DVD Proc. ACM SIGCHI International Conference on Advances in Entertainment Technology 2006 (ACE 2006), Hollywood, USA, Jun. 2006.

# A Method for Online Adaptation of Computer-Game AI Rulebase

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

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

## ABSTRACT

It is not an easy task to balance the level of computer controlled characters that play computer games against human players. In this paper, we focus on a method called Dynamic Scripting (DS) that has been recently proposed for this task. This method online updates rule weights in rulebase that describe 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. To cope with this problem, we propose a complementary technique to DS that replaces inefficient rules with newly generated rules. Evaluation of the proposed technique is conducted confirming its effectiveness.

## Categories and Subject Descriptors

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

## General Terms

Game design

## Keywords

Dynamic Scripting, Rulebase, Computer games, Computer controlled characters

## 1. INTRODUCTION

AI research in computer and video games has a relatively long history starting from its applications to chess and other board games, and then now to many more genres including shooting, racing, fighting and strategy games [1]. In our study, we tackle the fighting genre where a player controlled character fights against another character controlled by computer.

For the fighting genre, applicable to others though, it is crucial to well balance the level of the computer controlled character (or AI character) with that of the player [2]. In particular, it is important to dynamically increase the level of the computer controlled one to the level that matches the player level which normally increases as he or she plays longer. If this fails, the player will get bored with the game. In practice, it is almost impossible to prepare hard-coded AI scripts in advance that could perform this task [3].

Among a variety of techniques, we focus on Dynamic Scripting (DS) recently proposed by Spronck [3]. DS is an unsupervised learning algorithm that has a simple but efficient mechanism for dynamically constructing a proper script by a set of rules from given rulebase. Though relatively new, DS has already started drawing interests from game industry [4]. In this paper, we propose a technique to complement DS for the case where rulebase is not initially well designed. In this case, thereby, the original DS mechanism of updating rule weights does not work. The proposed technique replaces inefficient rules in the rulebase by newly generated rules.

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 technique in Section 3. Section 4 discusses evaluation results, and Section 5 concludes the paper and describes our future work.

## 2. DYNAMIC SCRIPTING

Figure 1 depicts the basic mechanism of DS, which is outlined as follows:

1. Rulebase consisting of multiple rules is assigned to a group of AI characters of the same type.
2. A set of rules are 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.

*The 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*ACE 06*, June 14-16, 2006, Hollywood, California, USA.
Copyright 2006 ACM 1-59593-380-8 /06/0006 ...$5.00.

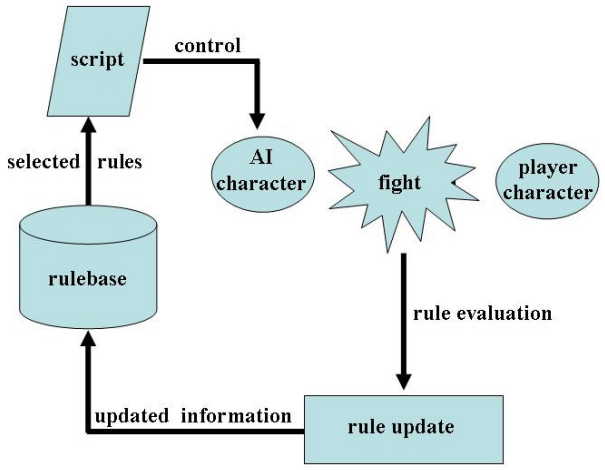


**Figure 1: Basic mechanism of Dynamic Scripting**

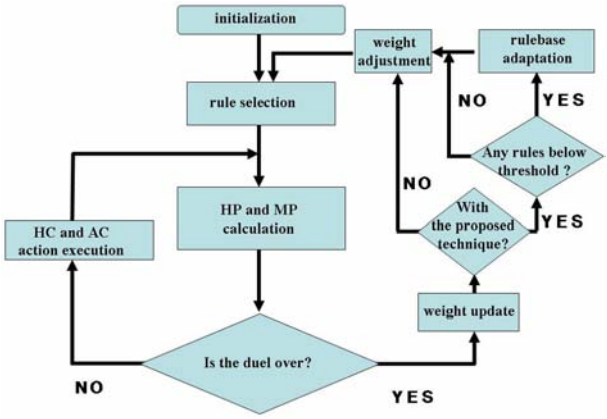


**Figure 2: Flowchart of the simulator**

4. Each rule weight in the script is updated according to the fighting result.
5. Go to 2

# 3. ONLINE ADAPTATION OF RULE BASE

In Chapter 6 of Spronck's Ph.D. Thesis [3], a procedure was presented for improving rulebase by adding to it new rules derived from a manual analysis of the result from an offline learning algorithm. However, this procedure is limited for use during game development. In addition, due to manual extraction of new rules, the reliability of the procedure varies according to the expertise of persons who perform the analysis. If some inefficient rules still remain in the rulebase, DS might not be able to catch up with high-level human players.

In this section, to complement the aforementioned procedure, we propose a technique for online adaptation of rulebase of DS. The basic idea behind our technique is that insufficient rules are replaced by randomly generated new rules not previously stored in the rulebase. The proposed technique is validated using the game simulator described below.

## 3.1 Game Simulator

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. In the current study, rulebase consists of 50 rules, and the weight of a rule is bounded to have the value between 0 and 1000. For each duel between HC and AC, the 20 highest-weight rules are selected from the rulebase to represent 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, due to having no realistic meaning, the following rules are not included in the rulebase.

- 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 $\geq 3$

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

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 the rulebase, and each rule weight is initialized to 500. A duel starts from the rule selection step, where, as mentioned earlier, the 20 highest-weight rules are selected from the rulebase 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 the rules in the script whose all four conditions hold, while an action of HC is selected and executed according to its pre-defined rules.

**Table 1: Summary of the actions available to the two characters and their relation to HP and MP in the game simulator**

| 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**

| Type\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* |
| **Type\Value** | **3** | **4** | **5** |
| 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* |

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

At the rulebase adaptation step, all rules in the rulebase whose weight is less than 20 are replaced by rules whose parameters are randomly assigned an integer value from 0 to 5 and weight is initialized to 500. Such new rules are subject to the condition that they are meaningful rules not duplicate with those currently or previously stored in the rulebase.

## 4. EVALUATION RESULTS

We validated our rulebase adaptation technique using two experiments. In the first experiment, we compared the performance of AC without the rulebase adaptation step (henceforth called ACDS) and that of AC with the rulebase adaptation step (henceforth called ACRA) when they dueled the same HC under two extreme conditions. In the second experiment, we compared their performances for many types of opponent HCs.

### 4.1 Experiment 1

We employed two conditions in this experiment. In one condition, the rulebase was initialized by manually selected 50 strong rules. In the other condition, it was initialized by manually selected 50 weak rules. The common opponent HC was hard-coded as follows:

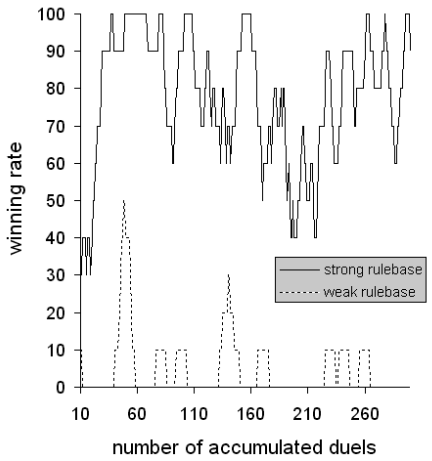


**Figure 3: The wining rate of ACDS among the recent ten duels in % for both initially strong rulebase and weak rulebase**

- 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 ACDS among the recent ten duels over the number of accumulated duels for both initially strong rulebase and weak rulebase. The winning rate of ACDS with the initially strong rulebase rises suddenly and then somewhat saturates. However, the winning rate of ACDS with the initially weak rulebase remains low. This indicates that, through update of rule weights, proper selection of rules in the script could be done for the former while this could not be achieved for the latter.

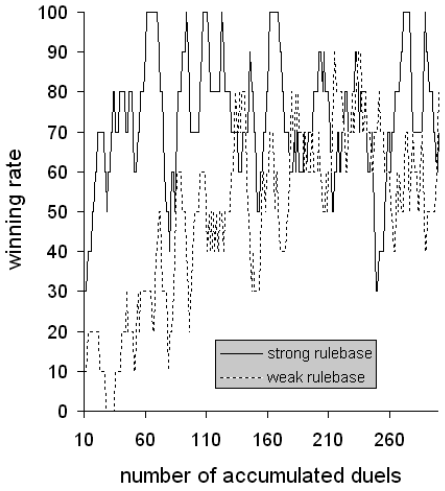


**Figure 4: The wining rate of ACRA among the recent ten duels in % for both initially strong rulebase and weak rulebase**

**Table 3: Comparisons of the obtained points between ACDS and ACRA for multiple types of HCs**

| HC Type | ACDS | ACRA |
|---|---|---|
| 1 | 63 | 37 |
| 2 | 26 | 74 |
| 3 | 21 | 79 |
| 4 | 42 | 56 |
| 5 | 43 | 55 |

Figure 4 shows the wining rate of ACRA among the recent ten duels over the number of accumulated duels for both initially strong rulebase and weak rulebase. It is clearly seen that a strong script beating the opponent HC could eventually be obtained for both conditions. However, the learning speed of ACRA is slightly lower than that of ACDS for the initially strong rulebase.

## 4.2 Experiment 2

Here, both ACDS and ACRA dueled with many types of opponent HCs. This experiment was composed of 100 experimental trials, and each experimental trial consisted of 300 duels. For each experimental trial, rulebase was initialized by a set of randomly generated meaningful rules at the initialization step. In addition, the total number of wins of ACDS against the opponent HC and that of ACRA were compared, and the one with the higher value would gain one point. Below is the summary of their common opponent HCs.

**Type-I HC** performs *HP attack type 1* for all conditions.

**Type-II HC** performs it HP attack type 3 if its MP $\geq$ 10; ortherwise, *HP attack type 1*.

**Type-III HC** performs *heal* if its MP $\geq$ 10 and its HP $<$ 50; and performs *HP attack type 3* if its MP $\geq$ 10 and its HP $\geq$ 50; otherwise, *HP attack type 1*.

**Type-IV HC** performs randomly between *HP attack type 1* and *HP attack type 2* if its MP $<$ 10; otherwise, selects the actions randomly out of 5 action candidates.

**Type-V HC** performs *HP attack type 1*, *HP attack type 2*, *HP attack type 3*, *MP attack*, and *heal* with the probability of 30%, 20%, 20%, 20%, and 10%, respectively, if its MP $\geq$ 10; otherwise, *HP attack type 1*.

Type-I HC represents a novice human player with little degree of strategy. Type-IV HC and Type-V HC represent a human player with dynamic strategy. Type-III HC is the same HC as the one used in Experiment 1.

Table 3 shows the obtained points of both ACDS and ACRA for all types of opponent HCs. It can be readily seen that the performance of ACRA is superior to that of ACDS for all opponent types, except for Type-I HC. Since Type-I HC is a relatively weak opponent, update of rule weights alone is sufficient for ACDS to quickly construct a stronger script regardless of the quality of the initial rulebase. As discussed in the previous section, for such a weak opponent, ACRA has the learning speed lower than ACDS. As a result, for this particular case, the total number of wins of ACRA tends to be less than that of ACDS in an experimental trial, leading to lower points.

# 5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a technique to complement DS in the case where rulebase is not initially well designed. The proposed technique replaces inefficient rules in rulebase by new rules. At present, new rules are randomly found from all rule candidates that have never been assigned to the rulebase. As shown in our experiments, DS with the proposed technique is superior to DS without the proposed technique, especially when the quality of initial rulebase is poor. However, due to its random finding of new rules, the former has slightly lower learning speed than the latter for proper initial rulebase. The problem is left here as our future work.

# 6. REFERENCES

[1] D. M. Bourg and G. Seemann. *AI For Game Developers.* Oreilly & Associates Inc, 2004.

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

[3] P. Spronck. *Adaptive Game AI.* Ph.D. Thesis, Maastricht University Press, Maastricht, The Netherlands, 2005.

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