CD-ROM Proc. 1st International Conference on Digital Interactive Media Entertainment and Arts
(DIME-ARTS 2006), Oct. 2006, Bangkok, Thailand.

Investigation of Various Online Adaptation Methods of Computer-Game AI Rulebase in Dynamic Scripting

Syota Osaka
Intelligent Computer Entertainment Laboratory
Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu
Shiga 525-8577, Japan
rs008023@se.ritsumei.ac.jp

Ruck Thawonmas
Intelligent Computer Entertainment Laboratory
Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu
Shiga 525-8577, Japan
ruck@ci.ritsumei.ac.jp

Tomoya Shibazaki
Intelligent Computer Entertainment Laboratory
Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu
Shiga 525-8577, Japan
rs018031@se.ritsumei.ac.jp

ABSTRACT

In this paper, we investigate various online adaptation methods of computer-game AI rulebase in a technique called Dynamic Scripting (DS). DS is a technique for balancing the level of computer controlled characters that play computer games against human players. It online updates rule weights in rulebase that influence the behavior of the computer controlled character. However, the weight updating mechanism of DS is not effective if improper initialization of the rulebase is done. In our previous work, we proposed a complementary method to DS that replaces inefficient rules with randomly generated rules. In the present work, we propose three more methods and compare them with the previously proposed one and with the original DS, using a simulator in which one adaptive character duels against one hard-coded character (HC). Our finding is that the method that replaces an inefficient rule with the rule least similar, among the given candidates, to the inefficient one is of the best performance, in terms of the winning rate against HC, for mediocre and weak initial rulebase conditions.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Knowledge acquisition, Parameter learning

General Terms

Game design

Keywords

Dynamic Scripting, Rulebase, Computer games, AI characters, Non player characters

1. INTRODUCTION

AI research on evolving game strategies has a relatively long history starting from its applications to chess, checkers, and then now to many commercial games such as soccer games [1]. Our research target is the fighting genre where player controlled characters fight against other characters controlled by computer.

For the fighting genre, balancing the level of the computer controlled character (or AI character) with that of the player is important [2]. In particular, in order to keep the game challenging, it is important to dynamically increase the level of the computer controlled one to the level that matches the player's level. This is because the player's level normally increases as he or she plays longer. Failure to do so will make the player lose interest in the game. In practice, the vast majority of AI characters are controlled by scripts. It is almost impossible to prepare hard-coded AI scripts in advance that could perform the balancing task.

Among a variety of evolving game techniques, we focus on Dynamic Scripting (DS) proposed by Spronck [3]. DS is an unsupervised learning algorithm with a simple but efficient mechanism for dynamically constructing a proper script by a set of rules from given rulebase. It has already gained interests from game industry [4]. In our previous work [5], we proposed a method to complement DS for the case where rulebase is not initially well designed. In such a case, the original DS mechanism of updating rule weights does not work. The proposed method replaces inefficient rules in the rulebase with randomly generated rules.

In this paper, we propose three more methods for online adaptation of rulebase and compare them with the one proposed in [5] and with the original DS. Performance comparison is done using a simulator in which one adaptive character duels against one hard-coded character. The rest of the paper is organized as follows. In Section 2, the outline of DS is given, followed by the description on the proposed methods. Section 3 describes the game simulator, and Section 4 discusses performance evaluation. Section 5 concludes the paper and describes our future work.

2. DYNAMIC SCRIPTING WITH ONLINE ADAPTATION OF RULEBASE

2.1 Basic Mechanism

The basic mechanism of DS is depicted in Fig. 1, and is outlined as follows:

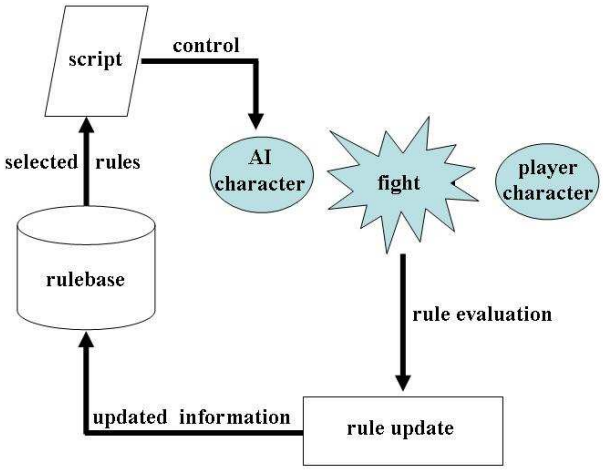


Figure 1: Dynamic Scripting mechanism

1. Rulebase consisting of multiple rules is assigned to a group of AI characters of the same type.
2. A set of rules are stochastically selected from the rulebase, according to their weights, to construct the script of the AI character.
3. The AI character fights against the player controlled character according to the content of its script.
4. Each rule weight is updated according to the fighting result.
5. Go to 2

2.2 Online Rulebase Adaptation Methods

Loopholes in the game design might lead to the presence of inefficient rules in rulebase. Under this situation, DS could not catch up with human players. To solve this problem, we previously proposed a framework for online adaptation of rulebase in [5]. The basic idea behind this framework is that every rule with weight less than a given threshold is discarded and replaced with a new rule not duplicate with M rules previously discarded as well as all rules currently stored in the rulebase. Therein, a method was proposed that randomly generates a new rule for each discarded rule. Henceforth, this method is called RAN.

In the present work, every discarded rule is replaced with a rule selected from N such new rules. We propose three methods, MOS, LES, and MIX, having different mechanisms in rule selection. Each method is described below as follows:

MOS: Select the rule most similar to the rule with the highest weight in the rulebase.

LES: Select the rule least similar to the discarded rule.

MIX: Select the rule most similar to the rule with the highest weight in the rulebase and, at the same time, least similar to the discarded one.

The similarity between two rules is determined by the inverse of the distance between them. For MIX, the rule with the maximum value of $\alpha - \beta$ is selected, where α is the distance to the discarded rule, and β the distance to the rule with the highest weight in the rulebase. The definition of distance and the description of the game simulator, for performance evaluation, are given in the following section.

3. GAME SIMULATOR

3.1 Game System

As in [5], in our simulator, there are two AI characters, a hard-coded character (HC) representing a human player and an adaptive character based on DS (AC). Each character has two parameters, HP (Hit Point) and MP (Magic Point). A character wins when HP of its opponent becomes zero while its HP remains. HP and MP of a character will be decreased if it is attacked by the opponent. MP of a character will also be decreased if it performs some particular actions. Table 1 shows the actions available to the two characters and their relation to HP and MP. Rulebase consists of 50 rules, and the weight of a rule is bounded to have its value between 0 and 1000. For each duel between HC and AC, 20 rules are stochastically selected, according to their weights, from the rulebase to construct the script of AC. An if-then rule is represented by six digits described from left to right as follows:

First digit: fixed to 1

Second digit HP condition of AC

Third digit: MP condition of AC

Fourth digit: HP condition of HC

Fifth digit: MP condition of HC

Sixth digit: action of AC

where, for each rule, all four conditions are connected by AND operation.

Table 2 describes the parameters for both the conditional part and the action part of a rule. In our simulator, the following rules are not included in the rulebase because they have no realistic meaning.

- all rules with the second digit = 5 or the fourth digit = 5
- all rules with the sixth digit = 0
- all rules with the third digit = 5 and the sixth digit ≥ 3

Henceforth, all other rules are called *meaningful* rules.

3.2 Distance Computation

With the above rule representation, the distance between rules i and j, $d(i,j)$, used for rule selection in Section 2.2, is computed as follows:

$$d(i,j) = d_{cond}(i,j) + \gamma d_{act}(i,j),$$

Table 1: Summary of the game actions and their relation to HP and MP

Action Type	MP Consumption	Action Description
HP attack type 1	0	Decrease HP of the opponent by 8 to 12 with the average value of 10
HP attack type 2	0	Decrease HP of the opponent by either 0 or 20 with the average value of 10
HP attack type 3	10	Decrease HP of the opponent by 18 to 22 with the average value of 20
MP attack	10	Decrease MP of the opponent by 18 to 22 with the average value of 20
Heal	10	Increase HP of itself by 28 to 32 with the average value of 30

Table 2: Description of the rule parameters

Parameter Type	Parameter Value		
	0	1	2
HP Condition	always holds	HP $\geq$ 75	50 $\leq$ HP $<$ 75
MP Condition	always holds	MP $\geq$ 75	50 $\leq$ MP $<$ 75
Action	no action	*HP attack type 1*	*HP attack type 2*
Parameter Type	**Parameter Value**		
	0	1	2
HP Condition	25 $\leq$ HP $<$ 50	0 $<$ HP $<$ 25	HP = 0
MP Condition	25 $\leq$ MP $<$ 50	10 $\leq$ MP $<$ 25	MP $<$ 10
Action	*HP attack type 3*	*MP attack*	*heal*

where $d_{cond}(i,j)$ and $d_{act}(i,j)$ represent the conditional-part distance and the action-part distance between rules i and j, respectively, and γ is an adjustment factor.

As shown in Table 2, every digit in the conditional part, except the first digit, has a value indicating a range of the corresponding parameter, and adjacent values indicate adjacent ranges. As a result, we define the conditional-part distance between two rules as the sum of the absolute difference between each conditional parameter of them. For example, the conditional-part distance between rules 143205 and 120433 is 10, from $|4-2|+|3-0|+|2-4|+|0-3|$.

On the contrary, adjacent values of the sixth digit do not always imply that the corresponding actions are similar. The action-part distance between two rules is thus subjectively defined and represented by the matrix shown below.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 3 & 4 \\ 2 & 2 & 0 & 2 & 4 \\ 3 & 3 & 2 & 0 & 3 \\ 4 & 4 & 4 & 3 & 0 \end{pmatrix}$$

In the above matrix, the ijth element represents the distance between actions i and j.

The adjustment factor γ is for balancing the conditional-part distance and the action-part distance. It is determined as follows:

$$\gamma = \frac{\text{average conditional part distance}}{\text{average action part distance}}$$

Assuming for simplicity that parameter values are uniformly assigned[1], γ becomes 125/36.

[1]In this case, the average conditional-part distance is $((1\times 10+2\times 8+3\times 6+4\times 4+5\times 2)/36)\times 4$, and the average action-part distance is $(1\times 2+2\times 6+3\times 6+4\times 6)/25)$.

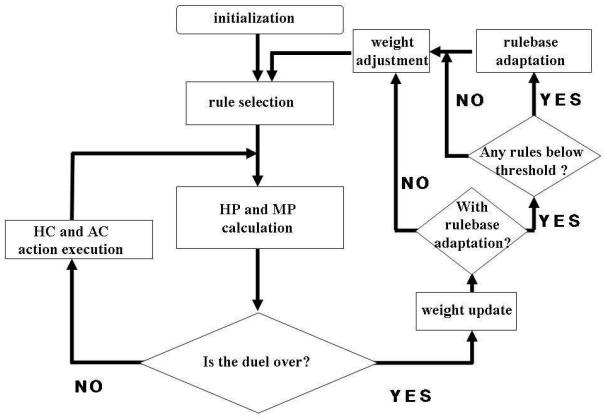


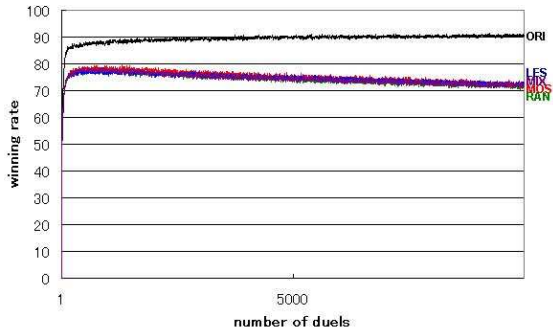
Figure 2: Simulator flowchart

3.3 Simulation Steps

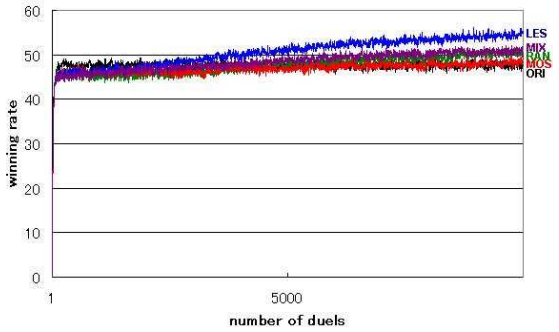
Figure 2 shows the flow chart of the simulator for a series of duels between HC and AC. At the initialization step, 50 rules are assigned to rulebase, and each rule weight is initialized to 500. A duel starts from the rule selection step, where, as mentioned earlier, 20 rules are stochastically selected from the rulebase, according to their weights, to construct the script of AC for that duel; in addition, each character is assigned 100 HPs and 100 MPs.

At the HC and AC action execution step, an action of AC is randomly selected for execution from the action part of one of the script rules whose all four conditions hold. An action of HC is selected and executed according to its pre-defined rules, given in the next section.

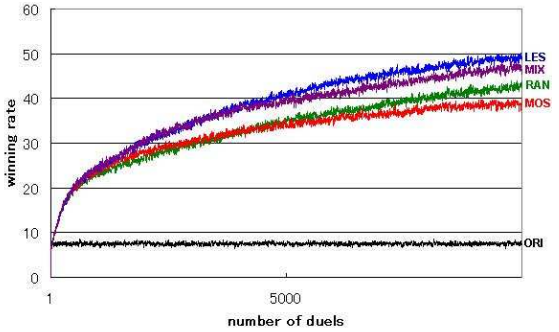
Once a duel is over the weight of the ith rule, rule[i].weight,



(a) Strong Initial Rulebase



(b) Mediocre Initial Rulebase



(c) Weak Initial Rulebase

Figure 3: The winning rate of each method

in the script of AC is updated at the weight update step as follows:

$$\text{rule}[i].\text{weight} = \min(\max(\text{rule}[i].\text{weight} + \text{HP(AC)} - \text{HP(HC)}, 0), 1000), \quad (1)$$

where HP(AC) and HP(HC) denote the remaining HP of AC and that of HC at the end of the duel, respectively. At the end of this step, all rules in the script are returned to the rulebase.

At the rulebase adaptation step, every rule in the rulebase with weight less than 20 is replaced with a new rule by one of the methods in Section 2.2. Such a new rule is selected from a set of N candidates, where N is set to 5. The parameters of these candidates are randomly and uniformly assigned an integer value from 0 to 5 for the conditional part and from 1 to 5 for the action part. The candidate parameters are subject to the condition that they define meaningful rules not duplicate with M rules previously discarded as well as all rules currently stored in the rulebase, where M is set to 100. The weight of a selected rule is then initialized to 500.

At the weight adjustment step, the weights of all rules in the rulebase are adjusted such that their sum is equal to that at the initialization step, i.e., 25000.

Table 3: The average and the standard deviation of the winning rate at the 10000th duel

Method	Initial Rulebase Type					
	Strong		Mediocre		Weak	
	Avg.	Std.	Avg.	Std.	Avg.	Std.
ORI	90.29	9.94	47.64	18.29	7.250	8.28
RAN	71.40	14.94	50.20	17.69	42.89	19.97
MOS	71.92	15.37	48.32	17.255	38.83	22.09
LES	72.30	15.35	55.32	17.24	49.98	18.81
MIX	72.08	15.25	51.37	17.68	46.75	19.61

4. PERFORMANCE EVALUATION

We compared the performances of AC without the rulebase adaptation step (henceforth, called ORI), RAN, MOS, LES, and MIX, when each method dueled the same opponent HC used in [5], under the same initial rulebase condition. The performance index in use is the wining rate against the opponent HC among the recent ten duels at the given number of duels; unless stated otherwise, it is simply called the winning rate. Three types of initial rulebase, strong rulebase, mediocre rulebase, and weak rulebase, were prepared by trial-and-error selecting the initial 50 rules such that the wining rate of ORI at the 300th duel was about 90%, 50%, and 10%, respectively. For each method under each initial rulebase condition, the experiment was composed of 1000 experimental trials. All results discussed below are average values of 1000 experimental trials.

The aforementioned opponent HC was hard-coded as follows:

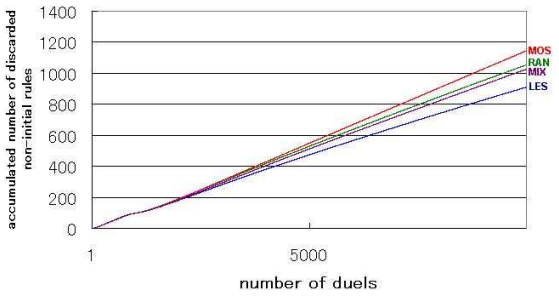
- If its MP $\geq$ 10 and its HP $<$ 50 then *heal*
- If its MP $\geq$ 10 and its HP $\geq$ 50 then *HP attack type 3*
- Otherwise, *HP attack type 1*

Figure 3 shows the wining rate of each method over the number of duels under each initial rulebase condition.

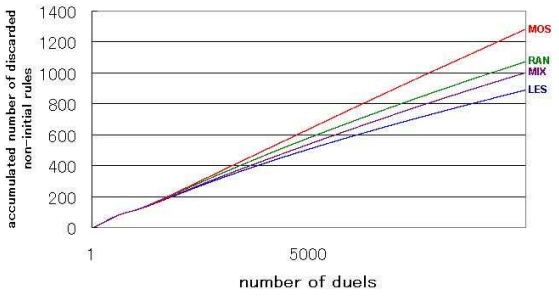
Table 3 lists the average number and the corresponding standard deviation of the winning rate at the 10000th duel of each method.

The above results are summarized as follows:

- The performance of the original DS (ORI) depends heavily on the quality of the initial rulebase. In particular, there is no sign of learning for the weak initial rulebase condition.



(a) Mediocre Initial Rulebase



(b) Weak Initial Rulebase

Figure 4: The accumulated number of discarded non-initial rules

- DS with methods for online adaptation of rulebase, RAN, MOS, LES, and MIX, perform significantly better than ORI for the weak initial rulebase condition.
- LES is of the best performance among all methods for both the mediocre and weak initial rulebase conditions.

Based on these observations, one question arises: why does addition of a new rule most similar to the rule with the highest weight in the rulebase, have less effect in the performance than addition of a new rule least similar to the discarded one? Our conjecture is as follows. In the beginning of a duel sequence, a rule with the highest weight might not always be an efficient rule while a discarded rule represents more precisely an inefficient rule, especially for both the mediocre and weak initial rulebase conditions.

Figure 4 shows the accumulated number of rules that are generated for replacing inefficient initial rules, but are later discarded. Since the number of LES is less than that of both MOS and MIX, The results in this figure validate the above conjecture.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated four methods that complement Dynamic Scripting by replacing inefficient rules in rulebase with new rules. As shown in our performance evaluation, LES, the method that replaces an inefficient rule with the rule least similar, among the given candidates, to the discarded one is of the best performance, in terms of the winning rate, for both the mediocre and weak initial rulebase conditions. In addition, its converged winning rate under these two conditions is about 50, which represents a challenging game level, not too easy or too difficult.

However, LES still requires a relatively large number of duels to increase its performance. Research on how to increase the learning speed is left as our future work.

6. ACKNOWLEDGMENTS

The first author was supported in part by the Ritsumeikan University's Kyoto Art and Entertainment Innovation Research, a project of the 21^{st} Century Center of Excellence Program funded by Ministry of Education, Culture, Sports, Science and Technology, Japan.

7. REFERENCES

[1] S.M. Lucas and G. Kendall. Evolutionary computation and games. *IEEE Computational Intelligence Magazine* 1(1):10-18, Feb. 2006.

[2] B. Pfeifer. Narrative Combat: Using AI to Enhance Tension in an Action Game. In *Game Programming Gems 4*, pages 315-324, 2004.

[3] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma. Adaptive Game AI with Dynamic Scripting. *Machine Learning*, 63(3):217-248, 2006.

[4] P. Spronck. Dynamic Scripting. In *Game AI Programming Wisdom 3.* Ed. Steve Rabin, pages 661-675. Charles River Media, Hingham, MA, 2006.

[5] Ruck Thawonmas and Syota Osaka. A Method for Adapting Computer-Game AI Rulebase. In DVD Proc. ACM SIGCHI International Conference on Advances in Entertainment Technology 2006 (ACE 2006), Hollywood, USA, Jun. 2006.