Challenging AI: Evaluating the Effect of MCTS-Driven Dynamic Difficulty Adjustment on Player Enjoyment

Simon Demediuk
RMIT University
Melbourne, Australia
simon.demediuk@rmit.edu.au

Marco Tamassia
RMIT University
Melbourne, Australia
marco.tamassia@rmit.edu.au

Xiaodong Li
RMIT University
Melbourne, Australia
xiaodong.li@rmit.edu.au

William L. Raffe
University of Technology (UTS)
Sydney, Australia
william.raffe@uts.edu.au

ABSTRACT

Providing a challenging Artificial Intelligent opponent is an important aspect of making video games enjoyable and immersive. A game that is too easy, or conversely too hard may frustrate or bore players. Dynamic Difficulty Adjustment is a method that aims at improving the traditional methods of difficulty selection, by providing an opponent that tailors the challenge it presents to players such that it is at an optimal level for them. This research presents a player evaluation of three different Dynamic Difficulty Adjustment approaches using Monte Carlo Tree Search and measures their impact on player enjoyment, realism and perceived level of difficulty. In particular, it investigates the effect that different win/loss ratios, employed by Dynamic Difficulty Adjustment, have on player enjoyment.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**; • **Human-centered computing** → Human computer interaction (HCI).

KEYWORDS

Dynamic Difficulty Adjustment, Monte Carlo Tree Search, Artificial intelligence, Video Games

ACM Reference Format:
Simon Demediuk, Marco Tamassia, Xiaodong Li, and William L. Raffe. 2019. Challenging AI: Evaluating the Effect of MCTS-Driven Dynamic Difficulty Adjustment on Player Enjoyment. In *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '19), January 29–31, 2019, Sydney, NSW, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3290688.3290748

ACSW '19, January 29–31, 2019, Sydney, NSW, Australia

ACM ISBN 978-1-4503-6603-8/19/01…$15.00
https://doi.org/10.1145/3290688.3290748

1 INTRODUCTION

The video games industry reached over $100 Billion in revenue in 2017 [24]. In order to break into this competitive market, developers need to ensure their games present the right level of challenge to players in order to provide an enjoyable gaming experience.

There are a number of different ways in which current video games present a level of challenge to the player. One way to do this is by steadily increasing the difficulty of the levels of the game as the player progresses. Another approach that game developers use is to allow for the player to chose the difficulty of the game by selecting it directly before the game begins i.e "easy", "medium" or "hard". While originally this only allowed for a few selections, game developers have since developed methods to allow for a larger variety of difficulty settings.

Both approaches, however, have a number of limitations. In the case of selecting a difficulty before the game has begun the player needs to have some assumptions about their current skill level at the game in question, as well as assumptions about how the developers assign the difficulty to the game. If these assumptions are not correct, this can lead to some dissatisfaction with the game, and depending on the game in question it may not be easy to switch this difficulty once progress has been made by the player. A fixed difficulty system also may not incorporate the skill level progression of the player as they get better at the game.

Another limitation of both the increased level system and the fixed difficulties is the jump between each, i.e going from one level to another or from one difficulty setting to another. This jump needs to be carefully calibrated, as a too big a jump will provide too much of a challenge to players, too little a jump and there may be no noticeable difference for the players. In both cases, the player may become disengaged and frustrated with the game. This then leads to another limitation which is, game developers need to group all players into a very small number of different skill levels, and this small grouping is likely to only satisfy the mean number of players.

In an early attempt to address these issues, Dynamic Difficulty Adjustment [16] was introduced. Dynamic Difficulty Adjustment is a method that allows for the level of challenge the game presents to the player to be at the right level for them, and to adjust as their skill level changes. This also allows for players to enter a state of 'Flow' [11] in which the player becomes immersed and focused on

the game [10]. The problem then becomes what is the right level of challenge to present to the player.

Current Dynamic Difficulty Adjustment systems tend to aim for a win/loss ratio for the player at 50% [3], and this is for good reason. This level of challenge provides a fair game to the player. A fair game is a concept from ranking systems such as Elo [13], wherein a competitive game, a fair game represents either player being equally likely to win. In earlier works Malone [21] suggests that a fair game is motivating because it provides a challenge at an appropriate difficulty level. Coupled with the findings from Yannakakis et al. [28] that found empirically that a game's entertainment factor was at the highest when the game was set at an appropriate level of challenge for the player. Targeting a win/loss ratio for players at 50% makes sense for Dynamic Difficulty Adjustment systems for general video games.

In this paper, we present three Dynamic Difficulty Adjustment approaches that employ MCTS and test these algorithms in a 2D fighting game to understand the enjoyment, perceived difficulty and realism aspects of the different methods in comparison to an AI that is too difficult for the players. This work provides insights into the understanding of what makes a challenging AI opponent. In particular, the effect that different win/loss ratios, which the Dynamic Difficulty Adjustment agents target, have on the enjoyment of the game for player.

2 RELATED WORK

The concept of flow in games is crucial for ensuring that a player is immersed in the game. Cowley et al. [10] has applied Csikszentmihalyi's "Flow theory" [11] to video games. Cowley states that the player will continue to play and be interested in a game, so long as they are immersed in it. Sweetser et al. [26] also worked with Csikszentmihalyi's "Flow theory" to develop a model for evaluating games called GameFlow. It was used to predict the success of a game, by evaluating the game with regard to the following attributes: concentration, challenge, player skill, control, clear goals, feedback, immersion and social interaction. Although the work conducted by Sweetser et al. refers primarily to the concept of immersion, rather than challenge, definitions of immersion [8, 18] include challenge as a fundamental part of what makes a game immersive. Providing a challenging artificial opponent for a competitive game is required to make the game enjoyable [1].

One way to provide a challenging opponent is to use a technique known as Dynamic Difficulty Adjustment. An early implementation of this is a system called Hamlet [16], which attempts to adjust the difficulty of games during gameplay by controlling supply, demand, and strength of the in-game inventory. Once a player is found to be struggling in one area of the game, the system is able to modify the game environment in a number of different ways: changing the availability of in-game items, varying the strength of the player's weapons, and changing the gameplay of the enemies.

Yannakakis et al. [29] developed a Dynamic Difficulty Adjustment agent through the implementation of an adaptive AI system that used Bayesian networks to model the individual player. This player model was then used by the adaptive system to modify the artificial opponent. Avery et al. [5], and Bakkes et al. [6] also developed adaptive systems that can tailor the artificial opponent to the individual player. Avery et al. used co-evolutionary techniques, where Bakkes et al. employed a form of supervised learning. Both methods produce a suitable adaptive system, despite the fact that they used different metrics to determine whether the artificial opponent was suitable for the individual. Whilst this is not an extensive review of all the different approaches and techniques that have been published over the years on DDA, these papers provide excellent preliminary background information.

More recently there has been a push into Dynamic Difficulty Adjustment agents that use Monte Carlo Tree Search [15, 20, 23, 30]. As Monte Carlo Tree Search has been improved over the years and used successfully in complex games such as Go [14]. These MCTS approaches focus on limitating computational aspects of MCTS either through hardware, or expansion and simulation limitations. Whilst these different approaches can be successful at producing DDA agents in some game genres. Due to the limited computational time of some real-time games these approaches can be less effective, espically in cases where MCTS has very little time to process. In Section 5 we introduce a different approach to DDA agents using MCTS. The primary difference being our agents do not limit MCTS in anyway, rather we change how MCTS builds it search tree and the action selection policy MCTS uses. This allows for the maximum allotted time for MCTS to always be utilised, finding the most appropriate action. This approach is not limited to real-time games, but can also easily applied to any MCTS based AI opponent.

3 BACKGROUND

Games with large state spaces, such as Chess and Go, result in search spaces of prohibitive size. To deal with games of this kind, stochastic tree-search techniques are used, which are called "Monte Carlo" methods [22]. Monte Carlo methods do not completely explore a sub-tree, which may take an infinite amount of time; instead, they content themselves with randomly sampling parts of such subtrees and using the average of such sample as an estimate for the value of the entire sub-tree. Notice that this is only possible if a simulator of the game is available, so that game simulations can be run.

There are few different ways of using this technique, such as:

- assuming the player also has a model of each opponent player, which would help produce accurate simulations, but is not realistic;
- assuming the other players perform random actions, which is not very realistic since good players will choose better-than-average actions;
- assuming that other players are perfectly rational and therefore run minimax (or its appropriate variation), which has the disadvantage of requiring a complete tree exploration, nullifying the advantages of Monte Carlo techniques.

A hybrid approach is to use the same principle of minimax until a certain tree depth and only for deeper nodes using stochastic simulation to estimate values. This has the advantage of not requiring the exploration of the full tree and provides a heuristic that does not require domain expertise, making it a general technique.

Monte Carlo Tree-Search [9] further optimizes the minimax part of said hybrid approach by using *asymmetrical exploration* (see Figure 1). This is realized by progressively building the tree, and exploring only sub-trees that look more promising according to

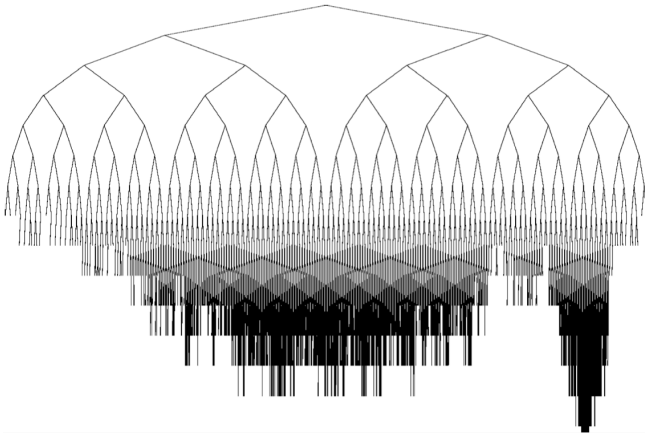

Figure 1: Example of asymmetric tree [9].

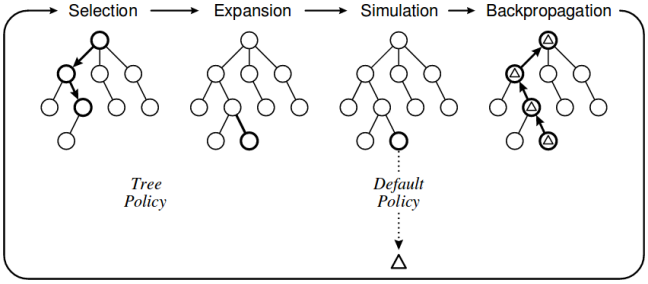

Figure 2: One iteration of the general MCTS approach [9]

the current estimates. In more detail, this is done by repeating four steps (see Figure 2):

- *Selection*: the tree is descended from the root along the path of the most promising actions according to the current estimates until a leaf is reached.
- *Expansion*: if the leaf reached is above a certain depth threshold, it is expanded by adding all possible actions as children.
- *Simulation*: the actions indicated by the path root–leaf are run in the simulator, followed by a sequence of random actions – up until the game ends or up to a certain length.
- *Back-propagation*: the value of the outcome of the simulation (or a heuristic, if the simulation stopped before the end of the game) is used to update the statistics of the nodes in the path root–leaf.

This results in a tree that deepens over time and as such improve the estimates of the values of its nodes with time. This gives MCTS a useful advantage over minimax: it can run during an allotted time and stop when an answer is needed, therefore maximizing the use of time.

MCTS, as explained, has a problem: depending on how the "promising node" is chosen, it may perform quite poorly. If the AI naively chooses a promising node the node with the highest value, it risks re-visiting the same path over and over, thereby only improving the estimates of the nodes along said path. However, because initial estimates are just samples, there is a risk of re-exploring a sub-optimal path while neglecting the optimal one. This can happen if the initial value optimal path is set according to an unlucky sample (e.g., one of the random actions for the player is a particularly poor choice, which would make the sample look worse than it actually is), or if the initial value of another path is set according to a lucky sample (e.g., one of the random actions for the opponent is a particularly poor choice, making the sample look better than it actually is).

A better way of choosing a "promising node" is to use UCT [19]. UCT stands for UCB1 applied to Trees, where UCB1 stands for Upper Confidence Bound [4]. UCB1 minimises a measure called "regret" in "bandits", which can be thought as depth-1 trees. Regret is the difference between the reward that could be earned by choosing optimally and the reward that was actually earned.

$$R_N = \mu^* n - \mu_j \sum_{i=1}^{K} E\left[T_j(n)\right] \tag{1}$$

where μ^* is the best possible expected reward, $E\left[T_j(n)\right]$ denotes the expected number of plays for action j and K is the number of actions.

This is implemented by adding to the value of a node a number that increases the more sibling nodes are visited and decreases the more the node itself is visited. The UCB1 formula is as follows:

$$UCB1 = \overline{X}_j + \sqrt{\frac{2 \ln n}{n_j}} \tag{2}$$

where $\overline{X}_j$ is the average reward from action j, n_j is the number of times action j was tried and n is the overall number of plays so far.

UCT generalises UCB1 to trees, where the behaviour of UCB1 in deeper, recently created nodes causes a drift in the probability distribution of rewards. UCT is proved to also minimise regret and minimally changes the UCB1 formula:

$$UCT = \overline{X}_j + 2C_p \sqrt{\frac{\ln n}{n_j}} \tag{3}$$

where $\overline{X}_j$ is the average reward from action j, n_j is the number of times action j was tried at the node, n is the sum of n_j for all actions and $C_P > 0$ is a constant.

4 FIGHTINGICE

To investigate the enjoyment of different Dynamic Difficulty Adjustment agents we chose to use a 2D fighting game, FightingICE[17]. FightingICE is a game developed primarily for research purposes in this genre. A 2D fighting game is one in which two opposing players fight each other in a 2-dimensional arena. Famous games in this genre are Street Fighter[27] and Mortal Kombat[7]. In this game, players attempt to reduce the opponent's total health to zero before their health is reduced to zero or the time runs out. In the latter case, the player with the most health wins. To achieve this players use a variety of different attacks; punches, kicks, and fireballs. Players can also move around the arena by walking or jumping, and can also block incoming attacks. Usually, one game consists of a best out of three rounds format.

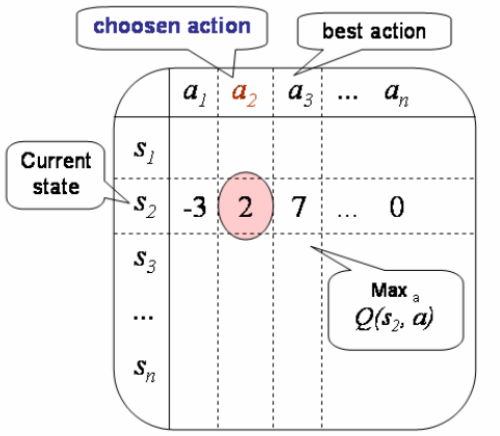

Figure 3: Challenge Sensitive Action Selection [2]

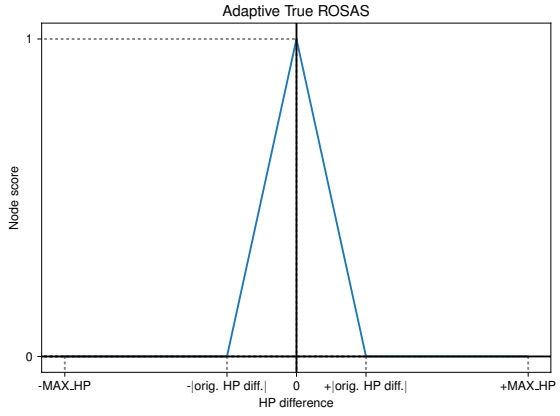

Figure 4: Score metric used by *Adaptive True ROSAS*

5 DYNAMIC DIFFICULTY ADJUSTMENT

Using Monte Carlo Tree Search, we built three Dynamic Difficulty Adjustment agents, which we call adaptive AI opponents, for the 2D fighting game FightingICE. These agents are all based on the idea that an 'ideal' challenging game in one in which each player has a 50% chance to win, on average the players should draw. For a 2D fighting game, a draw occurs when both players simultaneously reduce each others health to 0 or the total amount of health of both players is the same when time expires. Based on this information each of the adaptive AI opponents in this work aim to ensure that the health differential between the player and the adaptive AI opponent is as close to 0 as possible. Each agent is given a fixed amount of search time, search depth and simulation frames, based the distance from the player. In close (< 150 pixels) agents get 2 frames of time to search, search depth is set at 2 and 40 simulation frames. At a distance (> 150 pixels) agents get 6 frames of time to search, search depth is set at 8 and 60 simulation frames.

5.1 Challenge Sensitive Action Selection

The first adaptive AI opponent that we are testing in this work is based off Challenge Sensitive Action Selection [3]. This method of Dynamic Difficulty Adjustment uses Reinforcement Learning [25] to build a Q-table, off-line, of state-action pairs. For each state, it ranks the actions based on the Q-value of the actions (see Figure 3. It begins the game by selecting the mean ranked action for the initial state and continues to play this rank of action for each state until the game is re-evaluated after a fixed number of actions. Based on the health differential between the player and the adaptive AI opponent, the rank of the action selected for the next cycle is changed, with the aim to keep the health differential as close to 0 as possible. If the player is at higher health then the adaptive AI opponent, the action rank is increased up by one rank, thus increasing the difficulty. If the player is at lower health than the adaptive AI opponent, the action rank is decreased by one rank, thus decreasing the difficulty. Actions with the same rank are chosen at random. This process then repeats each evaluation cycle.

Using this methodology for the action selection policy, we designed a new DDA agent that replaces Reinforcement Learning with MCTS-UCT to perform the tree building in real-time. The actions where ranked based on the assigned node score from the MCTS process. The actions are selected in a similar way with the adaptive AI opponent again choosing the mean action, to begin with, and changing the rank of the action selected after each evaluation cycle. In our case, the evaluation cycle was every 9 seconds.

5.2 Reactive Outcome Sensitive Action Selection

Reactive Outcome Sensitive Action Selection (ROSAS) [12] is the next agent in this work that we are testing. Similar to CSAS, this agent also uses MCTS-UCT to build a search tree, by which the playouts are performed and evaluated to the standard MCTS-UCT method. The scoring metric of the nodes is proportional to the total possible health point differential between the player and the agent (normalised to be between 0 and 1 for UCT), with a health point differential of 0 resulting in a node score of 0.5. Rather then selecting the node with the highest node score, ROSAS employs a different action selection policy, choosing the action that leads to a node score closets to 0.5:

$$\text{action} = \arg\min \begin{cases} 0.5 - r[a].\text{score} & \text{if } r[a].\text{score} \leq 0.5 \\ r[a].\text{score} - 0.5 & \text{otherwise} \end{cases}, \quad (4)$$

where r is the root node of the tree and $r[a]$ is the child of r corresponding to action a. This is the distance from the score of 0.5.

5.3 Adaptive True Reactive Outcome Sensitive Action Selection

The final Dynamic Difficulty Adjustment Agent that we are testing is Adaptive True Reactive Outcome Sensitive Action Selection (ATROSAS). This is an extension to True ROSAS [12]. The weakness with the first two algorithms is that MCTS builds the search tree asymmetrically, with actions that are stronger explored more. This means that there may be a very shallow exploration of the actions at the rank or node score of interest, meaning they may not be a true reflection of what will happen once that action is selected. ATROSAS leverages the strength of the asymmetric search tree

by changing the areas that are explored more, by changing the scoring metric in the node evaluation of MCTS. We do this with the variables of the health difference before the action and after the action is performed and incorporate this into the node score according to:

$$\text{node.score} = \begin{cases} \frac{y-x}{y} & \text{if } 0 \leq \text{HP difference} \leq |\text{orig. HP diff.}| \\ \frac{y+x}{y} & \text{if } |\text{orig. HP diff.}| \leq \text{HP difference} < 0 \\ 0 & \text{otherwise} \end{cases}, \quad (5)$$

where x is the HP difference after the action and y is the HP difference before the action. Figure 4 shows the node score metric function used by ATROSAS. ATROSAS can now use the standard MCTS action selection policy, as the best action is now the one that results in a 0 health point differential.

6 EXPERIMENTAL SETUP

To test the various Dynamic Difficulty Adjustment agents we conducted a human trial. In this trial, the human players played 1 game (three timed rounds) against each of the Dynamic Difficulty Adjustment agents, CSAS, ROSAS, and ATROSAS as well as a game against MCTS-UCT with standard node evaluation and action selection policies. Each round lasted 90 seconds and players began with 500 health points. We had 31 total participants. Each participant was given an initial questionnaire to find their age, gender, familiarity and skill level within the 2D fighting game genre (see Table 1. The participants did not know which agent they were playing against in each game during the trial. At the end of each game, the participants were asked the following questions:

- **On a scale of 1-5 how difficult was this opponent? (1 being least difficult, 5 being most difficult)**: This question was used to gauge the perceived level of difficulty of each of the agents. The ideal level of challenge would be for players to score the agents a 3 in this category, implying the agent was not to difficult and not too easy. We expect the pure MCTS-UCT to be the most difficult agent.
- **On a scale of 1-5 how enjoyable was this opponent? (1 being least enjoyable, 5 being most enjoyable)**: This question was aimed at developing an understanding of the enjoyment level of the agents. We would expect the Dynamic Difficulty Adjustment agents to be the most enjoyable, and ATROSAS to perform the best.
- **On a scale of 1-5 how realistic did you find the performance of this agent? (1 being least realistic, 5 being most realistic)**: With this question, we want to find out how realistic the agents were. An issue with many types of AI opponents adaptive or not, is they feel very artificial and playing against an unrealistic opponent can break game flow and immersion.

7 RESULTS AND DISCUSSION

After the trial, we collated the data for each of the games played and the corresponding answers to the questionnaires. We calculated the win percentages of each of the agents, shown in Table 2, to see if the agents were able to achieve the desired win percentage of 50%. As expected we observe that the MCTS-UCT had an extremely high win rate, this was not surprising as it was not trying to adapt to the player, rather, playing the game at the hardest possible difficulty

Sex	No.	Age	No.	Familiarity	No.	Skill	No.
M	25	20-30	20	5 (Most Familiar)	0	5 (High Skill)	3
F	6	30-40	8	4	0	4	11
		40-50	2	3	16	3	12
		50+	1	2	10	2	2
				1 (Least Familiar	5	1 (Low Skill)	3

Table 1: Pre-game questionnaire responses

Agent	Average Win %
CSAS	40.860
ROSAS	46.335
ATROSAS	51.075
MCTS-UCT	90.323

Table 2: Win percentage of the agents vs. the human players

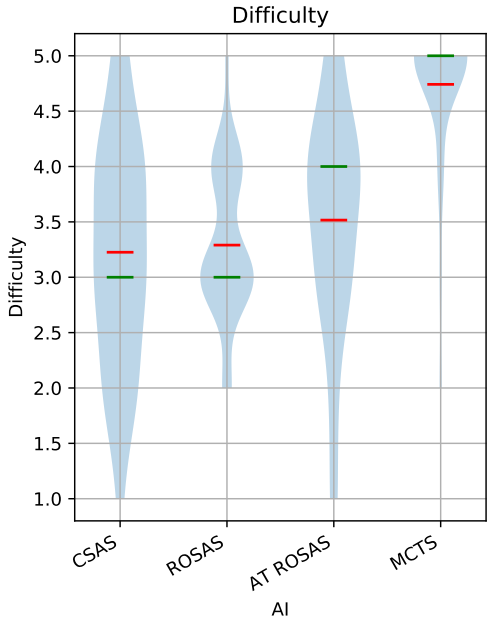

Figure 5: Violin plots of questionnaire responses to perceived difficulty (1 being least difficult, 5 being most difficult)

setting. ATROSAS had the closest win percentage nearest to the target of 50%, beating out both CSAS and ROSAS. As mentioned previously, this because ATROSAS takes advantage of how MCTS builds the search tree, and the actions that are most relevant get the most search time. This makes the outcomes of the actions it selects more accurate at achieving the target when compared to over the other two agents.

Figure 5, 6, and 7 are violin plots of the questionnaire responses after each game. The blue area in the plot is the actual distribution of the answers, where the mean answer is shown in red and the median is shown in green.

If we analyse these responses we can see that from Figure 5 the MCTS-UCT agent was the most difficult with the mean response at ≈ 4.7 (5 being the most difficult) and the distribution of responses

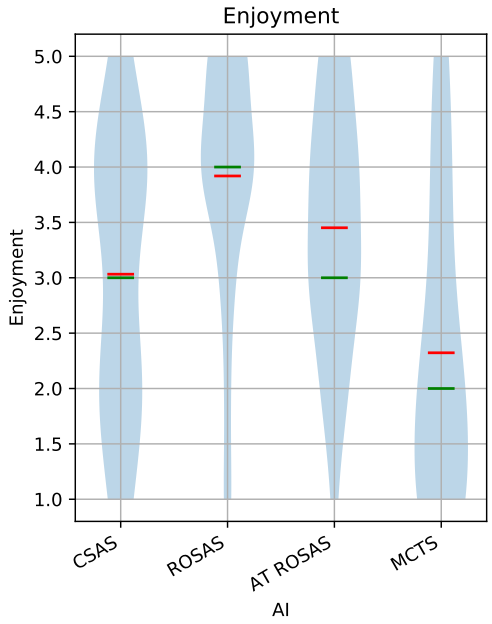

Figure 6: Violin plots of questionnaire responses to game enjoyment (1 being least enjoyable, 5 being most enjoyable)

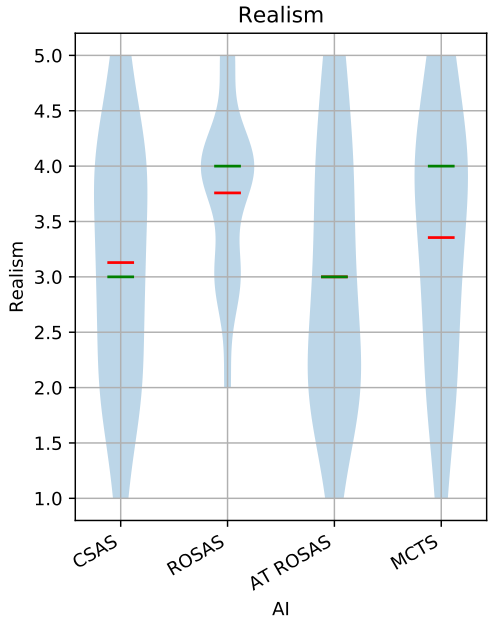

Figure 7: Violin plots of questionnaire responses to realism of opponent 1 being least realistic, 5 being most realistic)

closely packed around this number. MCTS-UCT was also the least enjoyable agent to play against (Figure 6) with a mean response of ≈ 2.4 (with 1 being the least enjoyable). This gives strength to the argument that if a game is too hard players will find it less enjoyable. In terms of realism, the distribution of answers is more uniform suggesting that it was hard to gauge how realistic the agent was, this may have been an effect of the agent being very aggressive and fast-paced, which can be hard to interpret for newer players of the genre.

The questionnaire responses to the agents were surprising. Players found that the ROSAS (mean of ≈ 3.3) and CSAS (mean of ≈ 3.2) agents on average to be at the right level of difficulty, see Figure 5. It is arguable that ROSAS is the better agent in this respect as its distribution is more centered around the mean, with CSAS distribution being more uniform across the range of responses. Whilst ATROSAS is a stronger algorithm (targets a win percentage closer to 50%) it is curious that people found it more difficult than CSAS and ROSAS. This could be caused by the fact that it better matched the health point difference and thus the players were never able to gain any real advantage against it, thus making it feel more difficult. This brings into question whether or not a win percentage of 50% which most DDA systems target really is the ideal level of challenge for non-competitive games.

When we investigate enjoyment (Figure 6) we find that ROSAS was the most enjoyable agent to play against (mean of ≈ 3.9, with 5 being the most enjoyable). The distribution of responses is again closely packed around this number, indicating that this is a more accurate result, then the means of the CSAS and ATROSAS. Whilst this agent didn't track the target win percentage as well as ATROSAS it was a more enjoyable agent to play against. We speculate that this may be caused by the fact that the agent doesn't track as well the health point difference, which may give a sense of a lead to the player, making it more enjoyable. In terms of realism, ROSAS is the most realistic agent (mean of ≈ 3.7, with 5 the most realistic, see Figure 7). Unlike the other agents, its distribution is centered more around its mean, giving more confidence to this result.

From these results, we can see that a win percentage of 50% is not the ideal level of challenge when it comes to enjoyment. It may be that whilst a 50% win ratio produces a fair game in a competitive environment when playing non-competitively this may not be the best if enjoyment is more of a concern. However, if the win percentage is too low which is the case for CSAS this may also result in a less enjoyable agent as it is too easy to beat. Finding the ideal win/loss ratio may also vary from genre to genre.

8 CONCLUSION AND FUTURE WORK

It is interesting that the most enjoyable and realistic Dynamic Difficulty Adjustment agent was not the best performing agent in terms of win percentage target. The ROSAS agent was perceived to be at the right level of challenge for the players, we speculate that this was caused by the fact that it had a slightly lower win percentage than 50%. This work attempts to investigate the question: what is the best level of challenge for a video game in a non-competitive environment? This is not a simple question to answer, although we find that win/loss ratio 50% is not ideal, understanding the motivations and the reasons behind why the player is playing the game in the first place need to be explored, before a "definitive" win/loss ratio can be found.

Understanding the motivations of the players, and the reasoning behind why they are playing the game is a step that is commonly overlooked by much of the research in the Dynamic Difficulty Adjustment field. More research is needed in this area to better

understand what is the ideal level of challenge when it comes to making an enjoyable video game. In future work we plan to change how Adaptive True ROSAS (as it is the strongest agent at achieving the target outcomes) works such that it can target different win/loss ratios. This can be achieved by altering the shape of the curve (see Figure 4), through a change in health point difference targets. This will enable further investigation into the correlation between win percentage and enjoyment. Coupled with more detailed questionnaires, to learn more about the motivations of players, we aim to find a more definitive level of challenge for developers of Dynamic Difficulty Adjustment agents in non-competitive games.

9 ACKNOWLEDGEMENTS

This work is funded by the Digital Creativity Labs (www. digitalcreativity.ac.uk), jointly funded by EPSRC/AHRC/InnovateUK [grant number EP/M023265/1].

REFERENCES

[1] Amy L Alexander, Tad Brunyé, Jason Sidman, and Shawn A Weil. 2005. From gaming to training: A review of studies on fidelity, immersion, presence, and buy-in and their effects on transfer in pc-based simulations and games. *DARWARS Training Impact Group* 5 (2005), 1–14.

[2] Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. 2005. Challenge-sensitive action selection: an application to game balancing. In *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*. IEEE, 194–200.

[3] Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. 2005. Extending reinforcement learning to provide dynamic game balancing. In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*. 7–12.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.

[5] Phillipa M Avery and Zbigniew Michalewicz. 2010. Adapting to human gamers using coevolution. In *Advances in Machine Learning II*. Springer, 75–100.

[6] Sander Bakkes, Pieter Spronck, and Jaap van den Herik. 2009. Rapid and reliable adaptation of video game AI. *Computational Intelligence and AI in Games, IEEE Transactions on* 1, 2 (2009), 93–104.

[7] Ed Boon and Tobias John. 1992. *Mortal Combat*. Game [Arcade]. Midway Games, Chicago, Illinois, U.S.

[8] Emily Brown and Paul Cairns. 2004. A grounded investigation of game immersion. In *CHI'04 extended abstracts on Human factors in computing systems*. ACM, 1297–1300.

[9] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[10] Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey. 2008. Toward an understanding of flow in video games. *Computers in Entertainment (CIE)* 6, 2 (2008), 20.

[11] Mihaly Csikszentmihalyi. 2008. *Flow : the psychology of optimal experience* (1st harper perennial modern classics ed. ed.). Harper Perennial, New York.

[12] Simon Demediuk, Marco Tamassia, William L. Raffe, Zambetta Fabio, Xiaodong Li, and Florian "Floyd" Mueller. 2017. Monte Carlo Tree Search Based Algorithms for Dynamic Difficulty Adjustment. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE.

[13] AE Elo. 1961. New USCF rating system. *Chess Life* 16 (1961), 160–161.

[14] Sylvain Gelly and David Silver. 2011. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence* 175 (2011), 1856–1875. http://www.sciencedirect.com/science/article/pii/S000437021100052X

[15] Ya'nan Hao, Suoju He, Junping Wang, Xiao Liu, Wan Huang, et al. 2010. Dynamic difficulty adjustment of game AI by MCTS for the game Pac-Man. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, Vol. 8. IEEE, 3918–3922.

[16] Robin Hunicke and Vernell Chapman. 2004. AI for dynamic difficulty adjustment in games. In *Challenges in Game Artificial Intelligence AAAI Workshop*. 91–96.

[17] Ritsumeikan University Intelligent Computer Entertainment Lab. 2017. Fighting Game AI Competition. *http://www.ice.ci.ritsumei.ac.jp/ ftgaic/* (2017).

[18] Charlene Jennett, Anna L Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. 2008. Measuring and defining the experience of immersion in games. *International journal of human-computer studies* 66, 9 (2008), 641–661.

[19] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[20] Xinyu Li, Suoju He, Yue Dong, Qing Liu, Xiao Liu, Yiwen Fu, Zhiyuan Shi, and Wan Huang. 2010. To create DDA by the Approach of ANN from UCT-Created Data. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, Vol. 8. IEEE, V8–475.

[21] Thomas W Malone. 1980. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*. ACM, 162–169.

[22] Christopher Z Mooney. 1997. *Monte carlo simulation*. Vol. 116. Sage Publications.

[23] Lingdao Sha, Souju He, Junping Wang, Jiajian Yang, Yuan Gao, Yidan Zhang, and Xinrui Yu. 2010. Creating appropriate challenge level game opponent by the use of dynamic difficulty adjustment. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, Vol. 8. IEEE, 3897–3901.

[24] SuperData. 2017. Market Brief - 2017 Digital Games & Interactive Media Year in Review. https://www.superdataresearch.com/

[25] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[26] Penelope Sweetser and Peta Wyeth. 2005. GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)* 3, 3 (2005), 3–3.

[27] Takashi Nishiyama. 1987. *Street Fighter*. Game [Arcade]. CAPMCOM, Osaka, Japan.

[28] Georgios N Yannakakis and John Hallam. 2006. Towards capturing and enhancing entertainment in computer games. In *Hellenic Conference on Artificial Intelligence*. Springer, 432–442.

[29] Georgios N Yannakakis and Manolis Maragoudakis. 2005. Player modeling impact on player's entertainment in computer games. In *User Modeling 2005*. Springer, 74–78.

[30] Alexander Zook, Brent Harrison, and Mark O Riedl. 2015. Monte-carlo tree search for simulation-based strategy analysis. In *Proceedings of the 10th Conference on the Foundations of Digital Games*.