DIFFICULTY SCALING OF GAME AI

Pieter Spronck, Ida Sprinkhuizen-Kuyper and Eric Postma
Universiteit Maastricht / IKAT
P.O. Box 616, NL-6200 MD Maastricht, The Netherlands
E-mail: p.spronck@cs.unimaas.nl

KEYWORDS
Gaming, game AI, difficulty scaling, artificial intelligence, machine learning, unsupervised online learning.

ABSTRACT

"Difficulty scaling" is the automatic adaptation of a game, to adapt the challenge a game poses to a human player. In general, a game of which the challenge level matches the skill of the human player (i.e., an "even game") is experienced as more entertaining than a game that is either too easy or too hard. In practice, when difficulty scaling is implemented in a game, it only adapts a few parameters. Even state-of-the-art games do not apply it to game AI, i.e., to the behaviour of computer-controlled opponents in a game. In prior work, we designed a novel online-learning technique called "dynamic scripting", that is able to automatically optimise game AI during game-play. In the present paper, we research to what extent dynamic scripting can be used to adapt game AI in order to elicit an even game. We investigate three difficulty-scaling enhancements to the dynamic scripting technique, namely (1) high-fitness penalising, (2) weight clipping, and (3) top culling. Experimental results indicate that top culling is particularly successful in creating an even game. We conclude that dynamic scripting, using top culling, can enhance the entertainment value of games by scaling the difficulty level of the game AI to the playing skill of the human player.

INTRODUCTION

The quality of commercial computer games (henceforth called "games") is directly related to their entertainment value (Tozour 2002a). The general dissatisfaction of game players with the current quality of artificial intelligence that controls opponents (so-called "game AI") makes them prefer human-controlled opponents (Schaeffer 2001). Improving the quality of game AI is desired in case human-controlled opponents are not available.

Many researchers and game developers consider game AI, in general, to be entertaining when it is difficult to defeat (Buro 2003). Although for strong players that may be true, novice players will not enjoy being overwhelmed by the computer. For novice players, a game is most entertaining when the game is challenging but beatable (Scott 2002). "Difficulty scaling" is the automatic adaptation of a game, to set the challenge the game poses to a human player. When applied to game AI, difficulty scaling usually aims at achieving an "even game", i.e., a game wherein the playing strength of the computer and the human player match.

In complex games, such as Computer RolePlaying Games (CRPGs), the incorporation of advanced game AI is difficult. For these complex games most game-AI developers resort to scripts, i.e., lists of rules that are executed sequentially (Tozour 2002b). AI scripts are generally static, and thus cannot adapt the level of difficulty exhibited by the game AI to appeal to both novice and experienced human players.

In our research, we apply machine-learning techniques to improve the quality of game AI. When machine learning is used to allow opponents to adapt while the game is played, this is referred to as "online learning". Online learning allows the opponents to automatically repair weaknesses in their scripts that are exploited by the human player, and to adapt to changes in human player tactics. Unsupervised online learning is widely disregarded by commercial game developers (Woodcock 2002), even though it has been shown to be feasible for games (Demasi and Cruz 2002).

In prior work, we designed a novel technique called "dynamic scripting" that realises online adaptation of scripted game AI, in particular for complex games (Spronck, Sprinkhuizen-Kuyper, and Postma 2004a). In its original form, dynamic scripting optimises the game-playing strength of game AI. The present research investigates three different enhancements to the dynamic scripting technique that allow it to scale the difficulty level of the game AI to create an even game, namely (1) high-fitness penalising, (2) weight clipping, and (3) top culling.

The outline of the remainder of the paper is as follows. First, it describes dynamic scripting and the three difficulty-scaling enhancements. Then, the results obtained by applying the enhancements to dynamic scripting are presented and discussed. Finally, the paper concludes and points at future work.

DYNAMIC SCRIPTING

Online learning of game AI entails that the AI is adapted while the game is being played. In this section we present dynamic scripting as a technique that is designed specifically for this purpose. A detailed exposition of dynamic scripting is provided by Spronck *et al.* (2004a).

Dynamic scripting is an unsupervised online learning technique for games, that is computationally fast, effective, robust, and efficient (Spronck *et al.* 2004a). It maintains several rulebases, one for each opponent type in the game. The rules in the rulebases are manually designed using domain-specific knowledge. When a new opponent is generated, the rules that comprise the script controlling the opponent are extracted from the rulebase corresponding to the opponent type. The probability that a rule is selected for a script is proportional to the value of the weight associated with the rule. The rulebase adapts by changing the weight values to reflect the success or failure rate of the associated rules in scripts. A priority mechanism can be used to determine rule precedence. The dynamic scripting process is illustrated in figure 1 in the context of a game.

The learning mechanism of dynamic scripting is inspired by reinforcement-learning techniques (Russell and Norvig 2002). It has been adapted for use in games because "regular" reinforcement-learning techniques are not sufficiently efficient for online learning in games (Manslow

2002). In the dynamic-scripting approach, learning proceeds as follows. Upon completion of an encounter, the weights of the rules employed during the encounter are adapted depending on their contribution to the outcome. Rules that lead to success are rewarded with a weight increase, whereas rules that lead to failure are punished with a weight decrease. The remaining rules are updated so that the total of all weights in the rulebase remains unchanged.

Weight values are bounded by a range $[W_{\min}, W_{\max}]$. The size of the weight change depends on how well, or how badly, a team member behaved during the encounter. It is determined by a fitness function that rates a team member's performance as a number in the range [0,1]. The fitness function is composed of four indicators of playing strength, namely (1) whether the member's team won or lost, (2) whether the member died or survived, (3) the member's remaining health, and (4) the amount of damage done to the member's enemies. The new weight value is calculated as $W+\Delta W$, where W is the original weight value, and the weight adjustment ΔW is expressed by the following formula:

$$\Delta W = \begin{cases} -P_{\max} \cdot \dfrac{b-F}{b} & \{F < b\} \\ R_{\max} \cdot \dfrac{F-b}{1-b} & \{F \geq b\} \end{cases} \tag{1}$$

where F is the fitness, b is the break-even value, and $R_{\max}$ and $P_{\max}$ are the maximum reward and maximum penalty respectively. The left graph in figure 2 displays the weight adjustment as a function of the fitness F.

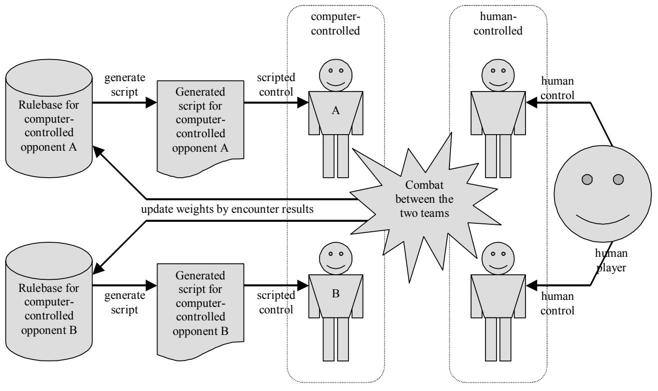
computer-controlled
human-controlled
generate script
scripted control
human control
Rulebase for computer-controlled opponent A
Generated script for computer-controlled opponent A
A
Combat between the two teams
update weights by encounter results
human player
Rulebase for computer-controlled opponent B
generate script
Generated script for computer-controlled opponent B
scripted control
B
human control

Figure 1: The dynamic scripting process. For each computer-controlled opponent a rulebase generates a new script at the start of an encounter. After an encounter is over, the weights in the rulebase are adapted to reflect the results of the fight.

DIFICULTY SCALING

Many games provide a "difficulty setting", i.e., a discrete value that determines how difficult the game will be. The purpose of a difficulty setting is to allow both novice and experienced players to enjoy the appropriate challenge the game offers. Usually the parameter influences opponents' strength and health. Very rarely the parameter influences opponents' tactics. Consequently, even on a "hard" difficulty setting, opponents exhibit inferior behaviour, despite their high physical strength and health.

This section describes how dynamic scripting can be used to create new opponent tactics while scaling the difficulty level of the game AI to the experience level of the human player. Specifically, it describes three different enhancements to the dynamic scripting technique that let opponents learn how to play an even game, namely (1) high-fitness penalising, (2) weight clipping, and (3) top culling.

High-Fitness Penalising

The weight adjustment expressed in formula (1) gives rewards proportional to the fitness value: the higher the fitness, the higher the reward. To elicit mediocre instead of optimal behaviour, the weight adjustment can be changed to give highest rewards to mediocre fitness values, and lower rewards or even penalties to high fitness values. With high-fitness penalising weight adjustment is expressed by formula (1), where F is replaced by F' defined as follows:

$$F' = \begin{cases} \dfrac{F}{p} & \{F \leq p\} \\ \dfrac{(1-F)}{p} & \{F > p\} \end{cases} \tag{2}$$

where F is the calculated fitness value and p is the reward-peak value, i.e., the fitness value that should get the highest reward. The higher p is set, the more effective opponent behaviour will be. Figure 2 illustrates the weight adjustment as a function of F, with (left) and without (right) high-fitness penalising.

Since the optimal value for p depends on the tactic that the human player uses, we decided to let the value of p adapt to the perceived difficulty level of a game, as follows. Initially p starts at a value p_{init}. After every fight that is lost by the computer, p is increased by a small amount p_{inc}, up to a predefined maximum value $p_{\max}$. After every fight that is won by the computer, p is decreased by a small amount p_{dec}, down to a predefined minimum value $p_{\min}$.

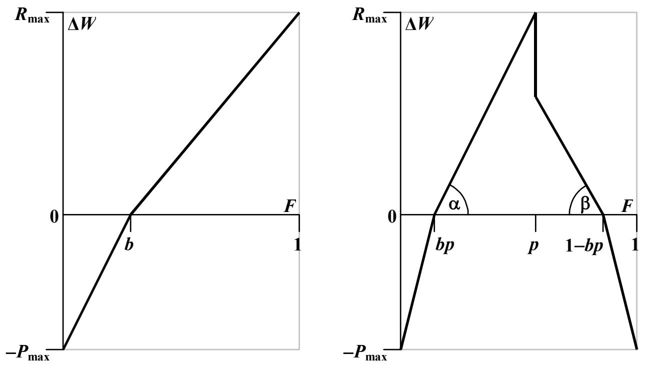

Figure 2: Graphs of the original weight-adjustment formula (left) and the high-fitness-penalising weight-adjustment formula (right). Angles α and β are equal.

Weight Clipping

The maximum weight value $W_{\max}$ determines the maximum level of optimisation a learned tactic can achieve. A high value for $W_{\max}$ allows the weights to grow to large values, so that after a while the most effective rules will almost always be selected. This will result in scripts that are close to optimal. A low value for $W_{\max}$ restricts weights in their growth. This enforces a high diversity in generated scripts, most of which will not be optimal.

Weight clipping automatically changes the value of $W_{\max}$, with the intent to enforce an even game. It aims at having a low value for $W_{\max}$ when the computer wins often, and high value for $W_{\max}$ when the computer loses often. The implementation is as follows. After the computer won a fight, $W_{\max}$ is decreased by W_{dec} per cent (but not lower than the initial weight value W_{init}). After the computer lost a fight, $W_{\max}$ is increased by W_{inc} per cent.

Figure 3 illustrates the weight-clipping process and parameters. The shaded bars denote weight values for arbitrary rules on the horizontal axis. Before the weight adjustment, $W_{\max}$ changes by W_{inc} or W_{dec} per cent, depending on the outcome of the fight. After the weight adjustment, in figure 3 the weight value for rule 4 is too low, and will be increased to $W_{\min}$ (arrow 'a'), while the weight value for rule 2 is too high, and will be decreased to $W_{\max}$ (arrow 'b').

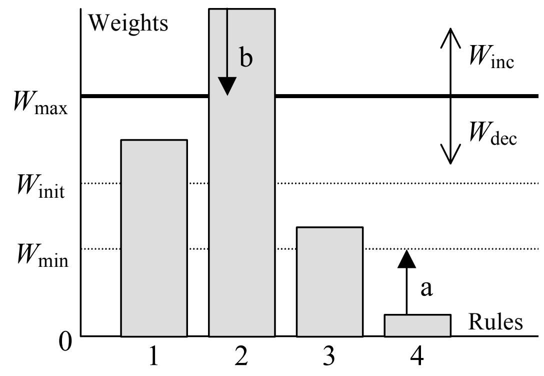

Figure 3: Weight clipping and top culling process and parameters.

Top Culling

Top culling is similar to weight clipping. It employs the same adaptation mechanism for the value of $W_{\max}$. The difference is that top culling allows weights to grow beyond the value of $W_{\max}$. However, rules with a weight greater than $W_{\max}$ will not be selected for a generated script. Consequently, when the computer-controlled opponents win often, the most effective rules will have weights that exceed $W_{\max}$, and cannot be selected, and thus the opponents will use weak tactics. Alternatively, when the computer-controlled opponents lose often, rules with high weights will become selectable (again), and the opponents will use strong tactics.

In figure 3, contrary to weight clipping, top culling will leave the value of rule 2 unchanged (the action represented by arrow (b) will not be performed). However, rule 2 will be unavailable for selection, because its value exceeds $W_{\max}$.

EXPERIMENTS

To evaluate the effect of the three difficulty-scaling enhancements to dynamic scripting, we employed a simulation of an encounter of two teams in a complex CRPG, closely resembling the popular BALDUR'S GATE games. We used this environment in earlier research to demonstrate the efficiency of dynamic scripting (Spronck *et al.* 2004a), and to test different measures to reduce the number of outliers that dynamic scripting occasionally generates (Spronck, Sprinkhuizen-Kuyper and Postma 2004b). Our evaluation experiments aimed at assessing the performance of a team controlled by the dynamic scripting technique using a difficulty-scaling enhancement, against a team controlled by static scripts. If the difficulty-scaling enhancements work as intended, dynamic scripting will balance the game so that the number of wins of the dynamic team is roughly equal to the number of losses. In the simulation, we pitted the dynamic team against a static team that uses one of five, manually designed, basic tactics (named "offensive", "disabling", "cursing", "defensive" and "novice"), or one of three composite tactics (named "random party", "random character" and "consecutive party").

Of the eight static team's tactics the most interesting in the present context is the "novice" tactic. This tactic resembles the playing style of a novice BALDUR'S GATE player. While it normally will not be defeated by arbitrarily picking rules from the rulebase, many different tactics exist that can be employed to defeat it, which the dynamic team will quickly discover. Without difficulty-scaling, the dynamic team's number of wins will greatly exceed its losses.

In our experiments we initialised $W_{\max}$=2000. We set W_{init}=100, $W_{\min}$=0, W_{inc}=10%, W_{dec}=10%, p_{init}=0.7, $p_{\min}$=0.65, $p_{\max}$=0.75, p_{inc}=0.01, p_{dec}=0.01, $R_{\max}$=100, $P_{\max}$=100, and b=0.3. We employed the same fitness function as in previous research (Spronck *et al.* 2004a), andd dynamic scripting with fitness propagation fallback (Spronck *et al.* 2004b).

For each of the tactics, we ran 100 tests in which dynamic scripting was enhanced with each of the three difficulty-scaling enhancements, and, for comparison, also without difficulty-scaling enhancements ("optimisation"). Each test consisted of a sequence of 150 encounters between the dynamic team and the static team. Because in each of the tests the dynamic scripting process starts with a rulebase with all weights equal, the first 50 encounters were used for finding a balance of well-performing weights. We recorded the number of wins of the dynamic team for the last 100 encounters. The results of these tests are displayed in table 1. Histograms for the tests with the "novice" tactic are displayed in figure 4.

To be recognised as an even game, we decided that the average number of wins over all tests must be close to 50. To take into account random fluctuations, in this context "close to 50" means "within the range [45,55]". In table 1, all cell values indicating an even game are marked in bold font. From table 1 the following four results can be derived.

First, optimisation (dynamic scripting without a difficulty-scaling enhancement) results in wins significantly exceeding losses for all tactics except for the "consecutive party" tactic (with a reliability > 99.9%; Cohen 1995). The "consecutive party" tactic is the most difficult tactic to defeat (Spronck *et al.* 2004a). Note that the fact that, on average, dynamic scripting plays an even game against the "consecutive party" tactic is not because it is unable to consistently defeat this tactic, but because dynamic scripting continues learning after it has reached an optimum. Therefore, it can "forget" what it previously learned, especially against an optimal tactic like the "consecutive party" tactic.

Second, high-fitness penalising performs considerably worse than the other two enhancements. It cannot achieve an even game against six of the eight tactics.

Tactic	Optimisation		Fitness Penalising		Weight Clipping		Top Culling	
	Wins	St.dev.	Wins	St.dev.	Wins	St.dev.	Wins	St.dev.
Offensive	61.2	16.4	**46.0**	15.1	**50.6**	9.4	**46.3**	7.5
Disabling	86.3	10.4	56.6	8.8	67.8	4.5	**52.2**	3.9
Cursing	56.2	11.7	42.8	9.9	**48.4**	6.9	**46.4**	5.6
Defensive	66.1	11.9	39.7	8.2	**52.7**	4.2	**49.2**	3.6
Novice	75.1	13.3	**54.2**	13.3	**53.0**	5.4	**49.8**	3.4
Random Party	55.8	11.3	37.7	6.5	**50.0**	6.9	**47.4**	5.1
Random Character	58.8	9.7	44.0	8.6	**51.8**	5.9	**48.8**	4.1
Consecutive Party	**51.1**	11.8	34.4	8.8	**48.7**	7.7	**45.0**	7.3

Table 1: Experimental results of testing the three difficulty-scaling enhancements to dynamic scripting on eight different tactics, compared to dynamic-scripting optimisation. For each combination the table shows the average number of wins over 100 tests, and the associated standard deviation. The cells marked with a bold font indicate the enhancement to be successful in forcing an even game against the associated tactic.

Third, weight clipping is successful in enforcing an even game against seven out of eight tactics. It does not succeed against the "disabling" tactic. This is caused by the fact that the "disabling" tactic is so easy to defeat, that even a rulebase with all weights equal will, on average, generate a script that defeats this tactic. Weight clipping can never generate a rulebase worse than "all weights equal".

Fourth, top culling is successful in enforcing an even game against all eight tactics.

From the histograms in figure 4 we derive the following result. While all three difficulty-scaling enhancements manage to, on average, enforce an even game against the "novice" tactic, the number of wins in each of the tests is much more "spread out" for the high-fitness-penalising enhancement than for the other two enhancements. This indicates that the high-fitness penalising results in a higher variance of the distribution of won games than the other two enhancements. The top-culling enhancement seems to yield the lowest variance. This is confirmed by an approximate randomisation test (Cohen 1995, section 6.5), which shows that against the "novice" tactic, the variance achieved with top culling is significantly lower than with the other two enhancements (reliability > 99.9%). We observed similar distributions of won games against the other tactics, except that against some of the stronger tactics, a few exceptional outliers occurred with a significantly lower number of won games. The rare outliers were caused by dynamic scripting, occasionally needing more than the first 50 encounters to find well-performing weights against a strong static tactic.

We further validated the results achieved with top culling, by implementing dynamic scripting with the top-culling enhancement in a state-of-the-art computer game, NEVERWINTER NIGHTS (version 1.61), testing it against the game AI implemented by the game developers, with the same experimental procedure as used in the simulation environment. Ten optimisation tests resulted in an average number of wins of 79.4 out of 100, with a standard deviation of 12.7. Ten tests with the top-culling enhancement resulted in an average number of wins of 49.8 out of 100, with a standard deviation of 3.4. Therefore, our simulation results are supported by the NEVERWINTER NIGHTS tests.

DISCUSSION

Of the three different difficulty-scaling enhancements we conclude the top-culling enhancement to be the best choice. It has the following three advantages: (1) it yields results with a very low variance, (2) it is easily implemented, and (3) of the three enhancements, it is the only one that manages to force an even game against inferior tactics.

Obviously, the worst choice is the high-fitness-penalising enhancement. In an attempt to improve high-fitness penalising, we performed some tests with different ranges and adaptation values for the reward-peak value *p*, but these worsened the results. However, we cannot rule out the possibility that with a different fitness function high-fitness penalising will give better results.

An additional possibility with weight clipping and top culling is that they can be used to set a desired win-loss ratio, simply by changing the rates with which the value of W_{max} fluctuates. For instance, by using top culling with W_{dec}=30% instead of 10%, leaving all other parameters the same, after 100 tests against the "novice" tactic, we derived an average number of wins of 35.0 with a standard deviation of 5.6.

In previous research we concluded that dynamic scripting is suitable to be applied in real commercial games to automatically optimise tactics (Spronck *et al.* 2004a). With a difficulty-scaling enhancement, the usefulness of dynamic scripting is improved significantly, since it can now also be used to scale the difficulty level of a game to the experience level of the human player. Notwithstanding this, a difficulty-scaling enhancement should be an optional feature in a game, that can be turned off by the player, for the following two reasons: (1) when confronted with an experienced player, the learning process should aim for optimal tactics without interference from a difficulty-scaling enhancement, and (2) some players will feel that attempts by the computer to force an even game diminishes their accomplishment of defeating the game, so they may prefer not to use it.

CONCLUSIONS AND FUTURE WORK

In this paper we proposed three different enhancements to the dynamic scripting technique that allow scaling the difficulty level of game AI, namely (1) high-fitness penalising, (2) weight clipping, and (3) top culling. From our experiments we conclude that a difficulty-scaling enhancement to dynamic scripting can be used to let the learning process scale the difficulty level of the tactics employed by the computer-controlled opponents to match the game-playing skills of the human player (i.e., to force an even game). Of the three difficulty-scaling enhancements tested, high-fitness penalising was, in general, unsuccessful, but the other two performed well. Top culling gave the best results. We also discovered that both weight clipping and top culling, besides forcing an even game, can be used to set a different win-loss ratio, by tuning a single parameter. We conclude that dynamic scripting, using top culling, can enhance the entertainment value of games by scaling the difficulty level of the game AI to the playing skill of the human player.

In future work, we intend to apply dynamic scripting, including difficulty scaling, in other game types than CRPGs. We will also investigate whether offline machine learning techniques, which can be very effective in designing tactics (Spronck, Sprinkhuizen-Kuyper and Postma 2003), can be used to "invent" completely new rules for the dynamic scripting rulebase. Finally, we aim to investigate the effectiveness and entertainment value of dynamic scripting in games played against actual human players. While such a study requires many subjects and a careful experimental design, the game-play experiences of human players are important to convince game developers to adopt dynamic scripting in their games.

REFERENCES

Buro, M. 2003. "RTS Games as a Test-Bed for Real-Time AI Research." *Proceedings of the 7th Joint Conference on Information Science* (eds. K. Chen *et al.*), pp. 481–484.

Cohen, P.R. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA.

Demasi, P. and A.J. de O. Cruz. 2002. "Online Coevolution for Action Games." *International Journal of Intelligent Games and Simulation* (eds. N.E. Gough and Q.H. Mehdi), Vol. 2, No. 2. University of Wolverhampton and EUROSIS , pp. 80–88.

Manslow, J. 2002. "Learning and Adaptation." *AI Game Programming Wisdom* (ed. S. Rabin). Charles River Media, pp. 557–566.

Russell, S. and P. Norvig. 2002. *Artificial Intelligence: A Modern Approach*, Second Edition. Prentice Hall, Englewood Cliffs, NJ.

Schaeffer, J. 2001. "A Gamut of Games." *AI Magazine*, Vol. 22 No. 3, pp. 29–46.

Scott, B. 2002. "The Illusion of Intelligence." *AI Game Programming Wisdom* (ed. S. Rabin). Charles River Media, pp. 16–20.

Spronck, P., I. Sprinkhuizen-Kuyper and E. Postma. 2003. "Improving Opponent Intelligence Through Offline Evolutionary Learning." *International Journal of Intelligent Games and Simulation* (eds. N.E. Gough and Q.H. Mehdi), Vol. 2, No. 1. University of Wolverhampton and EUROSIS, pp. 20–27.

Spronck, P., I. Sprinkhuizen-Kuyper and E. Postma. 2004a. "Online Adaptation of Game Opponent AI with Dynamic Scripting." *International Journal of Intelligent Games and Simulation* (eds. N.E. Gough and Q.H. Mehdi), Vol. 3, No. 1. University of Wolverhampton and EUROSIS, pp. 45–53.

Spronck, P., I. Sprinkhuizen-Kuyper and E. Postma. 2004b. "Enhancing the Performance of Dynamic Scripting in Computer Games." *Entertainment Computing – ICEC 2004* (ed. M. Rauterberg). LNCS 3166, Springer-Verlag, Berlin, Germany, pp. 296–307.

Tozour, P. 2002a. "The Evolution of Game AI." *AI Game Programming Wisdom* (ed. S. Rabin). Charles River Media, pp. 3–15.

Tozour, P. 2002b. "The Perils of AI Scripting." *AI Game Programming Wisdom* (ed. S. Rabin). Charles River Media, pp. 541–547.

Woodcock, S. 2000. "Game AI: The State of the Industry." *Game Developer Magazine,* August 2000.

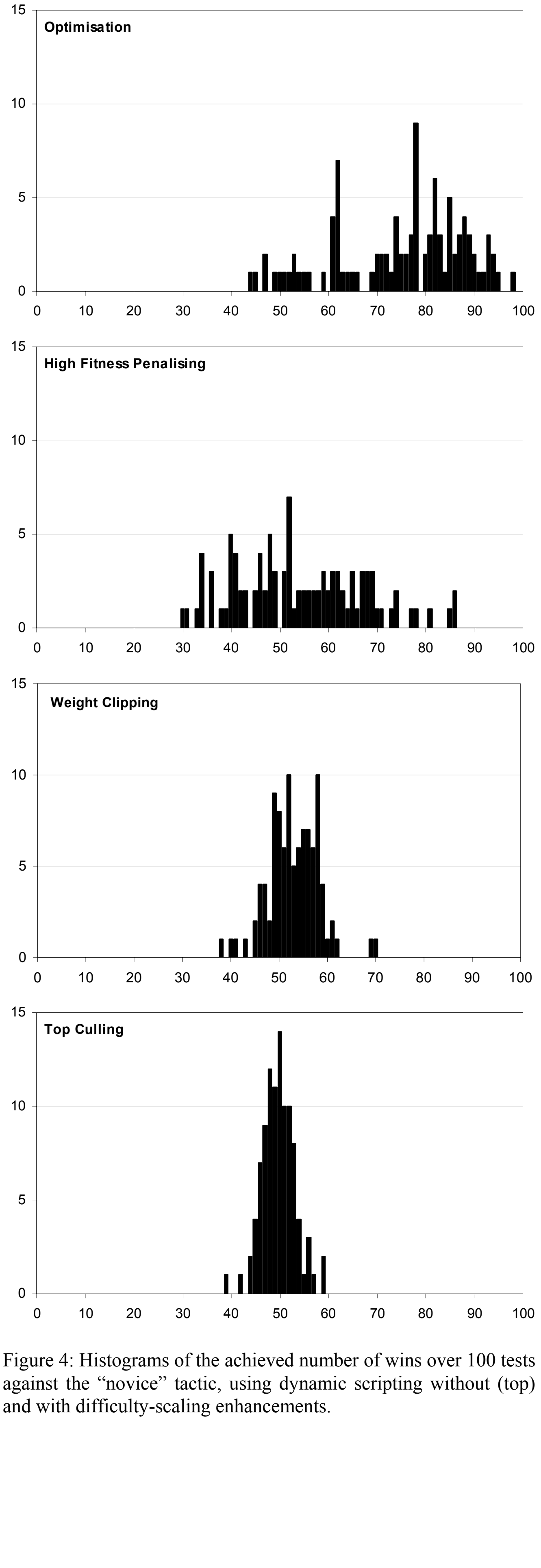

Figure 4: Histograms of the achieved number of wins over 100 tests against the "novice" tactic, using dynamic scripting without (top) and with difficulty-scaling enhancements.