An Adaptive Predator/Prey Game

Hao-Min Hsieh
Dept. of Information and Design
Asia University
louis.tech@m2k.com.tw

Ling-Ling Wang
Dept. of Information Communication
Asia University
ling@asia.edu.tw

Abstract

In most of current computer games, computer-controlled opponents are often guided by a limited set of fixed strategies. The behavioral repetition of computer-controlled opponents reduces the human player's enjoyment. Furthermore, the predefined fixed difficulty levels of a game are not satisfied by most human players. Human players prefer challenging games which keep level with them. Hence, in this study we proposed a fuzzy approach to adapting the opponents' tactics to behavior of the player such that the player's win/loss rate is kept at a desired rate. The value of the desired rate can be preset by the player according to the player's preference for challenge. Our experiments are conducted on a predator/prey game. The experimental results demonstrate that adaptation efficiency and robustness of the proposed method is sufficient for the game played by human players.

Keywords: adaptive game, artificial intelligence, fuzzy logic, predator/prey game

1. Introduction

Traditionally, game developers invest most resources in creating realistic graphics and sound effects. However, in recent years game developers have paid more resources in developing challenging games. Game artificial intelligence (AI) is the key to providing challenging gameplay experiences for most games [5]. Game AI refers to techniques needed in computer games to produce the illusion of intelligence in the behavior of computer-controlled opponents. Some AI techniques have been adopted in game AI [10]. Although the use of these techniques produces an improvement in game AI, it is currently still unsatisfactory. One reason is that computer-controlled opponents are often guided by a limited set of fixed strategies. Human players can thus predict behaviors of computer-controlled opponents after playing a game several times, and then feel the game boring [7]. Some game developers lend variety to behaviors of computer-controlled opponents through writing lots of elaborate scripts. Nevertheless, the process of script writing may be tedious and error-prone.

To solve the aforementioned problem, researches about adaptive game AI [1, 3-4, 9, 11-15] have been proposed. Adaptive game AI is game AI with the capabilities of self-correction and creativity [13]. It is traditional game AI incorporated with machine learning techniques. Learning can be applied before or after a game's release. Learning is called offline if it is applied before a game's release (or in the period of game development), whereas it is called online if applied after a game's release (or during gameplay). Spronck et al. [12] proposed a neuro-evolutionary technique to train offline a computer-controlled opponent of a shooting game by playing with a scripted opponent. Ponsen et al. [9] used an evolutionary algorithm to offline generate tactics for a real-time strategy game. Houlette [4] proposed player modeling to change opponents' tactics according to the player's profile which is a record of the player's behavior during gameplay. Demasi and Cruz [3] proposed coevolutionary approaches to online evolving computer-controlled opponents' tactics of a simple action game. Yannakakis and Hallam [14, 15] applied a neuro-evolutionary technique to train opponents offline for a multi-agent cooperation game. The trained opponents also evolve during gameplay based on the proposed interest-judging criteria so that the player has a continuous interest in the game. Andrade et al. [1] applied reinforcement learning to learn tactics of a fighting game. If the game is hard for the player, the opponent chooses a suboptimal tactic to achieve game balance. Spronck et al. [11] proposed dynamic scripting to online create scripts for the opponents. A script is composed of a set of if-then rules selected from a rulebase. Whether a rule is selected for a script depends on the weight of the rule. The weight of a selected rule is adjusted according to the results of battles after each round ends.

Although the above-mentioned approaches can adapt opponents' tactics to the player, the numbers of rounds needed for adaptation sometimes vary greatly even when playing against the same fixed-strategy opponent [11, 14-15]. Hence in this paper, a new fuzzy approach is proposed to adapt the game's tactics to behavior of the player such that "balance" between the player and computer-controlled opponents can be achieved effectively. In this study, we model intuitive and subjective human ideas of controlling opponents in a game into fuzzy rules. Fuzzy inference is applied during gameplay to guide the opponents. After each round of the game, we strengthen or weaken the opponents' skill according to the player's performance. Skill balance between the player and opponents is achieved through rules enabling and disabling.

The rest of the paper is organized as follows. In Sec. 2, we present a detailed description of a predator/prey game which is used as the test bed of our research. In Sec. 3, we describe the proposed fuzzy adaptive approach for the predator/prey game. The experiments conducted for evaluating the performance of the proposed approach are described in Sec. 4. Finally in Sec. 5, conclusions are summarized.

2. The Dead End World

In this study, we choose a predator/prey game called Dead End [15] as our test bed because it is an interesting multi-agent game and its graphics is simple. Hence, we can place emphasis on investigation of game AI but not graphics. Dead End is a two-dimensional multi-agent predator/prey game. Fig. 1 shows a snapshot of a Dead End game. In the game field, the x axis directs from left to right and the y axis from top to bottom. The characters at the upper place of the game field are Ghosts, and the one at the lower is Player. The top of the game field with no barrier is assumed as the exit. The dimension of the game field is 320 × 480 pixels, and that of Ghosts and Player is 20 × 20 pixels. Player is initially at the lower place of the game field, while Ghosts are at the upper ones.

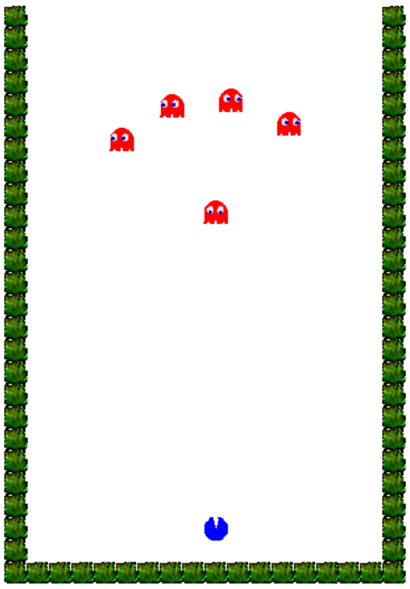

Fig. 1. The snapshot of the Dead End game

Player aims to reach the exit and avoid Ghosts, whereas Ghosts aim to defend the exit and to kill Player. Player owns health points which denote Player's health. A Ghost attack Player by touching Player. Once Player is touched by a Ghost, Player's health points decrease one by one until Player leaves the Ghost. Player dies and loses the game if the number of health points equals to zero. Additionally, if Player cannot reach the exit within a predefined period of time, Player loses the game. After Player either wins or loses the game, a new round starts. Player moves at double the Ghosts' speed, and thus it is impossible for a single Ghost to catch Player. Hence, all Ghosts must hunt cooperatively. In this research, our goal is to develop an adaptive approach to controlling these Ghosts effectively.

To test the adaptability of computer-controlled Ghosts during experiments, we design three types of fixed-strategy Player. These fixed strategies are somewhat like those which human players may adopt. They are described below from simple to complicated.

(a) Simple-avoidance (SA) Player: An SA Player can move in four directions (north, south, east, and west). It moves directly toward the closest exit if no Ghost is in its visible area (40 pixels around it). However, if Ghosts approach it and enter its visible area, the Player moves in the direction which is not blocked by Ghosts. It behaves similarly to a novice human player.

(b) Advanced-avoidance (AA) Player: The strategy of an AA Player is the same as that of an SA Player except that an AA Player may move in eight directions (plus northeast, northwest, southeast, and southwest). Therefore it can reach the exit more effectively than an SA Player.

(c) Cost-based (CB) Player: This strategy is obtained by modifying that proposed by Yannakakis et al. [15]. Compared with the other two types of Player, the CB Player performs a more effective ghost-avoiding and exit-achieving strategy. Its movement is determined based on the costs of its eight possible moving directions. In Fig. 2, the eight grid squares of dimension 20 × 20 pixels denote eight moving directions of the CB Player, and the coordinate of the dot on the perimeter of a grid square denotes the coordinate of the grid square. The cost of each moving direction is calculated through the function $C(x, y)$:

$$C(x,y) = \Delta E(y) + G(x,y) \tag{1}$$

$$\Delta E(y) = |y - y_e| \tag{2}$$

$$G(x,y) = \sum_{n=1}^{N} \frac{\rho}{|x_{g,n} - x| + |y_{g,n} - y|} \tag{3}$$

where (x, y) denotes the coordinate of a grid square, $\Delta E(y)$ is the distance from the grid square to the exit, and y_e is the y-axis coordinate of the exit. We ignore the distance in x-axis while evaluating $\Delta E(y)$ since the whole top of the game field is assumed as the exit. N is the number of Ghosts; $(x_{g,n}, y_{g,n})$ is the coordinate of the n^{th} Ghost's center. ρ is a parameter that indicates the weight of $G(x, y)$ to $C(x, y)$. Before starting a movement, CB Player first calculates the costs of the eight moving directions, and then moves 20 pixels in the direction with the minimal cost.

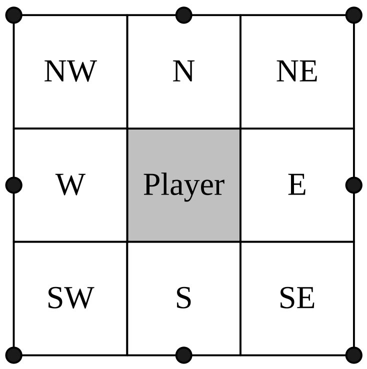

Fig. 2. Eight moving directions of CB Player

3. Generation of Adaptive Opponents

3.1 The fuzzy rulebase

In this study, intuitive and subjective ideas or strategy of controlling Ghosts are beforehand modeled into fuzzy rules. These rules with fuzzy inference are used to guide Ghosts during gameplay. In the following paragraphs, we first describe the proposed strategy for Ghosts, and then explain the implementation of this strategy in fuzzy logic.

To catch Player efficiently, Ghosts appear in the role of vanguard or fullback. Every time the game starts, one of Ghosts at the lowest place plays the vanguard and the others the fullbacks. The vanguard leads fullbacks to chase Player, and the fullbacks follow the vanguard and form a fan-like formation as shown in Fig. 1. We design five types of movement for the vanguard and fullbacks as summarized in Table 1. The type A and type C movements are used to chase Player to catch it. The type B movement is used to block Player by predicting Player's moving direction while Player tries to bypass Ghosts from the flanks. The use of type D and type E movements let Ghosts form a fan-like formation to prevent Player from approaching the exit.

To model the five types of movement into fuzzy rules, we define five fuzzy and one crisp input variables used in the antecedents of fuzzy rules, and five fuzzy output variables used in the consequents of fuzzy rules. The six input variables provide Ghost with information about distances to Player, to the exit, to the vanguard, to the closest fullback, and whether Player passes ghost, as listed in Table 2. The five fuzzy output variables are used to derive Ghost's moving directions, as listed in Table 3. The defuzzified output of a fuzzy rule is used as a weight of the derived moving direction. For example, the defuzzified output of a rule whose consequent is *chasePlayer* is used as a weight of the moving direction toward Player. When two or more rules are activated at a time, the weights corresponding to the same derived moving direction are summed. Then Ghost's moving direction is the derived one with highest weight.

Table 1. Five types of Ghosts' movement

Role	Movement
Vanguard	A: Chase Player. B: Block Player by predicting Player's moving direction.
Fullback	C: Chase Player. D: Approach or part from the vanguard to keep an appropriate distance. E: Part from the closest fullback to keep an appropriate distance.

Table 2. Input variables

Variable name	Meaning
distToPlayer	The distance from Ghost to Player
distToExitY	The y-axis distance from Ghost to the exit
distToPlayerX	The x-axis distance from Ghost to Player.
distToVanguard	The distance from Ghost to the vanguard
distToFullbackX	The x-axis distance from Ghost to the closest fullback
bePassed	A Boolean value denoting whether Player passes Ghost

Table 3. Output variables

Variable name	Meaning
chasePlayer	Move toward Player
partFromPlayerY	Part from Player in y axis
approachVanguard	Move toward the vanguard
partFromVanguardY	Part from the vanguard in y axis
partFromFullbackX	Part from the fullback in x axis

The fuzzy sets of each fuzzy input and output variable are designed by our experience. As shown in Figs. 3 to 12, we define each fuzzy variable with only two or one fuzzy set whose membership function is of triangle or trapezoid shape. To implement the five types of movement for Ghosts, we designed 40 fuzzy rules in total, 12 rules for the vanguard and 28 rules for the fullbacks. The correlation-minimum inference method proposed by Mamdani [6] is applied to fuzzy inference and the weighted average method [2] to defuzzification. The use of fuzzy rules and fuzzy inference results in the robustness of Ghosts' behavior during gameplay.

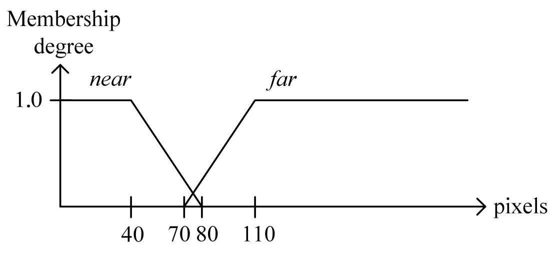

Fig. 3. The membership functions of *distToPlayer*

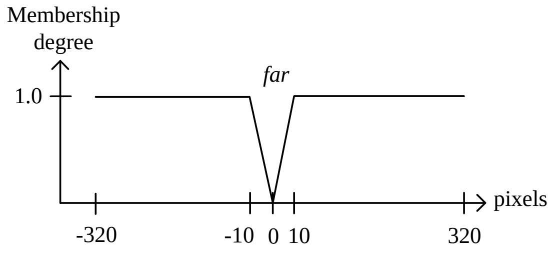

Fig. 4. The membership function of *distToPlayerX*

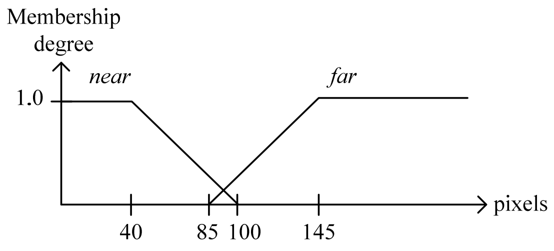

Fig. 5. The membership functions of *distToExitY*

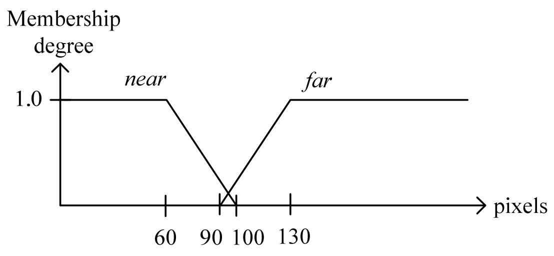

Fig. 6. The membership functions of *distToVanguard*

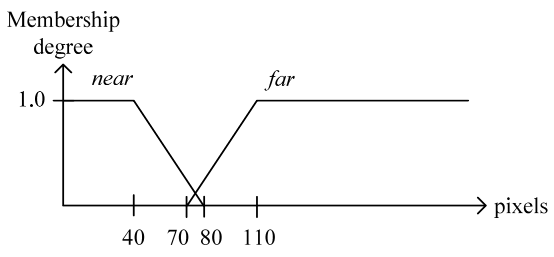

Fig. 7. The membership functions of *distToFullbackX*

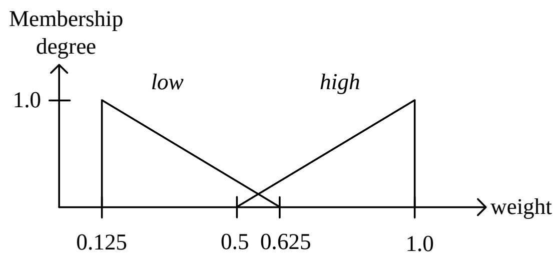

Fig. 8. The membership functions of *chasePlayer*

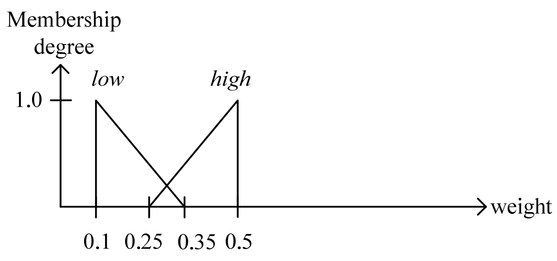

Fig. 9. The membership functions of *partFromPlayerY*

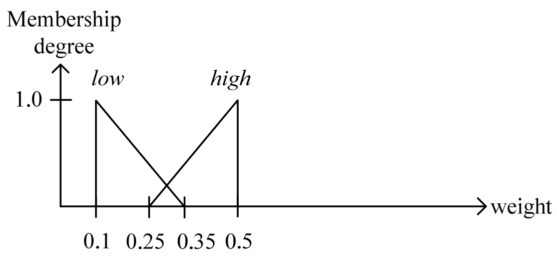

Fig. 10. The membership functions of *approachVanguard*

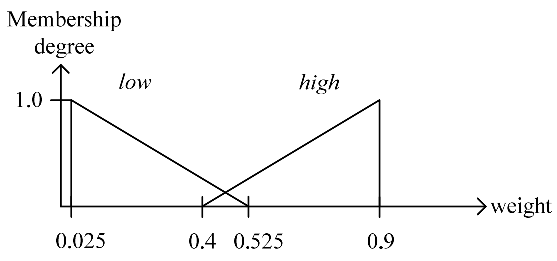

Fig. 11. The membership functions of *partFromVanguardY*

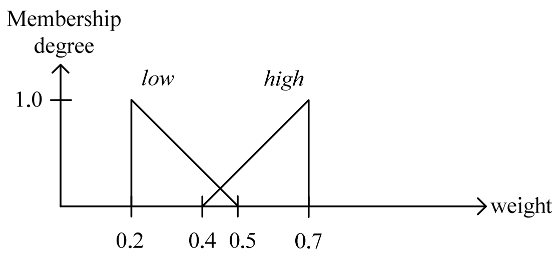

Fig. 12. The membership functions of *partFromFullbackX*

3.2 Online rule enabling/disabling

In this study an online rule-selection method is used to adapt Ghosts' tactics to behavior of the Player. This method is based on a rule enabling/disabling mechanism. As mentioned above, Ghosts are controlled by a fuzzy controller with a rulebase of 40 fuzzy rules. They behave well when all 40 rules are enabled. However, if some rules in the rulebase are disabled, Ghosts may behave unskillfully and become weaker due to the incompleteness of the strategy. Thus, if the game is hard for Player, we weaken Ghosts' skill level by disabling rules in the rulebase. In contrast, if the game is easy, we increase Ghosts' level by enabling rules which are disabled previously. Through the rule enabling/disabling mechanism, Ghosts may keep level with Player after a few rounds of adaptation. Fig. 13 shows the flowchart of the online rule-selection method, and each step of the flowchart is described in detail in the following.

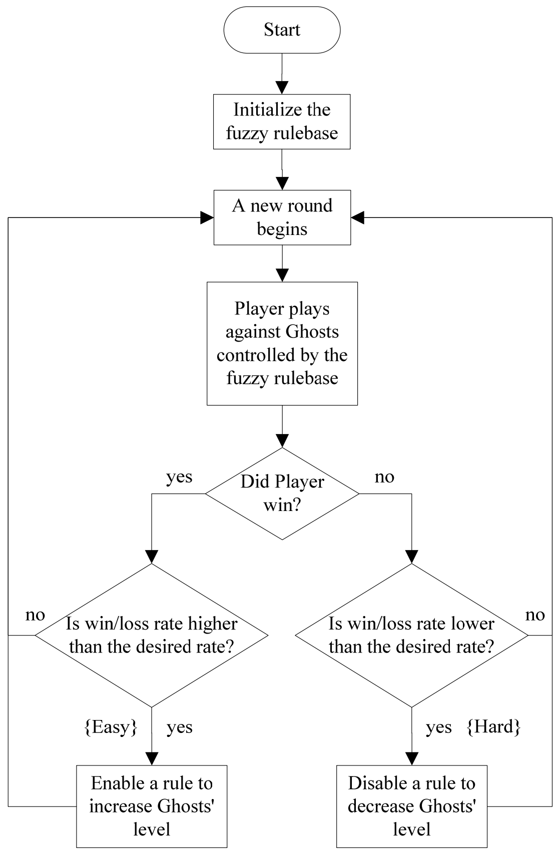
Start
Initialize the fuzzy rulebase
A new round begins
Player plays against Ghosts controlled by the fuzzy rulebase
Did Player win?
yes
no
Is win/loss rate higher than the desired rate?
Is win/loss rate lower than the desired rate?
no
no
{Easy}
yes
yes
{Hard}
Enable a rule to increase Ghosts' level
Disable a rule to decrease Ghosts' level

Fig. 13. Flowchart of online rule enabling/disabling

Whenever a win/loss occurs during gameplay (that is, a round ends), we evaluate whether the difficulty level of the game fits for Player according to Player's current win/loss rate. The win/loss rate is defined as the ratio of the number of wins to the number of losses from the game begins. Player's and Ghosts' skill levels are said to be balanced if Player's win/loss rate achieves a desired rate. The value of the desired rate can be preset by the player according to the player's preference for challenge. If the player wants a game with more (or less) challenge, he/she can set a desired rate with the value smaller (or larger) than 50%. The default value of the desired rate is 50% if the player does not set its value. The game is considered hard for Player if Player loses the game and the current win/loss rate is lower than the desired rate. On the other hand, if Player wins the game and the current win/loss rate is higher than the desired rate, we consider the game easy for Player.

When the game is hard for Player, we disable a rule in the rulebase to decrease Ghosts' skill level. Which rule is disabled depends on the contribution of the rule. The contribution of a rule is defined as the number of times for which the rule is adopted in the current round. A rule is said to be adopted if the moving direction derived by the rule is selected as the Ghost's moving direction. The least-contribution rule (a rule with least contribution but with contribution larger than zero) is disabled if the game is hard for Player. Zero-contribution rules are not considered for disabling since Ghosts' tactics may not be changed if they are disabled or not. The reason why we choose the least-contribution rule but not the most-contribution one is that Ghosts' skill level may change dramatically if the most-contribution rule is disabled. Since there are five fullbacks but only one vanguard in the Ghosts' team, the probability that the rules for fullbacks are adopted is five times of that for the vanguard. Therefore the contributions of the rules for the fullbacks are divided by five when selecting rules for disabling.

If the game is easy for Player, we enable a rule which is disabled previously in the rulebase to increase Ghosts' skill level. The procedure of enabling rules is different from that of disabling rules. First, we divide all rules in the rulebase into groups according to the types of the consequents of rules and then calculate the contribution of each group. The contribution of a group is the sum of contributions of all enabled rules in this group. Second, we calculate the sum of input membership degree values for each rule disabled previously. Then, the rule with the highest sum value is enabled. If there are two or more rules with the same sum of input membership degree values, the rule whose group has less contribution is prioritized. The reason why we prioritize the least-contribution group but not the most-contribution one is that if the game is easy for Player, Ghosts' current strategy might be ineffective, and thus the rules in the most-contribution group might also be ineffective.

4. Experimental Results

In the study, we implemented the proposed approach in Java language and utilized the NRC

FuzzyJ Toolkit [8] (a comprehensive API of fuzzy logic) for fuzzy inference. In our experiments, there are five Ghosts and one Player in the Dead End game. The initial number of Player's health points is 40, and the value of the parameter of CB Player is 2950. The time limit of a round is 20 seconds. That is, if Player does not reach the exit after playing the game for 20 seconds, Player loses the game and a new round starts. During gameplay if all rules are disabled or the outputs of all enabled rules are zero, a specific rule which guides Ghost directly toward Player is enabled. The use of the specific rule prevents Ghosts from being at a standstill.

The experiments were aimed at evaluating adaptation efficiency and the flexibility for achieving different desired win/loss rates. In the experiments, we ran the game with 20 initially enabled rules which are obtained by manually selecting core rules from the total 40 rules. The experiments were conducted by playing against three types of fixed-strategy Player, that is, SA, AA, and CB Players. We set three desired win/loss rates: 25%, 50%, and 75%. For each Player, we ran one hundred rounds and recorded the win/loss rates every round. The experimental results shown in Figs. 14 to 16 demonstrate that the proposed method can adapt Ghosts' tactics for each type of Players and achieves the desired win/loss rates after fifty or even less of rounds. These figures also show the robustness of our method because the win/loss rates are kept well.

5. Conclusions

In this study we proposed a fuzzy approach to adapting opponent's tactics to the player based on a rule enabling/disabling mechanism, and the player's win/loss rate can be kept at a desired rate by our method. This approach was tested on a multi-agent predator/prey game called Dead End. The experimental results show that the proposed method can efficiently adapt opponents' tactics to different types of fixed-strategy player and achieve the desired win/loss rates in fifty or even less of rounds. Thus adaptation efficiency of this method is well sufficient for the games played with human players.

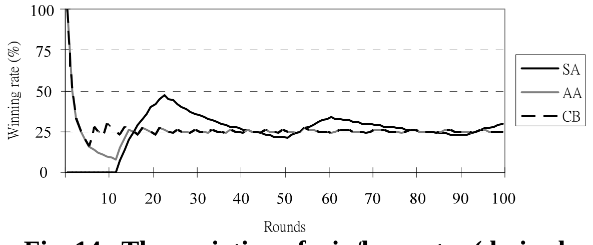

Fig. 14. The variation of win/loss rates (desired rate = 25%)

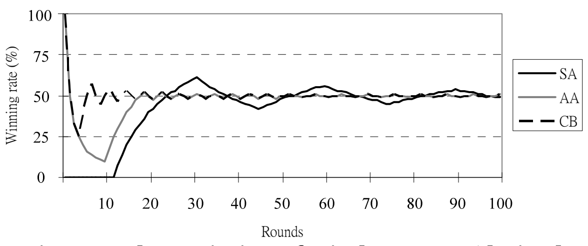

Fig. 15. The variation of win/loss rates (desired rate = 50%)

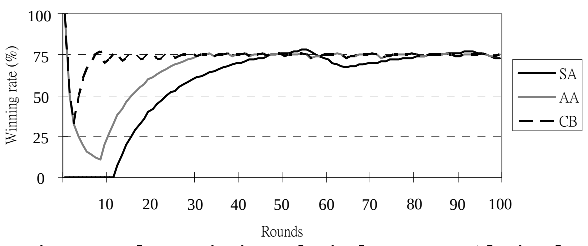

Fig. 16. The variation of win/loss rates (desired rate = 75%)

Acknowledgements

This work was supported by the National Science Council of Taiwan under the grant NSC 96-2221-E-468-010-MY2.

References

[1] G. Andrade, G. Ramalho, H. Santana, V. Corruble, "Challenge-sensitive action selection: an application to game balancing," *Proc. of IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2005, pp. 194-200.

[2] R. Babuska, *Fuzzy Modeling for Control*, Kluwer Academic Publishers, Boston, 1998.

[3] P. Demasi and A.J. de O. Cruz, "Online coevolution for action games," *International Journal of Intelligent Games and Simulation*, Vol. 2, No. 2, 2003, pp. 80-88.

[4] R. Houlette, "Player modeling for adaptive games," in AI Game Programming Wisdom 2, S. Rabin Ed. Charles River Media, Hingham, MA, 2004, pp. 557-566.

[5] J.E. Laird and M. van Lent, "Human-level AI's killer application: computer game AI," *Proc. of AAAI 2000 Fall Symposium on Simulating Human Agents*, 2000, pp. 80-87.

[6] E.H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers*, Vol. 26, No. 12, 1977, pp. 1182-1191.

[7] J. Manslow, "Learning and Adaptation," in AI Game Programming Wisdom, S. Rabin Ed. Charles River Media, Hingham, MA, 2002, pp. 557-566.

[8] NRC FuzzyJ Toolkit,

http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html

[9] M. Ponsen, H. Muñoz-Avila, P. Spronck, and D.W. Aha, “Automatically generating game tactics with evolutionary learning,” *AI Magazine*, Vol. 27, No. 3, 2006, pp.75-84.

[10] S. Rabin, “Common game AI techniques,” in AI Game Programming Wisdom 2, S. Rabin Ed. Charles River Media, Hingham, MA, 2004, pp. 3-14.

[11] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, “Adaptive game AI with dynamic scripting,” *Machine Learning*, Vol. 63, No. 3, 2006, pp. 217-248.

[12] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, “Improving opponent intelligence through offline evolutionary learning,” *International Journal of Intelligent Games and Simulation*, Vol. 2, No. 1, February 2003, pp. 20-27.

[13] P.H.M. Spronck, “Adaptive game AI,” Ph.D. dissertation, Maastricht University, Maastricht, The Netherlands, 2005.

[14] G.N. Yannakakis, and J. Hallam, “A generic approach for generating interesting interactive Pac-Man opponents,” *Proc. of the IEEE Symposium on Computational Intelligence and Games*, Essex University, UK, April 2005, pp. 94-101.

[15] G.N. Yannakakis, J. Levine, and J. Hallam, “An evolutionary approach for interactive computer games,” *Proc. of the 2004 Congress on Evolutionary Computation*, Portland, Oregon, USA, June 2004, pp. 986-993.