapplied to NPCs controlled by MCTS. The resulted real win-rates corresponding to the simulation time with target win-rate are calculated and shown in TABLE I.

TABLE I. WIN-RATE VALIDATION OF THE THREE FITTING CURVE

Target win-rates	4th degree curve		5th degree curve		6th degree curve	
	Simulation time(ms)	*Real win-rate*	*Simulation time(ms)*	*Real win-rate*	*Simulation time(ms)*	*Real win-rate*
30%	28	37%	29	41%	27	47.3%
50%	105	56%	92	52%	77	54.7%
70%	777	69.3%	604	58%	496	63.7%

Comparing the resulted real win-rates with the three target win-rates, it turns out the real win-rates tested for the 4th degree fitting curve is the closest to the target win-rates. Therefore the 4th degree fitting curve is chosen in the experiment, as shown in Figure 3.

And the regression function is

$$y = -5.67*x^4+17.6*x^3-11.1*x^2-0.81*x+65.6 \quad [1]$$

D. The Effectiveness of DDA Based on MCTS

A well-designed game should be able to create Flow experience for players so that they are immersed in the game and lose track of time and worries [10]. Flow arises when the balance between challenge and ability is reached, which can be achieved by DDA. Applying DDA based on MCTS, it is feasible to optimize players' satisfaction by creating a match between the challenge of the game and the ability of different types of players:

- For normal players, Flow arises when an even game (50% win-rate of NPCs) is created. By the performance function of MCTS, simulation time with the win-rate of 50% can be applied to create an even game.

- For the veterans who prefer to more challengeable games, simulation time with win-rate of 70% from the performance function could be set.
- For the novices who prefer to less challenge, simulation time with win-rate of 30% could be applied.
- Though the type of the player is not defined, the resulted range of the performance curve, from 20% to 70% is wide enough to meet the ability of players of all types, which means dynamic adjustment of game challenge is realizable.

V. TO CREATE DDA BASED ON ANN FROM MCTS

DDA based on MCTS still has drawbacks in terms of its time efficiency and consumption of system resources. These limitations makes the DDA by MCTS is not suitable for multi-player on-line games. Hence we propose DDA based on knowledge from MCTS, which is more resource efficient. In the experiment, Artificial Neural Network (ANN) is applied to acquire knowledge from MCTS simulations and imitate the performance of MCTS [7].

This section explains the process of creating DDA by ANN trained by data from MCTS simulation. Subsection A describes how to train neuro-controllers for NPCs by data collected by MCTS. Subsection B demonstrates the performance curve of ANN-controlled NPCs. DDA. And in Subsection C, the limitations of DDA based on ANN are discussed.

A. To train ANN by data from MCTS

In the experiment, the three-Layered Backpropagation (BP) Artificial Neural Network is applied to train neuro-controllers by data from MCTS to control the performance of Ghosts. The following steps are followed to train ANN by data collected by MCTS simulation.

1) Step1 Selecting data

Data about every move of Pac-Man and Ghosts are recorded in each gameplay in the previous MCTS simulations. And all data are utilized without optimizing, no matter the result is win, lose or draw for the Ghosts, because it is the most satisfactory opponent AI that is to be created by DDA rather than the most challengeable opponent AI.

2) Step2 Training

In the experiment, the tool of WEKA (Waikato Environment for Knowledge Analysis) is applied to train ANN by data from each simulation time of MCTS.

- Attributes that indicates the states of Pan-Man and the two Ghosts, as well as the environment in each move are selected to as inputs of ANN for the two Ghosts. Almost the same types of inputs are selected for the two Ghosts except one more input – ghost0Direction – for Ghost1 so as to control the cooperative behavior. TABLE II shows the selected inputs for ANN of Ghost0.
- The direction of each Ghost, which needs to be determined at the decision-making points, is the output of ANN, as in TABLE III. And the attribute of ghost direction holds four values – Up, Down, Right and Left.
- The number of nodes in hidden layer is determined by that of the input layer and output layer, as shown in TABLE IV.

TABLE II. INPUTS FOR ANN OF GHOST 0

	Attribute	Description
1	stepNum	Step number in each gameplay
2	playerX	Horizontal displacement of Pac-Man
3	playerY	Vertical displacement of Pac-Man
4	playerDirection	Direction of Pac-Man
5	playerUpBean	Pellet at up side of Pac-Man
6	playerDownBean	Pellet at down side of Pac-Man
7	playerLeftBean	Pellet at left side of Pac-Man
8	playerRightBean	Pellet at right side of Pac-Man
9	ghost0X	Horizontal displacement of Ghost0
10	ghost0Y	Vertical displacement of Ghost0
11	ghost0Up	Wall at up side of Ghost0
12	ghost0Down	Wall at down side of Ghost0
13	ghost0Left	Wall at left side of Ghost0
14	ghost0Right	Wall at right side of Ghost0
15	ghost1X	Horizontal displacement of Ghost1
16	ghost1Y	Vertical displacement of Ghost1
17	ghostsDistanceX0	Horizontal distance between Ghost0 and Ghost1
18	ghostsDistanceY0	Vertical distance between Ghost0 and Ghost1
19	ghost0PlyaerDistanceX	Horizontal distance between PacMan and Ghost0
20	ghost0PlyaerDistanceY	Vertical distance between PacMan and Ghost0
21	ghost1PlyaerDistanceX	Horizontal distance between PacMan and Ghost1
22	ghost1PlyaerDistanceY	Vertical distance between PacMan and Ghost0

TABLE III. OUTPUTS FOR ANN OF GHOSTS

Ghost	Attribute	Description	Values
Ghost0	ghost0Direction	Direction of Ghost0	Up, Down, Right, Left
Ghost1	ghost1Direction	Direction of Ghost1	Up, down, right, left

TABLE IV. NUMBER OF NODES IN EACH LAYER

Ghost	Input layer	Hidden layer	Output Layer
Ghost0	22	13	4
Ghost1	23	13	4

3) Step3 Initializing ANN

To initialize ANN is to connect the training data (weights and bias at each node) to the neural network source code, which means "brains" of the two Ghosts are constructed. The "brain" can generate appropriate output at each decision point based on knowledge learned from the data of MCTS simulation.

B. Performance of ANN-controlled NPCs

After ANN corresponding to each simulation time of MCTS are built and tested, the performance curve of Ghosts controlled by ANN can be generated, as shown in Figure 5. Even though ANN is independent of time, the simulation time of MCTS from which ANN is constructed is used as the independent variable of the curve because it is the simulation time of MCTS that is to be adjusted by DDA based on ANN.

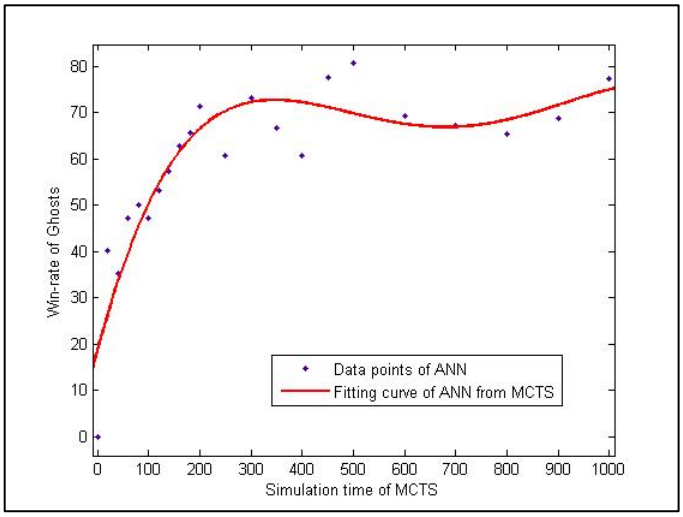

Figure 5. Perfomance of Ghosts controlled by ANN from MCTS of different simulation time

It can be observed the performance curve of Ghosts controlled by ANN demonstrates a similar increasing trend of MCTS and it is also able to offer a wide enough range to realize DDA. However, the discrete points spotted show some obvious oscillations. These oscillations, which may add to the uncertainty of performance curve of ANN, are possibly due to the nature of ANN, which is discussed in next subsection.

C. Limitations of DDA based on ANN

ANN controls NPCs by knowledge acquired in MCTS simulations. So it has no dependency on computation resources and is ideal for multi-player online games. However ANN also has its own limitations.

- ANN is a mechanism with high nonlinearity and approximation, it's often not so straightforward to understand exactly what a neural network is doing or predict precisely what output the network will generate [11]. Sometimes ANN may not produce the result as we expected. This is why some obvious oscillations are appear on the performance curve of ANN compared with that of MCTS.
- In the experiment, ANN can be seen as a passive learner or approximator of MCTS. All it does is to produce output based on knowledge from MCTS. However as is mentioned earlier, data to train ANN are selected without optimizing. Therefore it's possible that knowledge acquired by ANN may be generated from some "bad data", which means data from gameplays that are lost by NPCs. This explains why sometimes ANN may produce a quite bad performance even though the corresponding MCTS simulations generate a good result. For this problem, optimizing ANN by select "good data" is a solution.
- Another problem is that ANN can only be implemented based on ready knowledge. And the knowledge is acquired from gameplays between NPCs controlled by MCTS with player a specific strategy. Therefore player strategy model must be identified before DDA based on ANN is applied. Research on combining DDA based ANN with Strategy-based Player Modeling (SBPM) is considered as our future work.

VI. CONCLUSION

Based on the discussion above, it is proved that by adjusting the simulation time of MCTS which generates opponent AI, Dynamic Difficulty Adjustment can be created to optimize players' satisfaction in game experience. DDA based by MCTS is effective in dynamically matching the game challenge with ability of different types of players by adjusting intelligence of opponent AI. However, because of the great computation intensiveness and consumption of system resources, CI-based DDA is only applicable to standalone PC game. The alternative approach, DDA based on ANN from MCTS, is proposed to realize DDA for online game. The feasibility of creating DDA by ANN is validated. But approaches to control the performance ANN precisely and automatically need to be explored for real application

REFERENCES

[1] "Dynamic Game Balancing." Wikipedia. Retrieved on Mar, 21, 2010, from http://en.wikipedia.org/wiki/Dynamic_game_difficulty_balancing).

[2] Hunicke, R., Chapman, V., AI for Dynamic Difficulty Adjustment in Games. In Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence (AAAI '04) (San Jose, California) AAAI Press, 2004.

[3] I. S. JaSuoju, H., et al. Creating Challengeable and Satisfactory Game Opponent by the Use of CI Approaches, JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 2, No. 1, pp. 41 ~ 63, 2010

[4] Yannakakis, Giorgios N, "AI in Computer Games: Generating Interesting Interactive Opponents by the use of Evolutionary Computation". PhD thesis, University of Edinburgh, 2005.

[5] H.J. van den Herik, and B. Bouzy. Progressive strategies for Monte-Carlo Tree Search. New Mathematics and Natural Computation, 4(3), 2008.

[6] Auto-dynamic difficulty, retrieved on Mar, 21, 2010, from http://dukenukem.typepad.com/gamematters/2004/01/autoadjusting.html.

[7] Xiao, L., et al. To Create Intelligent Adaptive Game Opponent by Using Monte-Carlo for the Game of Pac-Man. in Natural Computation, 2009. ICNC '09. Fifth International Conference on. 2009.

[8] "Curve fitting". Wikipedia. Retrieved on Mar, 21, 2010, from http://en.wikipedia.org/wiki/Curve_fitting.

[9] "Confidence interval", Wikipedia. Retrieved on Mar, 21, 2010, from http://en.wikipedia.org/wiki/Confidence_interval.

[10] Chen, J., Flow in games (and everything else) in Communications of the ACM. 2007, ACM Press p. 31-34.

[11] David M. Bourg, G.S., AI for Game Developers. 2004, O'Reilly, ISBN 0-596-00555-5. Chapter14 Neural Networks.