Go with the Flow: Reinforcement Learning in Turn-based Battle Video Games

Elinga Pagalyte
116448744@umail.ucc.ie
School of Computer Science and Information Technology, University College Cork, Ireland

Maurizio Mancini
m.mancini@cs.ucc.ie
Insight Centre for Data Analytics, School of Computer Science & IT, University College Cork, Ireland

Laura Climent
l.climent@cs.ucc.ie
Insight Centre for Data Analytics, School of Computer Science & IT, University College Cork, Ireland

ABSTRACT

Game flow represents a state where the player is neither frustrated nor bored. In turn-based battle video games it can be achieved by Dynamic Difficulty Adjustment (DDA), whose research has begun rising over the last decade. This paper introduces an idea for incorporating DDA through the use of Reinforcement Learning (RL) to agents of turn-based battle video games. We design and implement an RL agent that shows, in a simple environment, the idea of how a game could achieve balance through adequate choices in actions depending on the player's level of skill.

For achieving this purpose, we incorporated the design and implementation of state-action-reward-state-action (SARSA) algorithm to the agent of our implemented game. In addition, we added tracking of the on-going games and depending on the frequency of the player's repeated wins or losses, the rewards of the RL agent are modified. This modification of the rewards has an impact on the RL agent's actions, which involves an increase/decrease of the difficulty of the battle game. The evaluation performed shows that the idea of the paper is demonstrated, since players face personalized challenges that we believe are in range of game flow.

KEYWORDS

Game Flow, Dynamic Difficulty Adjustment, DDA, Reinforcement Learning, RL, SARSA, Turn-based battle video game

ACM Reference Format:
Elinga Pagalyte, Maurizio Mancini, and Laura Climent. 2020. *Go with the Flow:* Reinforcement Learning in Turn-based Battle Video Games. In *IVA '20: Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents (IVA '20), October 19–23, 2020, Virtual Event, Scotland Uk.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3383652.3423868

1 INTRODUCTION

Although research in Dynamic Difficulty Adjustment (DDA) has been rising [**?**], its application in today's video games is still lacking in some smaller areas. The importance of introducing DDA in video games is evident, for example, in the research of Xue et al. [**?**]. They show that by improving the way in which a game responds to the player, the player's experience improves and a higher number of players will continue to play for a longer period of time. Most of the current research on DDA is done through different (RL and non-RL) approaches, different game genres, however, there are still some aspects that could be explored more. Many games and papers focus on the modulation of game difficulty, yet, they do so by giving or taking something away from either the player or their opponents. In terms of turn-based battle games, not all apply RL to create smarter opponents.

Battle video games, like *Pokémon*, implement a turn-based battle system: players are fighting enemies throughout the game, trying to reach a final goal. The difficulty of these games is constantly increased by introducing stronger enemies, i.e., ones with higher statistics (e.g., attack and defense points), rather than enemies that choose moves by taking into account the player's skill level. Other than becoming stronger, these opponents do not change in a way that would entice players who, perhaps, find it too difficult or easy. These types of games would benefit from modifying the enemy to maintain game flow, using RL to direct the opponent to choose moves that are less likely, or more likely, to cause it to win. Instead, RL is usually used to create a smarter player agent to play through theses types of games [**?**]. The development of a smarter opponent combined with DDA could lead to a more diverse, compelling gameplay that would attract players of all skill levels. Thus, the motivation here reaches towards introducing the idea of a new approach using RL to achieve DDA and game flow for turn-based battle games.

The aim is to modify the opponent during a battle by implementing a Reinforcement Learning (RL) agent that can choose its own actions (e.g., to hit, to heal, etc.), rather than changing its statistics (e.g., attack and defense scores). We test different rewards and exploration-to-exploitation ratios of the agent to see what works best for this setting. The rewards (positive and negative) are also swapped throughout the battle. This helps introducing DDA into the RL setting, as it encourages, or discourages, the agent from winning depending on the progress of the player.

In this paper, we also focus on game *flow*, that Csikszentmihalyi defines as the"the holistic sensation present when we act with total involvement" [**?**]. Flow can be modeled as a gameplay state in which there is a good balance of challenge and skill level for the player: these two dimensions must vary together to ensure flow, i.e., the more the player feels "challenged", the more they will need to be and feel "skillful", and vice-versa. We hypothesize that, DDA through RL will contribute to maintain a good balance between these two dimensions, by implementing a turn-based battle video game. Then, we evaluate the game to test 2 research questions: (1)

IVA '20, October 19–23, 2020, Virtual Event, Scotland Uk

ACM ISBN 978-1-4503-7586-3/20/09…$15.00
https://doi.org/10.1145/3383652.3423868

does a RL approach have an effect on the type of actions chosen by the opponent during battle? (2) does a RL approach improve the game experience in terms of player's flow?

2 BACKGROUND: FLOW & DDA

To achieve a flow state, a "balance between perceived challenges and perceived skills" is required, creating a "merging of action and awareness, a sense of control, and an altered sense of time" [**?**]. Figure 1 depicts the concept of "flow channel" applied to video game playing: making a game too challenging for the current player's skill level (e.g., by increasing the opponents' strength or adding puzzles that are too difficult to be solved) will make player feel frustrated (moving from the flow channel to the red area); conversely, proposing challenges that can be easily faced by the player will make them feel bored (moving from the flow channel to the blue area).

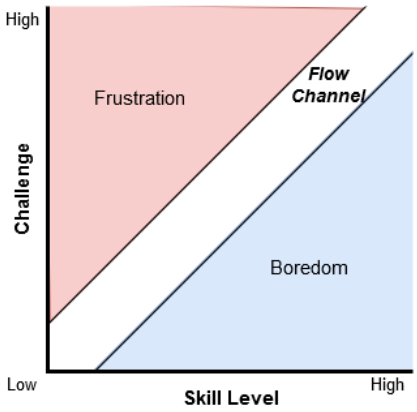

Figure 1: Csikszentmihalyi's flow model [?]. The player's gameplay experience is optimal if it is maintained in the flow channel, i.e., it never becomes too much or not enough challenging, respect to the player's skill level.

In order to keep the player in the flow channel as much as possible, a balance between the player's skill and the game difficulty has to be maintained, and this is the goal of Dynamic Difficulty Adjustment (DDA) [**?**]. DDA is a technique consisting in altering a game's properties during gameplay in order to make a game easier or more difficult. The adjustment is based on the player's progress throughout the game: if the player fails more than often, this will lead to feelings of frustration and the player ending the game; if the player wins too easily, boredom will quickly arise. DDA can result in the modification of the game parameters, features, behaviors and scenarios. It can target things such as the statistics of an enemy, i.e., how strong they are, how much health they have or what abilities they can use. It can alter the amount of spawned enemies or increase the amount of powerups the player will collect throughout the game. For games based on levels without enemies, it can result in changing the complexity of the levels. When applying DDA, the first objective is to Figure out which one(s) of these characteristics to adjust in order to help or hinder the player.

Another important factor of DDA is the tracking of the player's skill or progress. For example, Xue et al. [**?**] divide the player's progress in two parts: the current game level and the amount of tries it takes the player to successfully win the level. By computing a formula weighted by these factors, they can measure how well the player is responding to the game. Then, the difficulty of the following levels is adjusted accordingly.

3 STATE OF THE ART

Many papers tackle DDA in video games: some explore the use of RL (see Section 3.1), others adopt other machine learning approaches in which models are previously trained over a high number of plays to implement a more balanced play experience (Section 3.2). While the former approach does not need a long training phase, it does not necessarily tackle the creation of smarter opponents, as we aim to do: usually, RL is exploited to change the game's statistics or environment (e.g., the structure of the levels), or to create a smart player agent. The latter, instead, produces a better adaptation of the game difficulty but the technique cannot be easily transferred between players, as it will constantly require re-training.

Compared to existing works, the system we present in the paper learns from one player during gameplay rather than needing to be trained on a larger dataset. Also, according to the outcome of the learning process, it varies the actions of the player's opponent, instead of modifying the structure of the game. As a result, since the player's opponent learns during each battle rather than being trained beforehand, the game provides a challenge that is aimed towards the skills of the person playing the game, to achieve a flow experience.

We also explore the idea of increasing game difficulty, rather than just decreasing it, by surveying the output of the RL algorithm (Section 5) with many combinations of rates and rewards. Finally, whilst our system is tailored specifically to the case of turn-based battle games, the presented concepts can still be applied to games of different genres.

3.1 Reinforcement Learning and DDA

Many research works on games and RL aim at developing intelligent agents that would successfully play the game as the player themselves, such as in [**?**]. Softmax exploration with Q-learning is exploited to find an optimal battle strategy for the game Pokémon. The approach itself could be used to make better enemies in games, however, it would not comply with the DDA factor, as it does not adjust to the skills of the opponent agent; instead, it aims at creating the strongest possible player. The algorithm is set up to get the maximum result, i.e., to win every battle, and so it does not try to match its opponent's skill level. Other papers also tackle optimal strategies in turn-based strategy games [**? ? ?**]: they all focus on creating a strong opponent, or player, but they do not adapt their skill level to the opponent's one.

Works that do target DDA do so by adjusting the "physical" elements of the game, rather than the skills of the opponent [**?**]. Wender & Watson [**?**] and Amato & Shani [**?**] target the game *Civilization IV* to study RL approaches [**?**] in strategy games. The first work exploits Q-learning to add a RL-based agent to the game AI and let it perform actions according to the player's score (which is considered as the reward of the algorithm). The second one exploits Q-learning, model-based learning and learning with a factored

model to let the opponent choose one out of 3 action strategies, based on different world leader personalities. The system learns some state features, e.g., the the military strength of the player and the opponent, the amount of unoccupied land remaining, the population size, etc.

Other works, such as [**?**] use RL to determine the combat behavior of player's opponents to let them choose between fight and retreat, depending on some variables, such as the weapons state, distance from the enemy, number of enemies in the area, health points. The opponents become smarter in maximizing their ability to beat the player, but this is different from our approach. We aim at achieving DDA by letting the opponents modify their fight strategy in terms of the single attack or defense actions chosen during the fight. So, we aim at improving the player's experience by lowering and raising the opponents' fight skills, making the player feel that the fight has always a good balance of difficulty and fun.

Just a few works recently tried to achieve DDA using RL, e.g., Noblega et al. [**?**], aiming to achieve DDA (referred to as "game-balancing") in real-time games. Similarly to this work, we apply RL to a different game type, that is, turn-based battle games.

As highlighted in the works reviewed above, there is a general lack of research on the application of fight strategy planning systems to achieve DDA in turn-based games, to create well-balanced opponents.

3.2 Non-RL Approaches to DDA

Fernandes and Levieux [**?**] talk about directly adapting the difficulty towards the player's failure probability. Their system is based on a model of the challenge difficulty and, through a repetitive update of a logistic regression function, difficulty adjustment, and gameplay data recording, the model is kept up-to-date to monitor the game difficulty. The most important aspect is that the system does not require large amounts of gameplay data, meaning that newly released games can use it without having to scavenge for large amounts of data in order to train the model. Much like our research, it aims for an approach that would not require the model to be trained beforehand with large amounts of data whilst still achieving DDA.

The Hamlet system is a collection of functions integrated into the *Half Life* game engine [**?**]. The system constantly monitors the gameplay, and if it notices that the player is struggling to face enemy or challenges, it provides them with items that they can use in order to make the game easier. The main system heuristic function for determining game difficulty is the damage the player takes over time. By observing this information, the Hamlet system chooses to introduce more or less items to help the player. Although this approach does not apply RL nor does it create more adaptive enemies, it achieves some sort of flow by keeping the player contained in set states that match their skill.

Lora et al. [**?**] apply DDA through gameplay data clustering, in order to detect different playing styles in *Tetris*. The total score of a game and the play style indicates the level of the player, defining three levels: newbie, average, expert. Rather than challenging the player and constantly increasing game difficulty, the system aims at providing more help to players of lower level. Authors demonstrate that the player's average scores and the satisfaction increases when DDA is applied. Although the system was created mainly to be applied to Tetris, with some small adjustments it could be applied to similar puzzle games (e.g., *Candy Crush*). Whilst it does not have an enemy or RL, this approach is very interesting in that it tracks the player's progress closely and, much like the Hamlet system, it tries to keep the player in some sort of flow state in which their current skill level matches the presented difficulty, thus keeping them engaged in the gameplay.

4 TURN-BASED BATTLE VIDEO GAME

As previously mentioned, in this paper we illustrate the idea of making turn-based battle systems more interesting with the introduction of RL to achieve game flow. More specifically, we implemented a simple turn-based video game with a human player and an agent. The only objective for the player is to win as many battles as possible (within a game) against the agent.

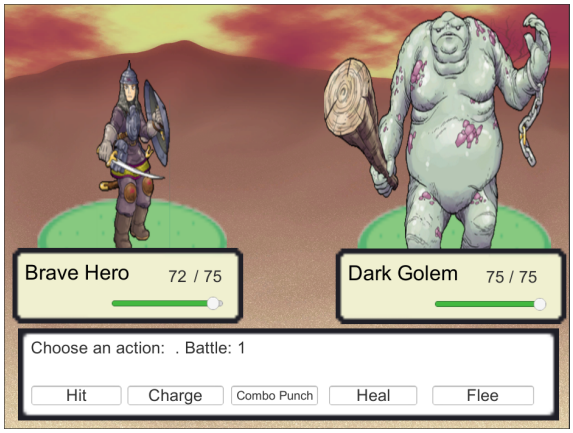

Figure 2: screenshot of the game's battle scene. The player's character is depicted on the left, while the opponent's character is on the right. Move selection for the player is shown in the bottom box.

4.1 Game Mechanics and Gameplay

The implemented game is a simple turn-based battle game, meaning that the player and opponent take turns to make a move. To keep it simple, the game is centered only around battling enemies. The player can of course quit at any time throughout the game.

Figure 2 shows the battle scene of the video game, which provides both the player and the opponent with four different action options. Both characters can attack their targets or heal themselves. At the start of each battle, both characters are restored to their initial state. This means that both are back to full health points (HP). The battle system performs the calculations and updates of the *units* HP after each play turn.

A *unit* is a frame for a character that describes its basic values: *health points (HP)*, *max health points*, *attack*, and *defense*. HP represents the unit's current state of HP, while max HP is the highest amount of HP a unit can have (75 units). The attack value indicates how well the character can deal damage. Defense shows how well a character can protect themselves against an attack. These

base values are important for determining the damage done, by use of a formula. If one attacks, it's attack points and the move's attack points, as well as the target's defense points will be put into a damage formula which then calculate the amount of damage that should be taken off the target's HP.

Both the player and opponent characters are given the same set of *moves* to be used in battle: a weak *hit* attack, a slightly more powerful *charge* attack, a *combo* attack, and a *heal* action. Additionally, the player is also given a *flee* action, which can be used to quit the game. By healing, the unit's HP increases by 10 points. The hit attack is thrown in as a control, it is weak (between 2 and 5 damage points) and very inefficient but the goal is to see how much either party uses it. The charge and combo attacks are the most beneficial as they can deal a lot of damage (charge up to 15, and combo up to 25 points at the very most). Combo is a combination of two to five hits (each between 2 and 5 points), meaning that the disadvantage of using it is that you may, by chance, get the smallest number of hits and thus deal a lot less damage than by using charge. Of course, on the other hand, if you get five hits of 5 points it will deal more damage than charge, however, that would be a rare achievement. Charge is a lot more reliable as it has a smaller range from which the damage can be calculated (between 10 and 15). So, whilst one is very consistent but not incredibly powerful, the other one is a lot more unpredictable with the amount of damage dealt.

5 REINFORCEMENT LEARNING AGENT

Q-learning is a very well-known and largely used RL algorithm which is highly popular in the area of DDA, as discussed in Section 3.1. In this section, however, we motivate our choice of using the SARSA (*State-Action-Reward-State-Action*) [**?**] algorithm over Q-learning. Even if they differ very slightly (i.e., just one difference in updating Q-values), SARSA is typically preferable when the agent's performance during the learning process is more important than the final performance. An extensive comparison between SARSA agents and Q-learning agents is presented in [**?**].

SARSA algorithm is an on-policy learner that updates the state after each action based on the reward obtained from the environment after performing such action (see Figure 3). The difficulty of the agent should increase slowly and according to the player's performance, rather than taking the optimal movements and becoming invincible. In addition, the agent should learn and adapt from early steps in the game and. This fits into our objective for the RL agent of avoiding "bad actions" since the early stages of the interaction.

5.1 States

We define the states of the RL agent based on its Health Points (HP) and the HP of the player. After an empirical evaluation of the number of states, the best performance was obtained with 17 states, which are shown in Table 1. There are five states where both parties have the exact same HP. In addition, there are 10 states where one party's HP is greater than the other's. In five of them the agent has more HP than the player, while in the remaining ones the opposite is true. Furthermore, there are two states in which one of the parties loses the game (one for the agent and one for the player), i.e., their HP are zero or below.

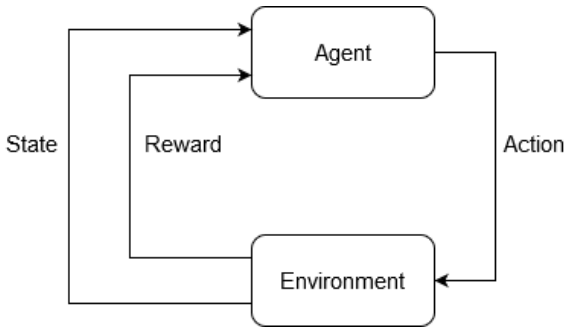

Figure 3: diagram of the SARSA algorithm.

State	Player HP	Agent HP
0	75	==
1	[51-74]	==
2	[31-50]	==
3	[16-30]	==
4	[1-15]	==
5	<Agent HP	[66-74]
6	<Agent HP	[46-65]
7	<Agent HP	[31-45]
8	<Agent HP	[16-30]
9	<Agent HP	[1-15]
10	[66-74]	<Player HP
11	[46-65]	<Player HP
12	[31-45]	<Player HP
13	[16-30]	<Player HP
14	[1-15]	<Player HP
15	0	-
16	-	0

Table 1: the states for our SARSA RL agent.

5.2 Rewards

We have defined two rewards for the RL agent: a positive reward for winning a battle (i.e., state 15) and a negative reward for losing a battle (i.e., state 16). Specifically we apply a reward 100/50 for winning and a penalty of -100/-50 for losing. The rewards can be configured before the start of any game. Section 6 evaluates the differences of the agent's behaviour based on the different rewards.

The agent keeps track of the number of times the player wins or loses in a row. If the player loses too many times in a row, the rewards are swapped, in order to lead the agent to lose instead of winning. The same occurs if the player wins too many times in a row: the rewards are switched back, so that the agent aims to win. Empirically, we determined that five wins/losses in a row is an adequate number to switching the rewards.

5.3 Exploration vs Exploitation

Exploration and exploitation are key concepts in RL algorithms. In the exploration, the agent gathers more information (by selecting random actions rather than taking the best action). In the exploitation, the agent takes the best action based on its current information. We have configured different rates for the exploration and exploitation, which can be set at the start of the game. We have fixed three configurations: (50,50), (30,70) and (70,30), where the first number represents the exploration and the second the exploitation. Section

6 evaluates the differences of the agent's behaviour based on these different configurations.

We have used the Epsilon-Greedy strategy. With this strategy, if the exploration vs exploitation rate is (30,70), it implies that the exploration policy is 0.3 and therefore the Epsilon-Greedy policy takes random actions 30% of the times. Furthermore, we have used Tabu Search Exploration in order to set certain actions as tabu for a certain number of times, and to allow the RL agent to explore the other actions more frequently.

The different combinations of rewards and exploration vs exploitation configurations are represented in the diagram in Figure 4.

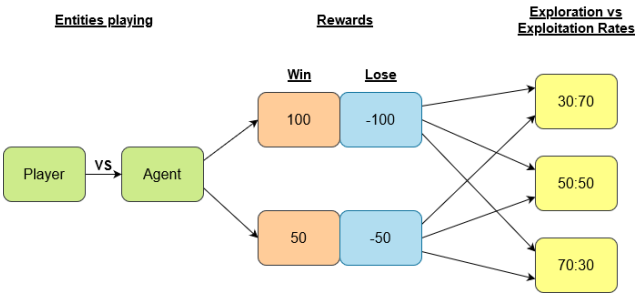

Figure 4: tree diagram with all possible parameters choices.

6 EVALUATION

We have conducted 2 evaluation studies in which participants play the turn-based battle video game against the RL agent presented in the previous section. As mentioned in Section 1, we aim to answer the following research questions: (1) does a RL approach have an effect on the type of actions chosen by the opponent during battle? (see study 1 in Section 6.1 and Section 6.2); (2) does a RL approach improve the game experience in terms of player's flow? (see study 2 in Section 6.3 and Section 6.4).

6.1 Study 1: RL agent performance and actions

In this study, 10 games were evaluated, each one consisting of 30 battles, giving a total of 300 battles played by four human players (two males and two females with age ranging from 20 to 40). Players' skills were comprised of two players being experienced, one being an intermediate level player and one player being a beginner. We refer to these 10 games with letters from *A* to *J*.

Games *A*, *H*, and *I* have rewards/penalties of (100,-100) and exploration vs. exploitation rates of (50, 50). Games *B*/*J* and *C*/*G* have as exploration vs. exploitation rates of (30,70) and (70,30), respectively (while keeping the same rewards). Instead, games *D*, *E* and *F* have rewards/penalties (50,-50), and their exploration vs. exploitation rates are, respectively: (50,50), (30,70) and (70,30).

We will now focus on the two games of study 1 denoted as games *A* and *B*. Both of these games were set up using similar rewards. Game *A* has a reward of 100, a penalty of -100 and exploration vs. exploitation rates of (50, 50): so, the agent was rewarded more for winning but also penalized more for losing; it also means that it had an equal opportunity to explore and learn throughout the game. Game *B* has the same reward and penalty as game *A*, however, its exploration and exploitation rates are (30, 70), meaning that the agent focused more on learning rather than exploring.

6.2 Study 1: Results & Discussion

Figure 5 shows the results of study 1. We can observe that, in most of the games, players usually quickly discover the best way to utilize the available moves. Specifically, it can be observed that the moves the players use the most are *Charge* and *Combo*. The explanation behind the popularity of these moves is that they are the attacks that produce the highest damage. On the other hand, actions *Heal* and *Hit* are used fewer times. As previously stated in Section 4, *Hit* is mostly used as a control move.

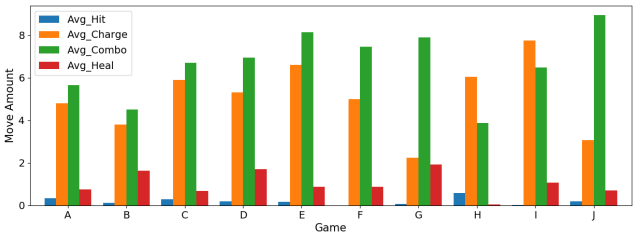

Figure 5: player's average moves over 10 games (A-J).

A more interesting outcome of study 1 is represented in Figure 6, in which it can be observed how the RL agent learned to select its actions. As we expected, while the players (all of them: beginner, intermediate and expert skilled) adapt quickly, the RL agent requires more time to learn. The action that is most often used by the RL agent is *Heal*. This action has no usage restriction, which means that can be used as many times as desired in a game.

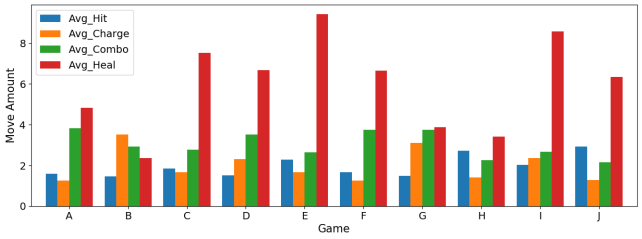

Figure 6: RL agent's average moves over 10 games (A-J).

We now analyze in more detail a specific game. Figure 7 shows the behavior followed by the RL agent in every battle of the game *A*. Recall that exploration vs. exploitation rates of game *A* are (50, 50). In this figure, it can be observed the overuse of *Heal* movement. Its maximum usage was in battle 17, where it was used 25 times. This produced a dragging out of the battle that was also reflected in the agent's HP in Figure 8, where you can see the changes in HPs throughout battles 16 – 20 (of game *A*). The beginning of every battle can be distinguished by the 75 HP for both lines (blue and red). In this figure, it can be observed that the use of *Heal* by the RL agent during the first half of the graph stretches out the battle. Whilst in some cases this move can be used to gain an advantage,

here it was overused. In this scenario, the agent usually lost the battle, because it was too focused on healing instead of attacking.

As the game continues, it can be observed in the figure that in the second half of the graph the RL agent does not use *Heal* movement as often (opting, instead, to attack more often).

In conclusion, game *A* represents well a typical performance of our RL agent and, consequently, the evolution of a battle in our video game. In addition, it shows how the frequent use of *Heal* is followed by a diminished use. Although this move displayed a negative outcome, the game play provides a challenge and with its application a lot of the battles were neck and neck for whether the player or the agent will win. The players acknowledged that in some cases "the agent plays like an actual player". In addition, one of the players commented, "it is challenging, the enemy plays like an actual player; sometimes the enemy spams the heal move as if they know they are backed into a corner and all they can do is heal until they gain an advantage". Of course, whilst the player may have won more often in some cases, the idea of flow can, in our opinion, still be seen in this study. Many games had battles where either could have won, we now see that through this (and also from the players comments) the idea of flow is not strictly confined to just the winning and losing of games. Consecutive wins by the player does not mean that the challenge provided is not suitable for that player. It would be noteworthy to keep track of and analyze the player's and enemy's usage of moves and HP throughout each battle in more detail in order to realise the effectiveness of difficulty balancing and when it should be, more suitable, applied.

The last part of study 1 analyzes the frequency of wins and losses of the players. Specifically, Figure 9 focuses in game *A* and game *B*. Game *B* is interesting in particular as it is the only game in which *Heal* is used less often (see Figure 7). In addition, in game *B* there was a high similarity between the player's moves and the RL agent's moves (see also Figure 5). Compared to other games, game *B*'s RL agent focused mostly on attacking and only healed when it deemed necessary. This behaviour can be explained by the exploration vs. exploitation rates: (30, 70), respectively. In our opinion, a higher rate of exploitation allowed the agent to learn faster from mistakes, such as healing consecutively. Therefore, it produced a result in which the game outcomes came out to a scenario of equal wins and losses, with maximum frequency of four wins/losses in a row. In contrast, game *A*'s player (with a RL agent with (50,50) rates) won 20 of the 30 games, with 6 wins in a row.

Figure 9 also provides more insights about the RL performance. Note that the player tends to win more often towards the start. The player loses few times at the beginning of the game, however, after a third of the game, they begin to lose more often, as the RL agent is learning the best actions at early stages of the game. As the games progress, the agent learns to challenge the player more successfully, such as in game *B*, which has an equal amount of wins and losses.

Overall, the games ranged from the player winning 80% to 50% of the battles. Most noteworthy were games *A* and *B*. In game *B* there was a result of 50-50 of win-lose ratio. The RL agent applied better actions, specially towards the end, playing in equal skills to the player. Game *A* showed how the agent learns from its losses and bad patterns, such as overusing *Heal*, and instead attacks with more powerful moves. Although it would get stuck in these loops of using *Heal*, it would only happen for a handful of battles in each game, and ultimately it would learn to use it less often. So, we can conclude that the best configuration of exploration vs. exploitation rates is: (30, 70) (configuration used in game *b*).

6.3 Study 2: Gameplay Experience

In the second evaluation study, we aim to test whether the player's gameplay experience, and hence game flow, is influenced by the gameplay type. To measure gameplay experience, we select 4 components of the Game Experience Questionnaire (GEQ) [**?**]: *successful* and *skillful*, measuring the level of competence the player experienced during gameplay; *challenge* and *effort*, corresponding to the level of challenge experienced by the player. Referring back to the model of flow described in Section 2, these 4 components can be matched to the 2 plane dimensions: *successful* and *skillful* to axis X *Skill Level*; *challenge* and *effort* to axis Y *Challenge*.

We define 5 gameplay types: easy (without RL), hard (without RL), 50-50 (using a RL-based agent with exploration vs. exploitation rates of (50,50)), 30-70 (using a RL-based agent with rates (30,70)) and 70-30 (using a RL-based agent with rates (70,30)).

Ten players of varying age and skill levels are asked to play 3 battles for each gameplay type. After each group of 3 battles, they have to fill out a questionnaire in which they have to rate the 4 gameplay experience components described above on a Likert scale going from 0 ("not at all") to 4 ("extremely").

We hypothesize that gameplays using the RL-based agent will result in a "better" gameplay experience, in terms of skill level and challenge. So, our hypothesis will be confirmed if the gameplay experience ratings of the RL-based agents are "closer" to a value of 2 (i.e., the middle value of the $0 - 4$ Likert scale) than the other types of agent. Also, we expect that the agent with the best configuration determined in study 1, the one with a (30,70) exporation vs exploitation rate, should provide a better flow experience.

6.4 Study 2: Results & Discussion

Table 2 reports the mean and standard deviation of the ratings of the 4 gameplay experience components for each gameplay type, across participants. We also computed paired t-tests with Bonferroni correction on all the combinations of gameplay types to find those exhibiting significantly different means. Results are reported in Table 3.

The ratings of most of the components (except for the *skillful* one, whose meaning was probably misunderstood by the participants) exhibit significant differences when the player's opponent is not driven by the RL agent vs when it is driven by the RL agent (Table 3). Also, the 3 RL-driven opponents did not show any statistically significant difference in terms of player's ratings, as reported in the last 3 rows of the table. That could mean that the face of using RL vs not using it at all makes a difference, but slight changes in the way the RL works do not have an impact on flow.

The mean values in Table 2 also show that the most and least challenging gameplays are those not using the RL-based agent, which is a reasonable result, providing an initial confirmation of our the hypothesis that a better flow can be achieved when the player's opponents is driven by a RL-based agent.

By hypothesizing that an experience of flow will result in middle scores (e.g., between 1.5 and 2.5) of the component ratings, we can

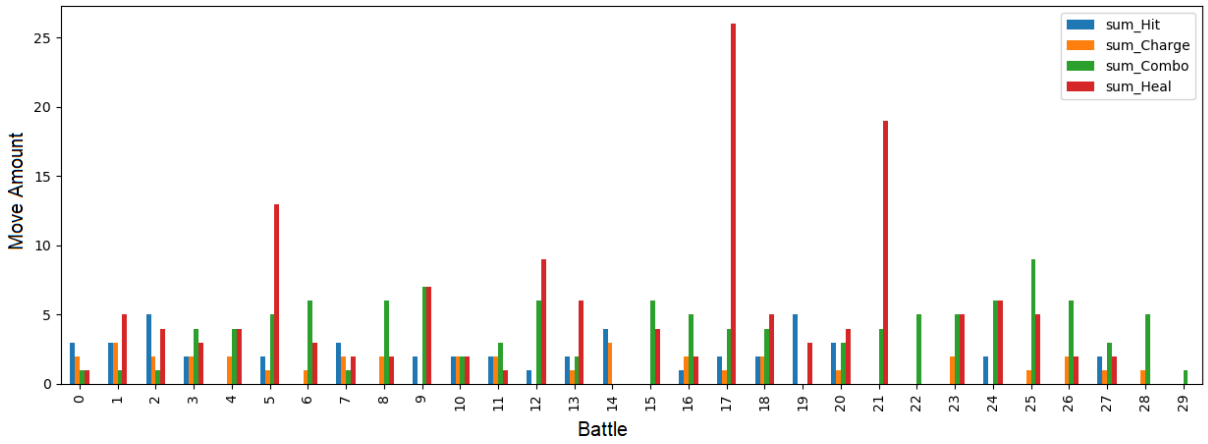

Figure 7: RL agent's moves sum in the 30 battles of game *A*.

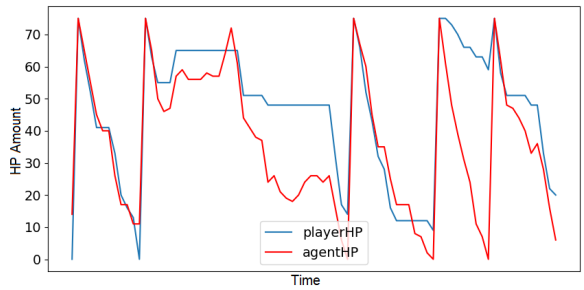

Figure 8: Player's and agent's HPs from battles $[16-20]$ of game *A*.

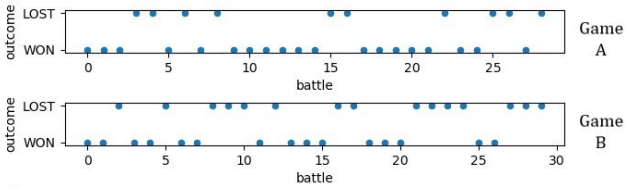

Figure 9: player's outcome for the battles in games *A* and *B*.

observe that, for example, the RL-based agent with an exploration vs exploitation rate of (50,50) has always been rated between 2.3 and 2.5 for the 4 gameplay experience components (even though statistical difference with the other gameplay types is not always achieved). Again, this result seems to reinforce the indication we received from the results discussed above, that a RL-based agent could help to improve flow in video games.

Regarding the second part our hypothesis, that is, the one about the 30-70 agent, we had some interesting results. First of all, the 30-70 agent was the one, among the ones driven by the RL algorithm, that obtained the highest scores in terms of success, skillful, challenge and effort (Table 2). So, despite it was the most challenging and the one requiring most effort to be defeated, it was also the one making feel the player more skillful and successful. It must be also noted that these scores were not significantly different from the ones of the other agents (Table 3), as we said above.

Comp.	Game type	μ	std		Comp.	Game type	μ	std
success	easy	3	1.33		challenge	easy	0	0
success	hard	**1.5**	0.85		challenge	hard	3.6	0.7
success	50-50	**2.5**	1.18		challenge	50-50	**2.2**	0.79
success	30-70	3.1	1.29		challenge	30-70	3	0.47
success	70-30	**2.2**	1.32		challenge	70-30	**2.5**	0.97
skillful	easy	2.8	1.23		effort	easy	0.1	0.32
skillful	hard	**2.3**	1.42		effort	hard	3.3	1.06
skillful	50-50	**2.5**	1.27		effort	50-50	**2.3**	1.06
skillful	30-70	2.8	1.4		effort	30-70	2.8	0.92
skillful	70-30	**2.5**	1.27		effort	70-30	**2.5**	1.08

Table 2: results of the gameplay experience evaluation study. For each component (successful, skillful, challenge, effort) and game type (easy, hard, 50-50, 30-70, 70-30) we report the corresponding mean and standard deviation. Medium values (between 1.5 and 2.5), i.e., those corresponding to a "balanced" gameplay experience, are highlighted in bold.

Gameplay type pairs	success	skillful	challenge	effort
easy vs hard	**0.03**	1	**0.001**	**0.001**
easy vs 50-50	1	1	**0.001**	**0.005**
easy vs 70-30	0.107	1	**0.001**	**0.001**
easy vs 30-70	1	1	**0.001**	**0.001**
hard vs 50-50	0.085	1	0.067	0.848
hard vs 70-30	1	1	**0.011**	0.107
hard vs 30-70	**0.002**	0.522	0.239	0.957
50-50 vs 70-30	1	1	1	1
50-50 vs 30-70	0.239	0.811	0.368	1
70-30 vs 30-70	0.1	0.811	1	1

Table 3: results of paired t-tests computed between all the combinations of gameplay types. Significant p-values ($p < .05$) after Bonferroni correction are highlighted in bold.

7 CONCLUSION AND FUTURE WORK

The main objective of this paper is to show a new approach to achieving game flow through the introduction of DDA and RL in turn-based battle games. With such purpose, we implement a turn-based battle video game by incorporating DDA through the definition of RL-driven agents that determine the opponent's moves. By modifying the agent's choices of actions rather than the environment or the agent's statistics, this paper aims to show how a smarter enemy might better adapt to the difficulty of the game and to every player. Therefore, providing challenge to more experienced players whilst encouraging beginners.

The designed and implemented RL agent learns through each turn and battle, learning by trial and error what kind of actions are the most adequate in the battle. We presented a system of states, rewards/penalties and exploration vs. exploitation rates for a SARSA algorithm that constitutes the RL agent.

After evaluating the RL agent, we observed that, in our opinion, the players are fairly challenged throughout most of the games. Although in some of the games the battles were won mostly by the players, it was realised that game flow does not solely depend on just the constant winning or losing but also on the way the battles play out. Overall, we see the RL agent learns to recognize the most important moves and patterns, obtaining the best results with the exploration vs. exploitation rates of (30, 70), due to a quicker learning rate. We observed how the RL agent learns smart patterns, such as healing himself once its HP life is compromised, using the strongest attacks, etc. Furthermore, the players ratings of the video game experience show that perhaps a better flow may be achieved when the player's opponents is driven by our RL-based agent. Of course, it must be taken into consideration that this study was done in a simple game environment and the introduction of the RL agent to more complex games may not yield the same results. However, the study does show the idea that this sort of approach can be used to achieve game flow and "better" enemies in (turn-based battle) games by having them learn to be "smarter".

This research work could still yield more results in the future. For instance, introducing the agent into a more complex battle system (such as *Dragon Quest XI* or *Pokémon*), in which we expect that the agent would take longer to learn the best actions due to the increase of options. In addition, the battle system presented in this paper become more complex: for example, by adding effects to certain moves, by providing an option for using an item that could make the player/agent stronger, or by allowing them to take less damage.

Furthermore, in the our evaluation study we observed that the RL agent sometimes has a preference for spamming *heal* throughout the battle, so a restriction could be introduced to discourage this behavior. For instance, a limit to the amount of times a move could be used in a row, or by giving the agent a penalty for spamming a move. Moreover, it would also be interesting to evaluate how the RL agent performs when playing with different types of players. A larger range of players with different skill levels and experience could be used, in addition to training the RL agent over the course of many more battles.

8 ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 which is co-funded under the European Regional Development Fund.