Combining Online and Offline Dynamic Difficulty Adjustment Through Procedural Generation in An Infinite Runner Video Game

Cooper Zuranski, Aayush Mailarpwar

Fall 2022

1 INTRODUCTION

In gaming, there is often a barrier to entry. Those who have not played a certain type of game before (or any at all) are often met with an unpleasant experience when starting out. Game difficulties are usually static, and even if adjustable, are discrete (easy, medium, hard). This often leads to a poor match between the challenge of the game and the ability of the player, making steady player improvement inefficient. This is both unsatisfying and frustrating, making the player's overall experience worse. Dynamic Difficulty Adjustment (DDA) alters in-game systems to adapt to the player's abilities and create balance between challenge and frustration [1]. Implementations of DDA fall under two categories which in the context of this paper are referred to as: online (intervention during live gameplay) or offline (using past data about a user to define the behavior of the game) [2]. How the player is modeled plays the most significant role in how the game changes and dictates the most suitable DDA approaches.

DDA can be implemented through the means of Procedural Content Generation (PCG). PCG is defined as the creation of contents (such as textures, terrains, music, or structures) automatically, often with random influences [2]. The primary motivation for including PCG is occasionally to save developer time, but most often the randomness of unique worlds is desirable. PCG can occur in the development process to create static environments to be used within a game, or during runtime. Sometimes runtime PCG is simply the generation of the upcoming level during loading but can also be approached by continuous generation during "live" gameplay. When combined with DDA, PCG can behave dynamically to create surroundings more appropriate to the user from a structural perspective (such as creating/altering obstacles, terrain, or item locations) as opposed to purely numeric (such as altering the player health, enemy damage, or player resource levels).

PCG is inherently present in "infinite runners" - a category of video games best characterized by games such as "Subway Surfers" or "Temple Run." In these games, the player is constantly running forward as the world approaches. The objective is to stay untouched for as long as possible, where the score is proportional to distance traveled and other elements like coins and powerups collected. These are mobile games, generally seen as having a low barrier to entry due to their simplicity. As a result, they often become quite boring once the player becomes moderately skilled– or frustrating due to easy sections directly adjacent to sections that are nearly impossible to survive, thus thwarting user expectations.

DDA-driven-PCG has been explored in previous research studies, however surprisingly none with infinite runner style game implementation. This category of games is a perfect vessel for DDA-driven-PCG as they are very accessible with a low barrier to entry, and inherently procedurally generated. Previously researched DDA approaches are typically either online [1, 3, 4] or offline [5, 6, 7, 8]. Though there is very little separation or definition of these ideas and their respective effects. Online DDA can make the immediate game more enjoyable, however all the learning of the player model is disposed of afterwards. This seems redundant as games are rarely intended to be played once. Offline DDA is generally more model based, creating a representation of the user's profile, whereas online DDA is typically based on probabilistic measures and the occurrence of predetermined events during gameplay. Offline DDA can also be characterized as long term such that a game adapts better to the player the more it is played, using the player's performance to make overarching design choices in how levels are generated subsequent plays.

Our aim is to test the effectiveness of combining both online and offline DDA in a procedurally generated game. We assert that the combination of the two will lead to greater player improvement than either DDA method alone. This improvement in turn makes the game experience more satisfying and enjoyable. We also explore implications of DDA when implemented via an "infinite runner" game. Within this paper "offline" DDA is characterized by varying PCG behavior based on a model of player performance over past sessions, and "online" DDA by varying challenge to keep the player engaged and appropriately challenged, both when doing extremely well or poorly. This is often described as the rubber band effect [1]. The intent of online DDA is to create balance within a game, either emphasizing comfort, challenge, improvement, or a combination of the three. Otherwise, the player's expectations may not be met, immersion may be ruined, or the game may be perceived as unfair [1]. Offline, however, makes a game experience more personable, with less immediate effect on player experience.

The testbed is an infinite runner with wraparound capabilities. That is, the environment is comprised of 5 lanes, the leftmost one is adjacent to the rightmost one. The player can move from side-to-side to avoid obstacles while running non-stop and regain health by collecting hearts from the environment. The player starts with 5 health points (HP) which are removed when the player collides with an obstacle. The game ends when the player runs out of HP. The player's score in each session is defined by the distance they traveled.

To implement offline DDA, information about the player's preference and performance is implicitly collected from gameplay performance data (such as objects collided with). This information will define how the next session's environment will be generated, and how the game itself will behave (e.g., speed adjustment or extra heart probability). This is done through a genetic approach, where the fitness function evaluates the player's performance to determine the characteristics of the current PCG parameters will influence the next generation of PCG behavior, and therefore difficulty of the next game session.

To implement online DDA, the health level and frequency of collision are tracked and used to decide when to intervene and enact "rubber banding." The goal being to keep the player from becoming over or under-challenged. When testing the combined impact of online and offline DDA, the offline player-specific data is used to dictate how the online adjustments will be customized for that player. For example, if a player is doing too well, and they historically struggle with concurrent diagonal obstacles, the probabilities of these being procedurally generated will increase momentarily.

The effects of no DDA, online, offline, and the combination of the two DDA methods will be compared and evaluated. We assess the player's improvement and the uniqueness of content produced in the offline/long-term behavior. To evaluate the effectiveness of one of the DDA methods a ratio of the generation difficulty rating (based off the PCG parameters' values) to the total distance traveled is assessed over successive game sessions. A ratio of difficulty to score is used to evaluate the improvement rate of the player. For example, if the game became more difficult but the player lasted just as long, this shows improvement. The baseline is the player's improvement with no DDA, and will be compared to online only, offline only, and the combination of both.

2 RELATED WORK: MODELING

DDA is inherently related to the field of affective computing – the attempt to characterize emotions by metrics for computing use [9]. This term was introduced by Picard to standardize the use of emotion in computing [9]. Picard's model of affective computing utilizes "sentics," the idea that emotions are communicated through one's motor system (physiologically), and thus, can be captured and mapped. However, depending on the context of the situation, emotions can be expressed more cognitively (internal thoughts) making them difficult to capture. On the topic of video game enjoyability, Picard asserts that a “pleasurable experience” is hard to define concretely as it varies due to the player’s background and/or the complex combinations of aesthetics and mechanics within a game. Csikszentmihalyi proposed a simplified view which only concerns the qualitative balance of game difficulty, referred to as “flow theory” [10]. Flow theory is described as the ideal ratio between challenge and skill for a given player, where any values above this threshold create anxiety and frustration, and any values below result in boredom [10]. Discrete difficulty level selection helps with this balance in games, but is a broad representation of many players, and does not address spikes in difficulty during gameplay or specific features. When applied to gaming, affective computing is realized as the “affective loop,” where a collection of sentic metrics are used to affect the user experience. The central idea behind DDA is the affective loop supporting the ideas presented in flow theory.

A DDA model that realizes the affective loop is the *Experience-Driven Procedural Content Generation* (EDPCG) model [2]. Yannakakis defines EDPCG as Player Experience Modeling (PEM), content quality, content representation, and the content generator [2]. PEM represents the player’s feedback and can be defined as subjective (such as questionnaires), objective (such as physiological readings), or gameplay based (such as statistics from the play session) [2]. The model is derived by identifying important aspects of the game: either following a “bottom-up” approach by using player performance metrics directly, or a “top-down” approach which models the player pseudo-psychologically by creating inferences about their personality and preferences [11]. According to the analysis provided in [2] the PEM is used to evaluate the generated content

- directly using specific metrics about the generated environment such as feature count
- through simulation of static agents playing and comparing their performances, or
- interactively by evaluating the content during live game play.

Interactive evaluation can be implicit or explicit. Implicit evaluation is achieved by using the player’s performance data. Explicit evaluation is performed by directly asking the player about their experience through questionnaires. Both have disadvantages: Implicit data can be noisy and of low-resolution, while explicit collection is intrusive on the gameplay and overall experience [2]. These evaluation techniques extend to other applications beyond gaming, and can be used in intelligent tutoring, product recommendations, adaptive instructions, and automated shopping [11].

2.1 ONLINE DDA

Hunicke used *Half-Life* as a testbed to evaluate if online DDA was noticeable, affected the player’s performance, or significantly affected the player’s perception of the gameplay [1]. If the player’s probability of death became greater than 40%, the intervention mechanism would act by altering enemy spawn patterns and damage levels – the idea being that the user would not likely notice. The findings reported that “post-play evaluations of adjusted games revealed no significant correlation between adjustment and enjoyment for novice players. However, (result) trends indicated that expert players reported slightly elevated levels of enjoyment.” Instead of intervening based on the player’s performance, Lopes et al altered the PCG of a play session based on the player’s current situation – location, time of day, or time available to play [3]. Due to the emphasis on context-driven and casual gaming, the testbed was a mobile game. The constraints of the player’s current time were accounted for by attempting to condense the full gaming experience into the time available. The session was divided into discrete chunks of time, each with a goal of progressively more challenging PCG, mimicking the difficulty goals of a longer game. For each subsequent chunk, the difficulty would adjust to the player’s performance to ensure the appropriate challenge level was delivered. This led to a more satisfying experience in short term sessions [3]. An additional approach by Silva et. al. alters the behavior of an opponent agent based on the player's performance [12]. As the player performs better, the reflex agent alters its attack and defense strategy by incorporating more abilities (such as spells and tracking) according to a mapping between sets of these abilities and a predefined equation the player's performance. This approach of altering behavior instead of variable attributes is more difficult to implement and less common, but it can create a more drastic change and does not rely as heavily on the specific intervention event [13].

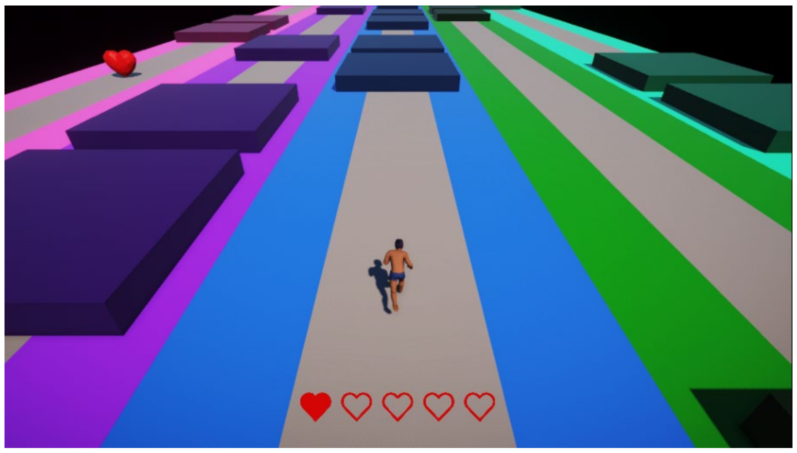
2.2 OFFLINE DDA

Offline DDA approaches differ from online DDA approaches in that they optimize for adaptation over multiple play sessions – and do not consider "real-time" intervention. This can provide much more appropriate and personal PCG styles adapted to each individual user [6]. Pedersen et al gathered explicit user feedback through questionnaires about perception (frustration, fun, and challenge) which were used to optimize the structural features of the game *Mario* (such as gap placement) within an artificial neural network [6]. Subsequent levels for a specific user are generated with a genetic algorithm where the fitness function is the difference between the player's reported perception and the neural network's output. Similar attempts to adjust difficulty through structural generation are outlined by Jennings-Teats et al using *Polymorph* [5] and Huber et al for a VR exercise game [7]. The former adjusted subsequent levels' difficulty based on explicit user feedback by using a Ranking SVM, which ranked different generation combinations [5]. The latter adjusted the generation of room locations within maze levels based on biodata and explicit feedback by using Deep Reinforcement Learning [7]. The goal was to create an appropriate workout for the player according to their current capabilities, and to create increasingly personalized levels the more they played. Similarly, Wheat et al had a pre-chosen difficulty level, where based on the player's performance, the PCG would be altered to deliver an environment fulfilling the requested difficulty [4]. Initial gameplay data was used to create agents which replicated a specific player's performance. Using Interactive Evolutionary Computation, these agents would evaluate the contents created by the PCG generator to avoid the need for the player to evaluate many combinations of the PCG generator's parameters. The next set of PCG parameters were generated using the difficulty of the current content experienced by the agents as the fitness function to a genetic algorithm.

3 METHODOLOGY & IMPLEMENTATION: TESTBED

The testbed used in the present study is an infinite runner game implemented using the Unity [8] game engine. The game layout consists of a 5-column grid where the player can move left and right one column at a time with "wraparound" between the leftmost and rightmost columns (Fig. 1). The generation occurs one row at a time at the top of the screen and moves towards the player to emulate the effect of running forwards at a constant speed. The objective of the game is to cover the maximum distance while avoiding obstacles to stay alive as long as possible. The player starts with 5 health points (HP) which are lost when the player hits an obstacle. Subsequently, the player is invincible for a duration of 3 rows to allow the player to recover from a previous mistake. The player can regain HP by collecting "hearts" spawned into the environment.

Throughout the present paper many aspects of the PCG are referenced in terms of a "chunk," which we define as the intersection between a column and row. As the world is generated one row at a time, the properties of each chunk are what fulfil the PCG. Chunks are either "safe" or "dangerous" where dangerous chunks have obstacles while safe ones do not. To prevent the generation of impossible barriers where the gaps of one row line up with the obstacles on the next across all 5 columns, each row can have a maximum of 2 danger chunks. As there are still situations where impossible paths can be generated over the duration of 3+ rows, a pathfinding algorithm is used to ensure that there is a valid path from the player's current location to one of the safe chunks in the row currently being generated (Fig.2). If none exists, another row is generated which has the exact same properties (number of HP, number of danger chunks, etc.). This generation can happen multiple times before spawning the new row into the game, even at high speeds.



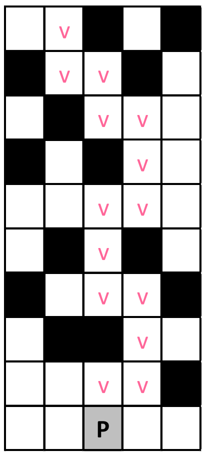


Fig. 1 Left: Screen capture of actual game environment as seen by the player.

Fig. 2 Right: Representation of an example world. P is player, v is a valid path, blacked-out cells are danger chunks.

3.1 <u>DATASET</u>

The data set includes the variables used within each game session to generate the environment. Most of these are concerned with the probabilities of features being generated within one row and are labeled "PCG variables."

- **'DC' - Danger chunk probability**: The probability of two danger chunks within a single row. There can be 0, 1, or 2 danger chunks within a single row. ¾ of 'DC' is the probability of only one danger chunk spawning within a row, while the remaining value in the probability mass function (PMF) results in no danger chunks (Eqn. 1). Higher values are more difficult, as more frequent obstacles limit the players' options of unobstructed paths, forcing them to plan farther ahead.

$$P_X(x) = \begin{cases} DC & x = 2 \\ \frac{3}{4}DC & x = 1 \\ 1 - \frac{7}{4}DC & x = 0 \end{cases}$$

Eqn. 1: PMF for 'DC'

- **'HS' - Heart spawn probability**: The probability of a heart being placed within a single row. There is no more than one per row, and it is placed with equal probability on one of the safe chunks. This value is generally very low, otherwise the player would never lose. Higher values are easier, as the more frequent hearts generate, the more chances a player must collect them and thus stay alive for longer.

- **‘S’ - Speed**: The speed of the constant forward movement of the player in rows per second. Although this is not directly part of the generated environment, it is procedurally varied along with the other variables. Faster speeds are more difficult and require higher dexterity from the player as they will have a shortened response time to avoid obstacles.
- **‘L’ - Linearity factor**: The probability of all danger chunks aligning vertically with the ones in the prior row. 3*L represents the probability of 1 danger chunk aligning (Eqn. 2). L generally defines how challenging the geometry of the path through the obstacles is. Lower values are more difficult as the more frequently danger chunks align, the less often the player has to move side to side to avoid obstacles. A selection over the PDF of L is dependent on the present danger chunks, and it therefore implemented as the “guaranteed” chunks to align in subsequent rows (ranging between 0-2 inclusive). That is, a selection of “1 guaranteed linear aligning chunk” when 2 are present would result in 1 aligning with one in the previous row, while the other present danger chunk randomly may or may not align. More examples of edge cases exist but are trivial testbed details and irrelevant to the focus of this work.

$$\boldsymbol{P_X(x) = \begin{cases} L & x = 2 \\ 3L & x = 1 \\ 1 - 4L & x = 0 \end{cases}}$$

Eqn. 2: PMF for ‘L’

The selection using each probability mass function is implemented by creating a distribution of ranges for each value in a Cumulative Distribution Function, then randomly generating a float between 0-100, where the range the float is within is the selection made for generation of the subsequent row. The probability distributions shown above were defined previously within the testbed in order to control PCG in a similar fashion to that of a well-established infinite runner game (e.g., Subway Surfers), and are arbitrary testbed details. They are trivial within the context of the work presented in the present paper and included solely as implementation details. Within the presented work, the significance of the PCG variables introduced above is how they are used within offline DDA to evaluate content difficulty. Other data involves a collection of the player’s performance, which is used within the genetic algorithm’s fitness function (along with the content difficulty), and when evaluating results. The use of these metrics is explained later within the context of the genetic algorithm. These metrics include:

- **‘DT’ Overall Distance Traveled:** The overall distance traveled within a session in terms of chunks. This aligns with the player’s main objective within the game. Although it is the most significant metric, it is only indicative of the final result of a session, not the player’s performance throughout.
- **‘HL’ Average Distance Between Heart Loss:** The average distance between instances of the player colliding with obstacles in terms of chunks. Better players do not rapidly decline – they tend to endure any spikes in difficulty, declining slowly. Additionally, average distance between HP lost indicates performance within a session, not merely the final result (as DT does).
- **‘PH’ Percentage of Hearts Collected:** The percentage of hearts collected when the player is not at maximum HP. The only hearts included in this calculation are ones that the player actually had the opportunity to collect. Since the player cannot collect hearts while at maximum HP (nor do they have reason to), these are disregarded within the calculation. Inclusion of PH is partially motivated by the fact that many video games’ scoring methods include bonuses for items collected. This further indicates the ability of the player to plan quickly enough to find a safe path leading to heart collection, a trait which is associated with higher scores.

HL and PH are also independent of play style. Some metrics commonly used to evaluate DDA can vary as a result of the player's style, such as jump amount or move frequency [6]. Some players may simply prefer to stay more towards the middle of the environment, but this does not mean they are less skilled.

The prior mentioned PCG variables' values for a given session are stored along with the player's metrics for that session. A total of 32 sessions are considered for each separate DDA approach being evaluated (i.e., no DDA, online only, offline only, and the combination of online and offline).

3.2 OFFLINE ONLY DDA

The offline only DDA sessions only affect the PCG variables before the next game session begins, not during. Genetic algorithms (GA) are commonly used for PCG, and used within our implementation of offline DDA to determine how the PCG variables should evolve over subsequent game sessions [7,8]. Our approach is to determine what variance in features suits an individual player better by evaluating their response to adjustments of the PCG variables. The overall goal being to enhance the progression of their skills by providing constant challenge at an adequate level for the specific player. An advantage of a GA approach is that the randomness introduced when splicing and mutating the PCG variables limits cyclic behavior. Cyclic behavior is observed when the player becomes pigeon-holed, continually altering the same PCG variable.

We use an initial population of four random sets of the PCG variables (DC,HS,S,L), each within a range of "medium" difficulties determined by early play-testing in order to provide a reasonable starting point. This is done because fully-random values can cause extreme PCG variable values, which may cause the player to die instantly, or not at all. This does not provide an appropriate challenge, as is the goal. The player will play one session with each of the four individuals in the population. 32 sessions are used to allow 8 generations of the GA to execute. Each generation has the same population size of 4 PCG variable sets. This is relatively small when compared to many examples of GA implementation but is necessary to limit the amount of time required by testing candidates, as allowing 8 generations to execute with a larger population size would take hours and be a burden to the testing subjects. As one of the purposes of DDA inclusion within a video game is to retain the player's focus and satisfaction, this burden would likely skew the results. Additionally, pruning is introduced to enable faster convergence, which is explained in the context of the Fitness function. The approach taken for the GA implementation is similar to the work presented in [4], where the fitness function is comprised of Relative Performance and Content Difficulty:

- **Relative Performance:** The player's relative performance within the current population of PCG variable sets. For Relative Performance (Eqn. 3), the metrics composing performance mentioned earlier are first standardized within the current population. This process of making the metrics "relative" to the current population allows them to be expressed in terms of percentage of the average for said relevant population. This is done as DT values may have magnitudes in the hundreds, while HL and PH are typically in the magnitude of tens due to the nature of what they represent. This difference in magnitude would make their impact on the performance rating insignificant (if done without normalizing). This "relative" approach is acceptable as the process of choosing the next generation of the relevant session is independent of all other generations. The "relative" performance calculation is not stored.

Once all the individual metrics are standardized within the relevant population, they are added in a weighted sum according to their significance. The ranking in magnitude of these weights were determined by considering the objective of the game, and which metrics best compose successful player performances in pursuit of that goal. Since the testbed's goal (and that of any infinite runner video game) is overall distance, DT is inherently the best indicator of performance. Higher HL values indicated better performances than PH in early playtesting. Therefore, HL is considered to be the second most significant parameter. The hyperparameter weights applied balance the significance of each metric in this ranking. Beyond the aspect of ranking, the weight values are arbitrary. Play-testing was considered to empirically determine weights, but the idea of performance within any domain is subjective. The Relative Performance metric solely exists to give a baseline for comparison between different sessions, then observing the trend of its change. Not to determine definitive values.

$$Relative\ Performance_n = 50\left(\frac{DT_n}{AvgDT_G}\right) + 35\left(\frac{HL_n}{AvgHL_G}\right) + 15\left(\frac{PH_n}{AvgPH_G}\right)$$

Eqn. 3: Session n's performance relative to current population of 4 PCG variable sets, 'G'

- **Content Difficulty:** The content difficulty rating for each PCG variable set in the current population. Performance alone is misleading. For example, if the content generated within one session is exceedingly easy, the player will likely have high performance metrics. Within a separate session however, the generated content is significantly more difficult, and yet the player's performance metric values are the same. Clearly, the player showed better skillfulness and execution in the latter. The content difficulty calculation (Eqn. 4) limits this impact. The content difficulty is calculated by summing the values of the relevant PCG variable set. Speed is normalized relative to the maximum speed such that all PCG variables contribute towards one quarter of the content difficulty value. The complement of L and HS are used as their values being lower means the content is more difficult. The maximum difficulty rating is a 4, where a higher value is more difficult.

$$ContentDifficulty_n = DC + (1 - L) + (\ 1 - HS) + \frac{S}{125}$$

Eqn. 4: Session n's difficulty rating. Between 0 and 4, where a higher value is more difficult

The fitness of a specific PCG variable set is a judgement of the player's overall performance within the relevant session. However, the player's performance is subject to the difficulty of the content generated. As a result, the fitness of a session is a ratio calculated by the relative performance scaled by the session content difficulty. This discourages the GA from only creating simple content, favoring content which is easy enough that the player performs well, but is still challenging to them. This process of providing an adequate challenge aligns with the aim of DDA enhancing the player's skill progression. Within the fitness equation (5), the complement of content difficulty is used. For example, an easy level (low Content Difficulty) should generate a lower fitness value than a hard level (high Content Difficulty) when the player performs extremely well. I.e., high performance within an easy level is not as significant as high performance within a hard level. Relative performance values average 100, while content difficulty is between 0 and 4. Directly using the content difficulty rating would lead to a narrow distribution of fitness values for the current population. To combat this, the content difficulty rating is weighted arbitrarily.

$$Fitness_n = \frac{Relative\ Performance_n}{5 \times (4 - ContentDifficulty_n)}$$

Eqn. 5: Session n's fitness relative to the current population

Using the fitness values generated for the current population, probabilities for selection are assigned accordingly. Pairs are selected according to this probability distribution. The new population is created after crossing over pairs of parents. Then, with a 50% probability each individual will have one PCG variable (selected with equal probability) mutated. This probability is relatively high, but with only 8 generations of testing, change needs to be significant each time. Additionally, when trying different GA configurations for mobile-game DDA in [17], three of the top 5 configurations used a 50% mutation rate. This was also the value used for mutation in [4]. The mutation process randomly alters a value by ± 30% of its current value. This is enough to create an effect but not enough to affect the gameplay in an outlandish manner (i.e., random value selection may allow hearts to spawn on nearly every single row, such that the player would never lose).

The GA may still create PCG variable sets which cause generation at an inappropriately low difficulty for the player. To combat the occurrence of such individuals within the population and converge faster, pruning is introduced on a per-player basis. Any individual PCG variable set which results in a generation where the player obtains a DT > 800 is removed and replaced by an set of PCG variables 30% more difficult than the pruned PCG variable set. Players who surpassed roughly 800 chunks would typically end up playing a single session for over 15 minutes, so this was determined to be the upper threshold. Similarly, a DT < 50 causes pruning to replace the individual with a 30% easier composition. This introduces a behavior reminiscent of simulated annealing as pruning occurs frequently at first, then decreasing drastically as the PCG variables become more appropriate for the player.

3.3 ONLINE ONLY DDA

Our online only DDA approach relies on the concept of rubber banding [1]. This is done by keeping a “rubber band” around the player and the opponent to prevent them from spreading too far apart. In our case, the opponent is the generated environment. In online DDA, we implement rubber banding by adjusting the game session’s PCG variables momentarily according to the performance of the player. The same set of PCG variables is altered in the online and offline cases, but here each session starts with the same values of the PCG variables. All alteration only happens temporarily within a single session. Intervention in an online DDA trial is implemented by monitoring the player with counters for each intervention criteria.

We define the response to the upper and lower limits of performance as down-banding and up-banding respectively. The intervention criterion for down-banding is when a player has traveled 50 or more chunks without hitting an obstacle or has remained at or above 3 HP for a duration of 100 consecutive chunks. The intervention criterion for up-banding is when a player currently only has 1 HP remaining or has lost HP within 6 chunks of the last loss. The intervention criteria for any game are somewhat arbitrary in nature, as they are specific to the individual game [1, 7, 13] and dependent on the policy of the DDA (comfort, discomfort, improvement, etc. [1, 11, 14]). This being said, the intervention criteria and subsequent actions for the present implementation were determined as a result of similar research [9, 12, 13, 14].

For the online only implementation, there is no bias as to which PCG variable is changed. The chosen variable is modified linearly by 30% of its current value and lasts for 30 chunks, then the PCG variables return to normal. The direction in which the PCG variables are modified is determined by the direction of the banding. If down-banding is occurring the game session should become more challenging, so variables such as ‘DC’ would increase while ‘HS’ would decrease. The durations used for intervention detection and modification were determined empirically to find a balance between enabling and disabling the player appropriately.

Online DDA is implemented by keeping counts for each criterion, checked each time the player enters a subsequent row. If a criterion is met, banding is initiated, and the difference made in the PCG variables' value is stored. Banding can't be reinitiated for the same criterion until the resulting banding ends, and the difference is restored. It will immediately occur again for the same criterion if it is still met. This indicates the player still needs the additional ease/challenge presented, as the up/down banding was not enough to bring them to an appropriate performance level. Additionally, multiple criteria can initiate banding at the same time. For example, the player has traveled 50 chunks without hitting an obstacle, initiating one instance of down-banding. Then shortly after, has reached 100 chunks traversed above 3 HP, causing a separate instance of down-banding to occur simultaneously. If the player is performing well in two areas, an additional challenge is appropriate.

The online only approach to DDA is good at retaining player engagement, but it is not customized to the individual player's style or abilities. A skilled player will stagnate and become bored with the predefined difficulty level. To maximize improvement of the player's abilities, personalization and long-term engagement are required in addition to short-term engagement, so the online and offline DDA are combined.

3.4 COMBINED OFFLINE AND ONLINE DDA

Here, the offline DDA aspects are implemented the same way as described above. However, the online DDA approach is modified. In the previous online only DDA implementation, the PCG variable that is altered during an intervention occurrence is chosen randomly. In the combined offline online DDA implementation, the variable is selected based on which PCG variable affects the player the most according to their performance data. Relative differences from the current PCG variable values and the average values of all PCG variables in the original generation are calculated before the beginning of the session. The largest relative difference is used to select which PCG variable will be modified during rubber-banding. The direction of modification depends on whether up or down banding is occurring. For down-banding, the combined algorithm would alter the variable which most affects the player such that is makes the game more difficult. For example, a higher 'H' value makes the game easier, but a higher 'D' value makes the game more difficult. This creates a heightened level of challenge which directly targets the player's weak spots, forcing the player to improve by applying brief rubber banding interventions.

$$Max\left(\left|DC_n - DC_{G1Avg}\right|, \quad \left|HS_n - HS_{G1Avg}\right|, \quad \left|L_n - L_{G1Avg}\right|, \quad \left|\frac{S_n - S_{G1Avg}}{125}\right|\right)$$

Eqn. 6: Session n's PCG variable to alter during banding. G1Avg is the variable's average value within the first generation.

All three DDA approaches are compared to the baseline, where no DDA is used, such that all PCG variables are constant over all game sessions. Players may still improve naturally but not due to any intervention from the game.

3.5 TESTING

This research aims to evaluate the effectiveness of combining both online and offline DDA in a procedurally generated game. We assert that the combination of the two will lead to greater player improvement than either DDA method alone. This improvement in turn makes the game experience more satisfying and enjoyable. In the current study, we utilize the gameplay-centered approach for player modeling, which is founded on the idea that play actions are connected to the player's overall experience since games affect the players cognitive processing, and in turn, emotions [2]. Our feedback evaluation is implicit, as explicit evaluation is often used in studies surrounding player enjoyment, and our research is based on affecting performance – enjoyment being the byproduct. Furthermore, a player's perception is not always accurate to the reality of their performance, especially when only playing the game a handful of times. We therefore choose to collect player response implicitly by judging their performance instead of asking explicit questions.

A participating player plays 32 game sessions under each form of DDA: no DDA, online only, offline only, and online-offline DDA combination. The order that each DDA implementation is tested is random to limit the influence of the player's improvement due to simply playing the game more times. The environment created in Unity automatically shuffles and tracks all of the DDA trials' execution and stores the player's data in a .csv file to be processed in results. Each player is allowed to play 1 session of the ”No DDA” variety to familiarize themselves with the testbed. Each player's age and level of video game experience is recorded as well. Players are asked about their observations after each DDA type is completed, although they are not informed that anything is changing. This is used for discussion purposes.

4 <u>RESULTS</u>

For the testing stage, 6 participants were recruited with an average age of 19 (referred to as A-F throughout discussion). None had played the game presented in the testbed before, but all had played an infinite runner before. 50% of participants (B, D, E) claimed to play video games frequently (more than 3 times per week).

Performance cannot be directly compared between players, as each performs at a different level. However, we can measure the difference in improvement levels from the baseline of each player to see which DDA implementation caused the greatest level of improvement. Using the ratio between performance and generated content difficulty (Eqn. 8) as the dependent variable, the passing of time over each subsequent session shows the rate at which players improved. To find these improvement rates, each player's data from each DDA implementation trial is linearly interpolated separately. The resulting slopes for each DDA implementation are then averaged over all players to evaluate which had the greatest effect. Focusing on the difference in rates eliminates the effect of individual player's different skill levels.

However, the performance equation used prior (Eqn. 3) exists solely for determining fitness of an individual “relative” to the current generations' population in order to evolve the next generation (within the offline DDA implementation). To get an accurate interpretation of the player's progression throughout the 32 sessions within each DDA implementation's trial, the player's performance within one session is made relative to all sessions within that DDA implementation. It is then normalized for the difficulty of each session for the same reasons mentioned prior before it is linearly interpolated to find the rate of change.

$$Performance_n = 50\left(\frac{DT_n}{AvgDT_{DDA}}\right) + 35\left(\frac{HL_n}{AvgHL_{DDA}}\right) + 15\left(\frac{PH_n}{AvgPH_{DDA}}\right)$$

Eqn. 7: Session n's performance relative to all 32 sessions within the respective DDA implementation. Similar to Eqn. 3, except averages (avg) here are taken over all 32 sessions within one DDA trial type instead of only one generation.

$$NormalizedPerformance_n = \frac{Performance_n}{ContentDifficulty_n}$$

Eqn. 8: Session n's performance relative to all 32 sessions within the respective DDA implementation, where Performance and ContentDifficulty are Eqn. 3 and Eqn. 4.

Recorded data was extremely noisy. This is indicated after linear interpolation was performed for each DDA type per participant. Resulting average R^2 values for each DDA type ranged between 0.027 and 0.112. The average normalized performance in all four DDA types were very similar, ranging from 30.9 to 31.7. This indicates that shuffling the order which participants played each DDA type did successfully negate the effect of more total playtime contributing to improved performance, which allows us to compare each DDA type independently.

Slope (Normalized Performance per session)								
	A	B	C	D	E	F	Avg Slope	Avg R^2
No DDA	0.2043	0.5283	0.1551	0.0589	0.2134	0.0119	0.195317	0.027783
Online	0.0625	0.4771	0.1918	0.0283	-0.18	-0.5059	0.0123	0.027833
Offline	-0.5946	-0.485	-0.2743	1.1611	0.7722	-0.2147	0.060783	0.0721
Combined	-0.8722	-0.3696	-1.1014	-0.3255	-0.0665	-0.7685	-0.58395	0.112117

Fig. 3: Table of linear interpolation results for each DDA type.

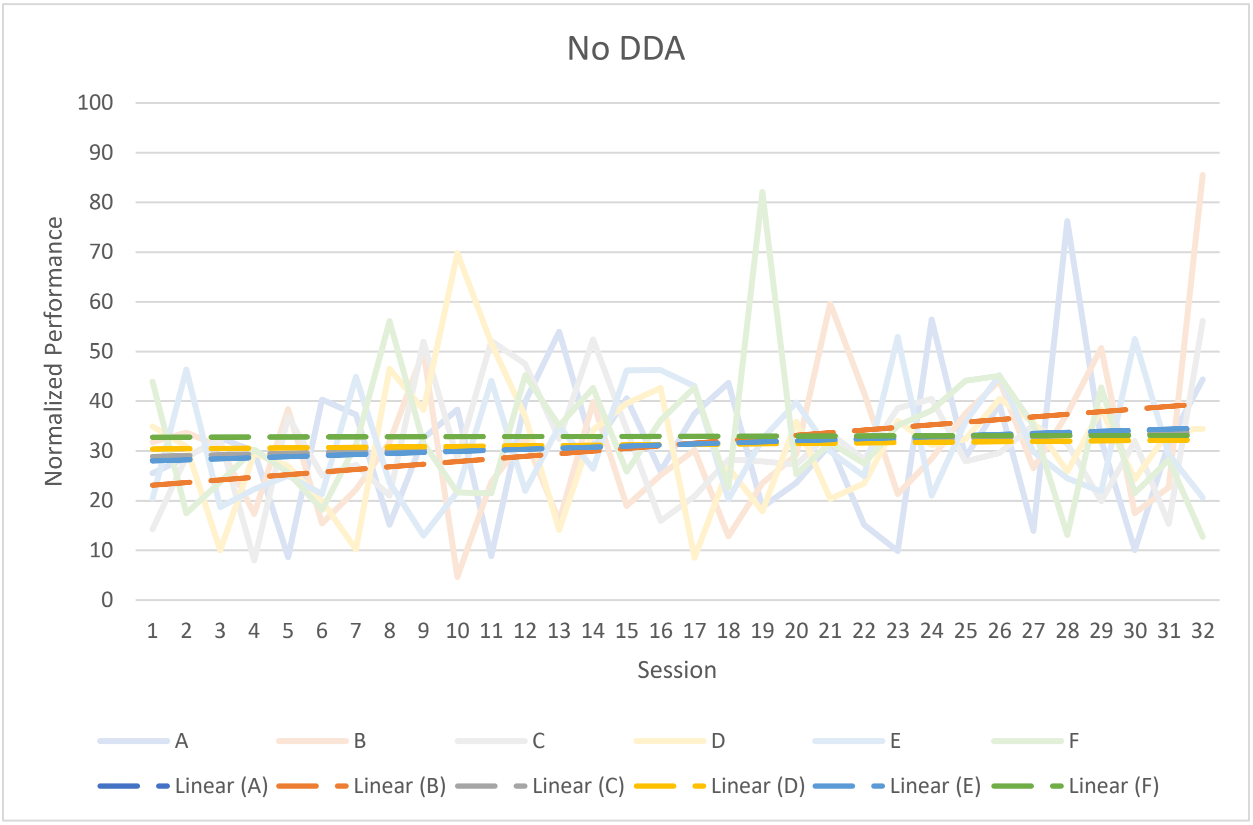


Fig. 4: Graph of recorded performance for each participant's performance per "No DDA" session and resulting trendline.

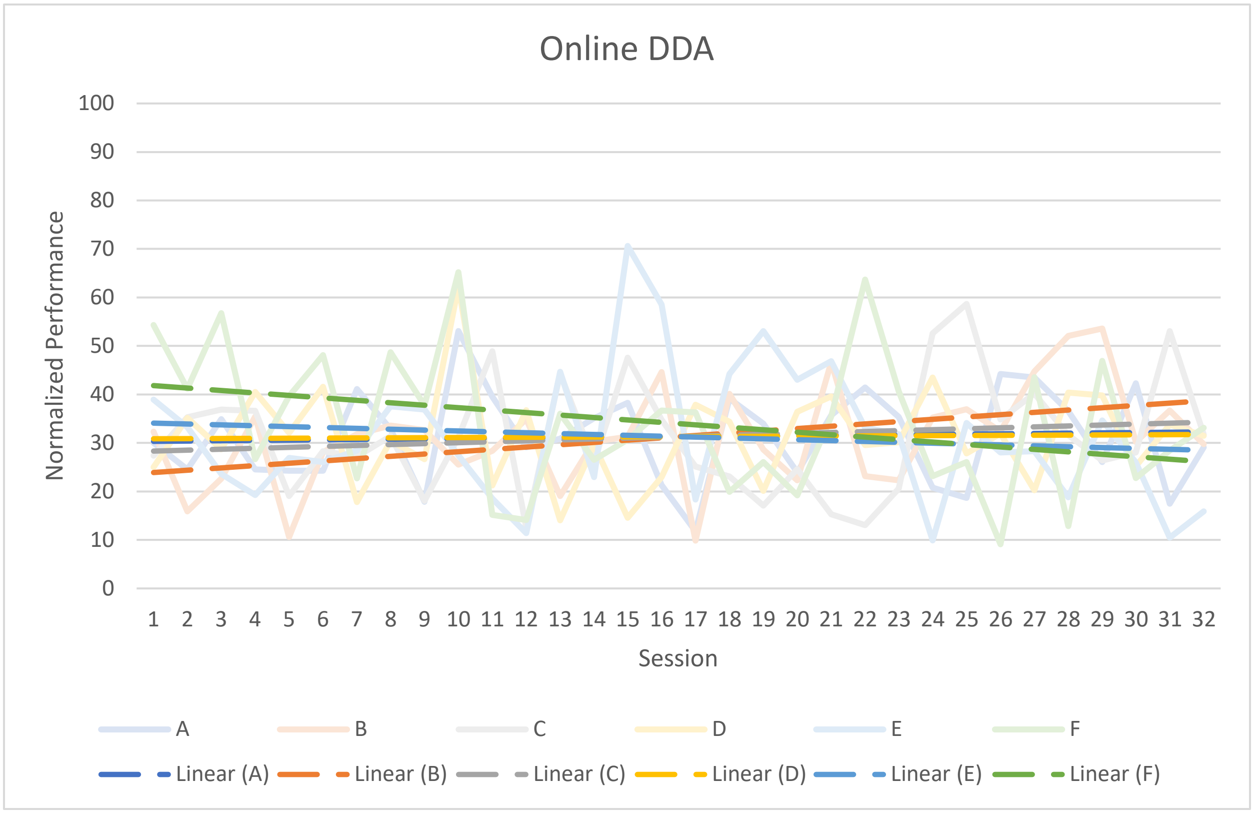


Fig. 5: Graph of recorded performance for each participant's performance per "Online DDA" session and resulting trendline.

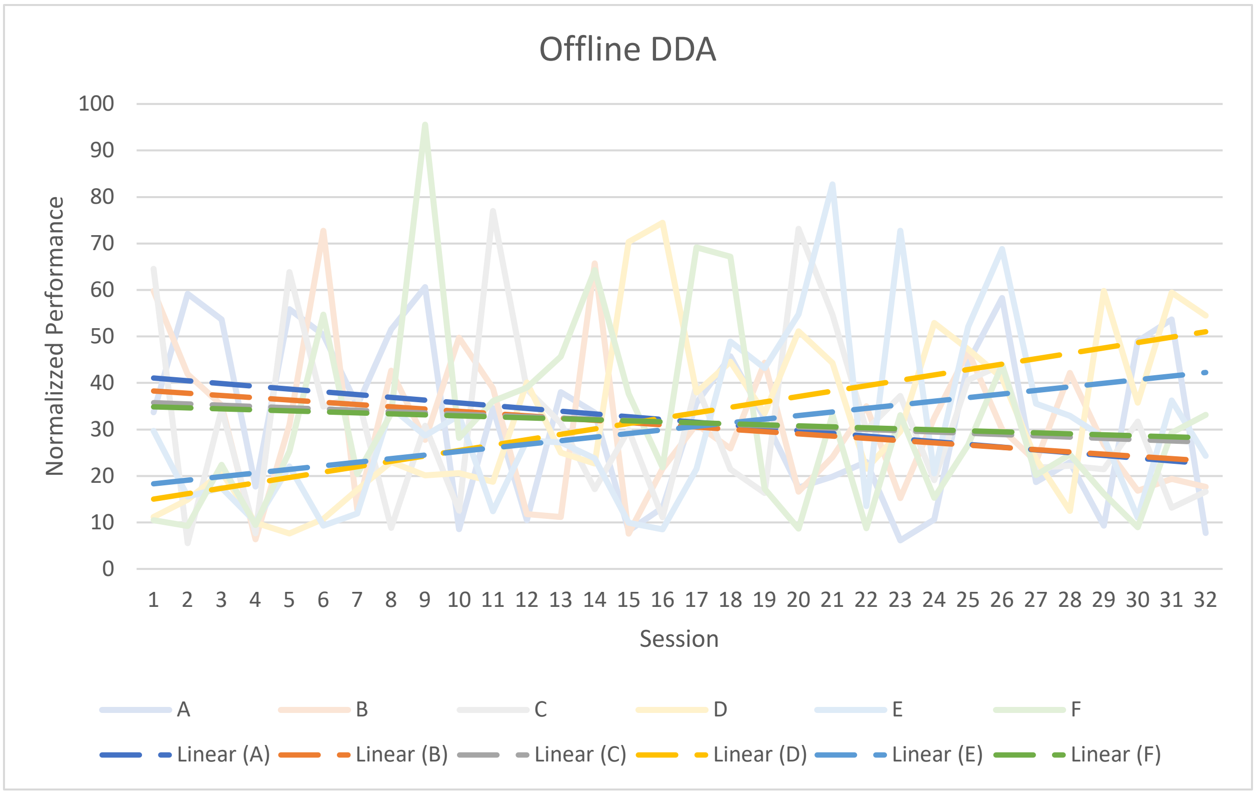


Fig. 6: Graph of recorded performance for each participant's performance per "Offline DDA" session and resulting trendline.

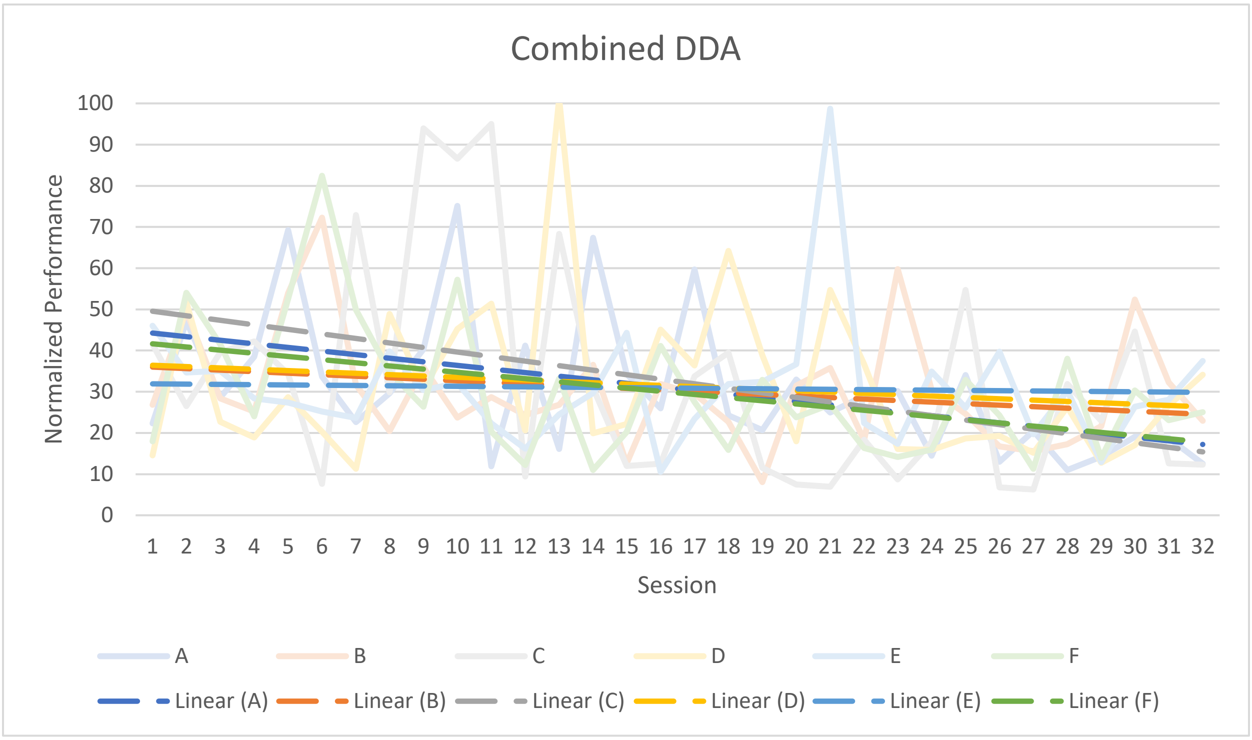


Fig. 7: Graph of recorded performance for each participant's performance per "Combined DDA" session and resulting trendline.

However, these results do not confirm the proposed hypothesis that combined DDA results in more improvement than online or offline alone (Fig. 3). The baseline of no DDA showed the highest rate of improvement, with an average improvement rate of 19.5% (gain in Normalized Performance) overall. The combined DDA actually indicated participants performed worse, with a 58.4% drop in initial performance. Online and Offline only DDA resulted in an average rate of 1.2% and 6.1% increase respectively.

Online DDA had an average sample variance (Fig. 8) lower than the baseline's, indicating some success in "rubber banding" implementation. Participants performed significantly above and below their average less frequently. However, both DDA implementations including offline DDA indicate a much greater variation in performances. This is expected at first as the genetic algorithm initially jumps around the solution space with pruning occurring frequently. Though this variation did not seem to lessen much in the online-only DDA, the average sample variation over the last 2 epochs of the combined DDA was 123.7, which is less than that of the baseline. In all of the prior mentioned cases, a lower variance indicates the testbed adapted to the player better.

Sample Variance							
	A	B	C	D	E	F	Avg
No DDA	225.2324	241.5968	135.2681	160.4824	128.9793	190.5948	180.359
Online	90.11428	108.3004	142.9326	95.50175	195.089	209.7708	140.2848
Offline	330.3056	274.365	354.3651	343.5482	370.5207	422.8741	349.3298
Combined	292.6315	197.8654	690.5108	365.2566	224.3953	258.0595	338.1199

Fig. 8: Table of sample variance for each participant and DDA type.

5 DISCUSSION

There are many limitations to research performed, many of which we believe resulted in the inability to make many conclusions. However, as it has been proven elsewhere that DDA can lead to elevated improvement rates [1,5,9,12,13,14], and our results indicate that each DDA method led to less improvement than the baseline, it is inferred that our results are inaccurate to begin with. I.e., our work certainly does not prove that a combination of online and offline DDA causes greater rates of improvement but doesn't directly disprove this either. We presume the following factors led to this inconclusiveness:

- **Testbed details:** The wrap-around mechanic convoluded testing. Most participants had not seem something like it before, and even if well-versed with infinite runner games, struggled with it. Although the extra strategy and geometrical-thinking it introduces into gameplay is fascinating, it was distracting from the evaluation at hand. Additionally, this uniqueness makes the reults less comprable to that of a regular infinite runner game. In general, making the testbed required many design-decisions that could continually separate it from commercial games, which raises questions of validity. Participants also stated that the cool-down feature was implemented poorly, as it was based on distance instead of time, and would be inadequately short at high speeds.
- **Dataset size:** Due to the time constraints of this work and the time requirements of participation, we were only able to recruit 6 participants. Any outliers in the collected data greatly affected any wholistic observations made. Regardless of this, a much larger dataset would be needed for any conclusions to be statistically veriafiable.
- **Participation length:** Completion of all four types of DDA with 32 sessions each took participants about 2.5 hours. In this timeframe, participants reported fatigue regardless of whether or not they found the game to be engaging. This very well could influence the results of later sessions, with fatigue leading to more mistakes and less motivation.
- **Number of sessions:** 32 sessions were chosen as a tradeoff between number of epochs and participation length. Even with this number of sessions affecting participation length too greatly, it was still a relatively short number of epochs for a GA to undergo – even with the unique implementation methods disscussed prior. It is uncertain whether or not the offline-only DDA would have settled given more time.
- **Parameters:** Furthermore, the GA seemed to consistently overcorrect when on its own. When pruning occurred due to a PCG variable set being too easy, this sensitivity would often immediately result in a PCG variable set of the other extreme. Likewise in online DDA, the interventions that occurred during banding were abrupt according to all participants. When interventions would initiate or end, the abrubt change in the environment would often catch players off-guard.
- **GA limitations:** Although a GA is common in the combination of PCG and DDA [7,9,11] it does not directly account for the relationships between different parameters in the fitness function. Not all factors contribute equally, and intra-relationships are far more complex (E.g., a high 'S' value is quite easy given it is paired with a reasonable 'L'). Using a GA in combination with a neural network such as in [6] would have been a better implementation.
- **Difficulty quantification:** Similarly, the prior-mentioned intra-relationships cause features to contribute to content difficulty in an uneven and non-linear manner. E.g. a 30% difficulty increase in each PCG variable makes the game far more difficult than 30%, or even 120%. This was noticable during testing, and lead to rapid jumps in difficulty experienced – though not reflected in our calculations.

Simulation using agents of different skill levels (as presented in [5]) would have allowed for appropriate determination of such hyper-parameters. And as stated prior, adaptation of an open-source testbed, algorithms which account for feature intra-relationship, and a larger dataset would be necessary in any future work in order to get accurate results and the ability to prove them.

6 CONCLUSION

In gaming, a poor match between the challenge of the game and the skill of the player can lead to boredom in one extreme, and frustration in the other. While this results in a worsened experience in casual gaming, in educational or exercise games (A.k.a. “serious” games), inadequate difficulty results in lower effectiveness [1,2,7]. The creation of better DDA systems is a key aspect both in entertainment accessibility, and serious gaming. DDA implementations are typically either online or offline, yet there is very little separation or definition of these ideas and their respective effects.

Our work combines methods from both online and offline DDA with the goal of creating a greater rate of player improvement than either DDA implementation alone. We have created a testbed within the Unity game engine in the style of an infinite runner with the ability to control DDA as online, offline, neither, or the combination of the two. The online DDA implementation intervenes when a player is playing poorly or exceptionally well, applying an appropriate level of challenge to foster the player’s engagement and improvement. The offline DDA implementation uses a GA with a fitness function defined by the player’s performance to evolve the parameters (referred to as PCG variables) with which subsequent sessions are procedurally generated. The combined DDA implementation determines and uses the PCG variables which affect the player the most to choose specific features of the game to alter during online intervention.

All three of these DDA implementations were tested (along with a baseline) by each study participant, though there were not enough participants to have confidence in the results. In a blind test of each DDA type presented, participants unanimously chose the combined DDA as both the most fun and appropriately-challenged type, though nothing can be proven about their improvement levels. It was also discovered that the presented combined DDA implementation was consistently able to begin converging on an appropriate challenge level for the player within the last 2 epochs. Though the methodologies presented seem to be promising, there are many limitations in the implementation details which further hinder the ability to draw conclusions from the presented work. Future work would need to consider appropriate determination of hyper-parameters, adopt a standard testbed, better account for feature intra-relationship, and utilize a much larger dataset to be able to get and prove accurate results.

7 TABLE OF ACRONYMS

Acronyms	**Full Form**
DDA	Dynamic Difficulty Adjustment
PCG	Procedural Content Generation
HP	Health Points
EDPCG	*Experience-Driven Procedural Content Generation*
PEM	Player Experience Model
VR	Virtual Reality
SVM	Support Vector Machine
DC	Danger Chunk Probability
HS	Heart Spawn Probability
S	Speed
L	Linearity Factor
PMF	Probability Mass Function
PDF	Probability Distribution Function
CDF	Cumulative Distribution Function
DT	Overall Distance Traveled
HL	Average Distance Between Heart Loss
PH	Percentage of Heart Collected
GA	Genetic Algorithm
G1	First generation of GA

8 REFRENCES

[1] Hunicke, Robin. (2005). *The case for dynamic difficulty adjustment in games.* ACM International Conference Proceeding Series. 265. 429-433. 10.1145/1178477.1178573.

[2] G. N. Yannakakis and J. Togelius, "*Experience-driven procedural content generation (Extended abstract),*" 2015 International Conference on Affective Computing and Intelligent Interaction (ACII), 2015, pp. 519-525, doi: 10.1109/ACII.2015.7344619.

[3] R. Lopes, K. Hilf, L. Jayapalan, and R. Bidarra, "*Mobile adaptive procedural content generation,*" May 2013.

[4] D. Wheat, M. Masek, C. P. Lam and P. Hingston, "*Dynamic Difficulty Adjustment in 2D Platformers through Agent-Based Procedural Level Generation,*" 2015 IEEE International Conference on Systems, Man, and Cybernetics, 2015, pp. 2778-2785, doi: 10.1109/SMC.2015.485.

[5] J. Teats, M. G. Smith and N. W. Fruin, "*Polymorph: dynamic difficulty adjustment through level generation*", In Proceedings of the 2010 Workshop on Procedural Content Generation in Games, ACM, (2010), pp. 11.

[6] C. Pedersen, J. Togelius and G. N. Yannakakis, "*Modeling player experience in Super Mario Bros,*" 2009 IEEE Symposium on Computational Intelligence and Games, 2009, pp. 132-139, doi: 10.1109/CIG.2009.5286482.

[7] T. Huber, S. Mertes, S. Rangelova, S. Flutura and E. André, "*Dynamic Difficulty Adjustment in Virtual Reality Exergames through Experience-driven Procedural Content Generation,*" 2021 IEEE Symposium Series on Computational Intelligence (SSCI), 2021, pp. 1-8, doi: 10.1109/SSCI50451.2021.9660086.

[8] U. Technologies, "More than a game engine," Unity, 2022. [Online]. Available: https://unity.com/pages/more-than-an-engine. [Accessed: 21-Nov-2022].

[9] R. W. Picard, *Affective Computing.* Cambridge, MA: The MIT Press, 1997.

[10] Christyowidiasmoro, R. C. A. Putra and S. M. Susiki, "Measuring level of difficulty in game using challenging rate (CR) on 2D Real time Strategy Line Defense game," 2015 International Electronics Symposium (IES), 2015, pp. 218-222, doi: 10.1109/ELECSYM.2015.7380844.

[11] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. Andr´e, "*Player modeling*," Artificial and Computational Intelligence in Games, vol. 6, pp. 45–59, 2013.

[12] M. Silva, V. Silva and L. Chaimowicz, "Dynamic difficulty adjustment on MOBA games", Entertainment Computing, vol. 18, pp. 103-123, 2017.

[13] G. K. Sepulveda, F. Besoain and N. A. Barriga, "Exploring Dynamic Difficulty Adjustment in Videogames," 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2019, pp. 1-6, doi: 10.1109/CHILECON47746.2019.8988068.

[14] E. Masanobu, H. D. F. B. and K. Mikami, "Dynamic Pressure Cycle Control: Dynamic Diffculty Adjustment beyond the Flow Zone," 2017 Nicograph International (NicoInt), 2017, pp. 9-14, doi: 10.1109/NICOInt.2017.12.

[15] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience, New York:Harper Perennial, 1990.

[16] M. Gonzalez-Duque, R. B. Palm and S. Risi, "Fast Game Content Adaptation Through Bayesian-based Player Modelling," 2021 IEEE Conference on Games (CoG), 2021, pp. 01-08, doi: 10.1109/CoG52621.2021.9619018.

[17] M. Weber and P. Notargiacomo, "Dynamic Difficulty Adjustment in Digital Games Using Genetic Algorithms," 2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), 2020, pp. 62-70, doi: 10.1109/SBGames51465.2020.00019.