Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games

Su Xue
Electronic Arts, Inc.
209 Redwood Shores Pkwy
Redwood City, CA, USA
sxue@ea.com

Meng Wu*
Electronic Arts, Inc.
209 Redwood Shores Pkwy
Redwood City, CA, USA
mewu@ea.com

John Kolen
Electronic Arts, Inc.
209 Redwood Shores Pkwy
Redwood City, CA, USA
jkolen@ea.com

Navid Aghdaie
Electronic Arts, Inc.
209 Redwood Shores Pkwy
Redwood City, CA, USA
naghdaie@ea.com

Kazi A. Zaman
Electronic Arts, Inc.
209 Redwood Shores Pkwy
Redwood City, CA, USA
kzaman@ea.com

ABSTRACT

Dynamic difficulty adjustment (DDA) is a technique for adaptively changing a game to make it easier or harder. A common paradigm to achieve DDA is through heuristic prediction and intervention, adjusting game difficulty once undesirable player states (e.g., boredom or frustration) are observed. Without quantitative objectives, it is impossible to optimize the strength of intervention and achieve the best effectiveness.

In this paper, we propose a DDA framework with a global optimization objective of maximizing a player's engagement throughout the entire game. Using level-based games as our example, we model a player's progression as a probabilistic graph. Dynamic difficulty reduces to optimizing transition probabilities to maximize a player's stay time in the progression graph. We have successfully developed a system that applies this technique in multiple games by Electronic Arts, Inc., and have observed up to 9% improvement in player engagement with a neutral impact on monetization.

Keywords

Dynamic difficulty adjustment, player engagement optimization, progression model

1. INTRODUCTION

Difficulty is a critical factor in computer games and is a challenging factor to set appropriately. Game developers often use pre-defined curves that manipulate the level difficulty as players advance. Although these difficulty curves are usually defined by experienced designers with strong domain knowledge, they have many problems. First, the diversity among players is large. Players have a wide variety of experiences, skills, learning rates, and playing styles, and will react differently to the same difficulty setting. Second, even for an individual player, one's difficulty preference may also change over time. For example, in a level progression game, a player who loses the first several attempts to one level might feel much less frustrated compared to losing after tens of unsuccessful trials.

In contrast to static difficulty, dynamic difficulty adjustment (DDA) addresses these concerns. Such methods exhibit diversity in the levers that adjust difficulty, but share a common theme: prediction and intervention. DDA predicts a player's future state given current difficulty, and then intervenes if that state is undesirable. The strength of this intervention, however, is both heuristic and greedy. The adjustment might be in the right direction, such as making a level easier for a frustrated player. But how easy should the game be to achieve optimal long term benefit is an open question.

In this paper, we will address these issues by defining dynamic difficulty adjustment within an optimization framework. The global objective within this framework is to maximize a player's engagement throughout the entire game. We first model a player's in-game progression as a probabilistic graph consisting of various player states. When progressing in the game, players move from one state to another. The transition probabilities between states are dependent on game difficulties at these states. From this perspective, maximizing a player's engagement is equivalent to maximizing the number of transitions in the progression graph. This objective reduces to a function of game difficulties at various states solvable by dynamic programming. While we focus on level-based games as the context of this presentation, our DDA framework is generic and can be extended to other genres.

The proposed technique has been successfully deployed by Electronic Arts, Inc (EA). We developed a DDA system within the EA Digital Platform, to which game clients request and receive dynamic difficulty advice in realtime. With A/B experiments, we have observed significant in-

*The first two authors contributed equally to this paper.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3041021.3054170

creases in core player engagement metrics while seeing neutral impact on monetization. Last, but not least, our DDA recommendations are also used by game designers to refine the game design. For example, when our service repeatedly recommends easier difficulty for a certain level, the game designer knows to decrease the pre-defined difficulty of that level to satisfy the majority of population.

To sum up, the core contributions of this paper are:

- We propose a DDA framework that maximizes a player's engagement throughout the entire game.
- We introduce a probabilistic graph that models a player's in-game progression. With the graph model, we develop an efficient dynamic programming solution to maximize player engagement.
- We describe a real-time DDA system that successfully boosted engagement of multiple mobile games.

In the remainder of this paper, we will first review related DDA research. We then introduce the graph model of player's progression and describe the objective function and our optimization solution. We will next report on the application of this DDA technique in a live game as a case study. Finally, we discuss the results of this case study and our future directions.

2. RELATED WORK

Personalized gaming is one of the major trends for digital interactive games in recent years [10]. Personalization approaches include content generation, personalized gameplay, and dynamic difficulty adjustment. The theme shared by almost all game difficulty adjustment studies is that they attempt to prevent a player from transiting to undesired states, such as boredom or frustration. There are several common challenges in difficulty adjustment. For example, how to evaluate a player's current state? How to predict a player's upcoming state? What levers are appropriate to use? How to adjust the levers to most appropriate difficulty level? In this section, we review how previous work addressed these questions from different perspectives.

Many approaches are based on the evaluation of players' skill and performance, and then adapting game difficulty to match the skill level. Zook et al. conducted a series of investigations following this idea [15, 16]. They proposed a data-driven predictive model that accounts for temporal changes in player skills. This predictive model provides a guide for just-in-time procedural content generation and achieves dynamic difficulty adjustment. Similarly, Jennings et al. automatically generate 2D platformer levels in a procedural way [9]. They developed a simulator where players play a short segment of a game for data collection. From this data, they constructed a statistical model of the level element difficulty. They also learned player skill model from the simulator data. Hagelback et al. [6] studied dynamic difficulty by measuring player experience in Real Times Strategy (RTS) games. They, too, use an experimental gaming environment to evaluate testing players' subjective enjoyment according to different dynamic difficulty schemes.

The majority of DDA systems rely upon prediction and intervention as their fundamental strategy. Hamlet is a well known AI system using Valve's *Half Life* game engine [8].

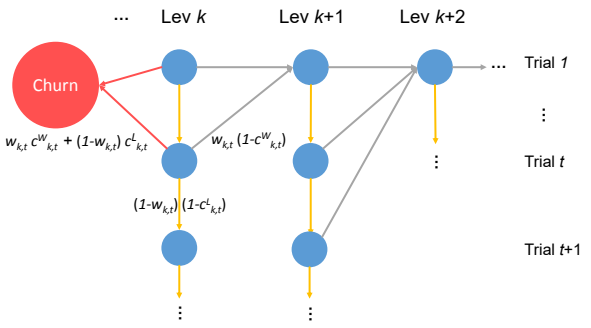

Figure 1: A player's progression model in a typical level-based game. We use a probabilistic graph consisting of player states (each circles) to model this progression. Two dimensions, levels and trials are used to identify different states. The directional edges represent possible transition between these states.

It takes advantages of the flow model developed by Csikszentmihalyi [3], that defines player states on two dimensions: skill and challenge. They suggested the game challenge should match the player skill, therefore some states are preferable while others are not. Hamlet predicts player states, and adjusts the game to prevent inventory shortfalls. Missura and Gartner [11] formalizes the prediction in a more rigorous probabilistic framework. They try to predict the "right" difficulty by formulating the task as an online learning problem on partially ordered sets. Hawkins et al. [7] takes players' risk profile into account when predicting player performance. Cautious players and risk takers also behave differently in response to dynamic balancing mechanics.

Researchers have also explored the use of various levers to achieve dynamic difficulty. It is preferred that the adjustment by the levers be invisible to players so that they do not feel coddled or abused. Popular levers in the DDA literature include procedural level content [9, 15, 16], off stage elements (such as weapon or support inventory) [8], and AI assistant or opponent [4, 13].

Almost all previous work shares a common limitation. These approaches focus on short-term effects, i.e., using the difficulty adjustment to immediately rescue player from undesired states. With the prediction and intervention strategy, these methods tend to perform greedy actions, often leading to short-term benefits, but failing to achieve long-term optimal impacts. In contrast, our proposed technique achieves DDA by maximizing a long-term objective, such as player's engagement throughout the entire game. In the following section, we describe how to model the entire game engagement, achieve global optimality, and keep players in the game.

3. PLAYER PROGRESSION MODEL

We focus on level-based games in this paper. In a level-based game, a player can unlock and advance to the higher levels only if the player wins the current level. There are many well known digital games e.g. Super Mario Bros. (Nintendo Co., Ltd.) and Plants vs. Zombies (Electronic Arts, Inc.) belong to the level-based game category. We first introduce a progression graph to model players' progression

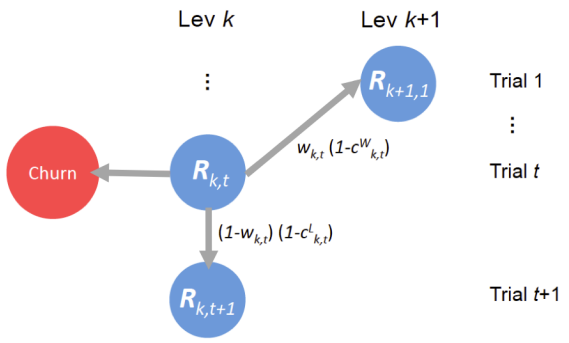

Figure 2: A zoom-in look of the state transitions and the associated rewards. The reward at $s_{k,t}$, i.e., $R_{k,t}$, is the weighted sum of the awards at the adjacent states. This property leads to reward maximization with dynamic programming.

trajectories in the game. The modeling approach described below can be generalized to other game types as well.

Defining appropriate states is the key to constructing a progression model. Specifically for level-based games, we simply define the player progression state with two dimensions, level and trial. A player can either advance to a higher level or remain on the current level with repeated trials. Fig. 1 illustrates the progression graph schema. We denote state that a player is playing the k-th level at the t-th trial as $s_{k,t}$. Within the progression graph, a player' progression can be represented by a sequence of transitions between states. For example, when a player completes one level, he will advance to a new first trial state in the next level. When a player fails and retries the same level, he will move to the next trial state on the same level. A special, but critical, state is the churn state. Players who enter the churn state will never return to the game. Hence, the churn state is an absorbing state avoided by the optimization process of DDA.

We now define the transitions between states (represented as directional edges in Fig. 1). A player can only transit to one of two adjacent live states from current live state: 1) the player wins and advances to the first trial of the next level, i.e., $s_{k,t} \rightarrow s_{k+1,1}$; 2) loses but retries the current level, i.e., $s_{k,t} \rightarrow s_{k,t+1}$. Technically, the assumption is not always true since players are able to retry the current level or play even lower levels after winning. Level replay rarely happens in most games, however. In addition to the transitions described above, all live states can directly transit to the churn state as player leave the game.

Given this structure, we need a probabilistic model of each transition that measures the likelihood of the transition happening. All outgoing transition probabilities sum to one. Since there are only three types of transitions, we can easily investigate each transition respectively.

Level-up Transition Starting at state $s_{k,t}$, players can level up to state $s_{k+1,1}$ only if they win and do not churn. Denoting the win rate (i.e., probability to win this level at this state) as $w_{k,t}$, and the churn rate after winning as $c^W_{k,t}$, we have the level-up probability as:

$$\Pr(s_{k+1,1}|s_{k,t}) = w_{k,t}(1 - c^W_{k,t}) \quad (1)$$

Retry Transition From $s_{k,t}$, players transits to retrial state $s_{k,t+1}$ only if they lose and do not churn. The probability of loss is $1 - w_{k,t}$. Denoting the churn rate after losing as $c^L_{k,t}$, we have the retry probability as:

$$\Pr(s_{k,t+1}|s_{k,t}) = (1 - w_{k,t})(1 - c^L_{k,t}) \quad (2)$$

Churn Transition Unless players make the above two transitions, they will churn. The total churn probability at $s_{k,t}$ is the sum of the churn probabilities after winning and after losing, i.e.,

$$\Pr(churn|s_{k,t}) = w_{k,t}c^W_{k,t} + (1 - w_{k,t})c^L_{k,t} \quad (3)$$

We illustrate the transition probabilities for a given state model in Fig. 1. This probabilistic graph model is the foundation of our optimization framework for dynamic difficulty adjustment in the next section. Note that we assume $c^W_{k,t}$ and $c^L_{k,t}$ are independent of $w_{k,t}$.

4. ENGAGEMENT OPTIMIZATION

4.1 Objective Function

With the player progression model, good game design and difficulty adjustment should seek to prevent players from falling into the churn state. DDA achieves higher engagement by adjusting win rates so that the player stays on a state trajectory with lower churn rates. While existing DDA techniques adapt difficulties at each state in a greedy and heuristic manner, our framework identifies optimal win rates for all states, targeting a global objective: maximizing a player's total engagement throughout the entire game.

Engagement indicates the amount of players' gameplay. There are multiple engagement metrics, e.g., the number of rounds played, gameplay duration and session days. In this paper, we chose to optimize the total number of rounds played. Three reasons drive this selection. First, the number of rounds a player plays is easily measured in the progression graph. It is the transition count before reaching the churn state or completing the game. Second, many engagement metrics turn out to strongly correlate with each other. We will discuss this observation in Section 5.4. Third, maximizing the number of rounds prevents degenerate solutions that rush a player to the completion state by making the game too easy. Any solution with round repetition will score higher than the shortest path through the graph.

We use R to denote the reward function, i.e., the expected total number of rounds a player will play through the entire game. While R hardly looks tractable, we convert it to a more solvable iterative form with the help of the Markov property of the progression graph model. We define reward $R_{k,t}$ as the expected total number of rounds played after the state $s_{k,t}$ (level k with trial t). Although we only consider the expectation of the reward in this paper, one could also optimize the variance.

As the player can only transit to two adjacent live states, $s_{k+1,t}$ and $s_{k,t+1}$, or churn, $R_{k,t}$ can be computed as the weighted sum of $R_{k+1,t}$ and $R_{k,t+1}$. The weights are the transition probabilities between the states. Mathematically, it is written as

$$\begin{aligned} R_{k,t} = \ & \Pr(s_{k+1,t}|s_{k,t}) \cdot R_{k+1,t} \\ & + \Pr(s_{k,t+1}|s_{k,t}) \cdot R_{k,t+1} + 1, \end{aligned} \quad (4)$$

where $\Pr(s_{k+1,t}|s_{k,t})$ is the probability that the player wins and levels up, and $\Pr(s_{k,t+1}|s_{k,t})$ is the probability that one failed and retries. Adding one represents the reward by completing that round. Transition to the churn state does not contribute to the engagement.

Furthermore, substituting the transition probabilities from Eqns. 1 and 3 into Eqn. 4 (see Fig. 2), produces

$$\begin{aligned} R_{k,t} =& w_{k,t}(1-c^W_{k,t})\cdot R_{k+1,t} \\ &+ (1-w_{k,t})(1-c^L_{k,t})\cdot R_{k,t+1}+1. \end{aligned} \tag{5}$$

Note that the original optimization objective, R, corresponding to $R_{1,1}$. Based on Eqn. 5, $R_{1,1}$ is a function of win rates at all states, $\{w_{k,t}\}$, where $\{c^W_{k,t}\}$ and $\{c^L_{k,t}\}$ are parameters that can be extracted from performance data (see details in Section 4.3). Dynamic difficulty adjustment reduces to solving optimal $\{w_{k,t}\}$ for maximizing $R_{1,1}$.

4.2 Solving for Optimal Difficulties

We need to solve an optimization problem that finds a set of optimal difficulties over all states, thus

$$\mathcal{W}^* = \arg\max_{\mathcal{W}} R_{1,1}(\mathcal{W}), \tag{6}$$

where $\mathcal{W} = \{w_{k,t}\}$. In practice, each $w_{k,t}$ is constrained by game design and content. We solve for optimal $\mathcal{W}$ under the constraint that $w_{k,t} \in [w^{\text{low}}_{k,t}, w^{\text{up}}_{k,t}]$.

With Eqn. 5, we can solve this optimization effectively with dynamic programming. Denoting $R^*_{k,t}$ as the maximum reward over all possible difficulty settings, we have:

$$\begin{aligned} R^*_{k,t} =& \max \;\; w_{k,t}(1-c^W_{k,t})\cdot R^*_{k+1,t} \\ &+ (1-w_{k,t})(1-c^L_{k,t})\cdot R^*_{k,t+1}+1, \\ &\text{s.t.} \quad w_{k,t} \in [w^{\text{low}}_{k,t}, w^{\text{up}}_{k,t}] \end{aligned} \tag{7}$$

We can see that $R^*_{k,t}$ is a linear function of $w_{k,t}$ under a constraint. Therefore, the optimal win rate for state $s_{k,t}$, $w^*_{k,t}$, can be found by:

$$\begin{aligned} w^*_{k,t} =& \arg\max_{w_{k,t}} \;\; w_{k,t}(1-c^W_{k,t})\cdot R^*_{k+1,t} \\ &+ (1-w_{k,t})(1-c^L_{k,t})\cdot R^*_{k,t+1}+1 \\ =& \arg\max_{w_{k,t}} \;\; w_{k,t}((1-c^W_{k,t})\cdot R^*_{k+1,t} - (1-c^L_{k,t})\cdot R^*_{k,t+1}). \end{aligned} \tag{8}$$

Eqn. 8 shows that given the maximal rewards of two future states, $R^*_{k+1,t}$ and $R^*_{k,t+1}$, the optimal difficult $w^*_{k,t}$ can be computed easily. As the player progression model is a directed acyclic graph, we can solve the optimization with dynamic programming. We start with a few destination states whose rewards are pre-defined and then compute the rewards of the previous states backward. The primary destination state is the end of the game, $s_{K+1,1}$, where $K = k_{max}$ is the highest level of the game. We assign zero to $R^*_{K+1,1}$ as the reward for completion of the game. Another set of destination states are those when the number of retrials exceeds a limit, i.e., $s_{k,T}$ where $T \geq t_{max}$. By having the upper bound of the retrial time, we can keep the progression graph to a reasonable size. This also prevents a player from too many retrials on a level when the optimization produces unrealistic results. In our experiment we set $t_{max} = 30$ and $R^*_{k,T} = 1$.

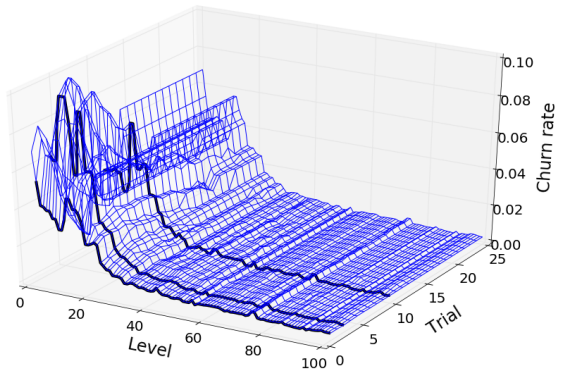

Figure 3: An example of a churn surface, which consists of $c^L_{k,t}$ at all states $s_{k,t}$. The churn rates at 1st, 3rd, 10th trials for each level are highlighted in black to illustrate how the churn rate evolves as a player's re-trials increases.

4.3 Churn Rates

We assume the churn rates in state transitions ($c^W_{k,t}$ and $c^L_{k,t}$) as known parameters. In practice, churn is identified as no gameplay during a period of time. We use a 7-day time frame that is common in the game industry and collect historical gameplay data of players at all states (levels and trials) to measure the churn rates. At state $s_{k,t}$, let $c^W_{k,t}$ be the ratio of players who churn after winning, and $c^L_{k,t}$ after losing. We view the churn rate over all states a two dimensions churn surface. Fig. 3 shows a visual example of a churn surface of $c^L_{k,t}$.

These estimates of churn rates take only two input features, level and trial, while ignoring other players' individual features such as age, play frequency, play style, skill, performance, etc. To further improve the accuracy of churn rates, we could take advantage of long-standing churn prediction research [2, 5, 12], by employing sophisticated predictive models on individual player features to improve performance. A major challenge of using runtime churn prediction is that it increases the complexity of dynamic programming optimization. Pre-computation with various possible churn rates at each state (via bucketing) would be needed. This mechanism is not employed by our current system, but will be worth exploring in the future.

5. CASE STUDY: DDA IN A LIVE GAME

We now present the case study with one of our DDA implementations, a mobile match-three game developed and published by EA. Classic example of match-three genre, e.g. Candy Crush Saga by King and Bejeweled by EA, has a game board contains multiple items in different colors and shapes. A player can swap two adjacent items in each move as long as three or more items of the same color become aligned together vertically or horizontally. The aligned items will be eliminated from the board. At each level, a player needs to achieve a specific goal, for example, score a number of points in a limited number of moves. The game features more than two hundred levels. A player starts from the lowest level and advances to the higher levels. Only if a player wins the current level, the next higher level will be unlocked.

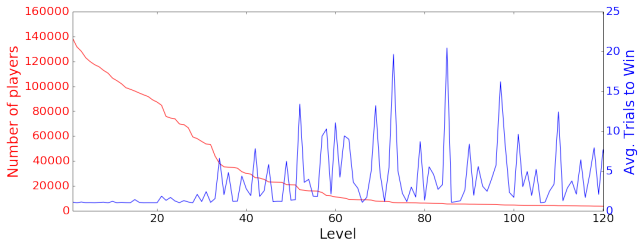

Figure 4: The retained population of players (red line) versus difficulties (blue line) by level. The red line represents, for each level, the number of players who have ever achieved it. The blue line represents the level difficulty, which is measured by 1/win rate, i.e., the average trials needed to win this level. We can observe the strong impact of difficulty on population retention, in particular for middle stage levels.

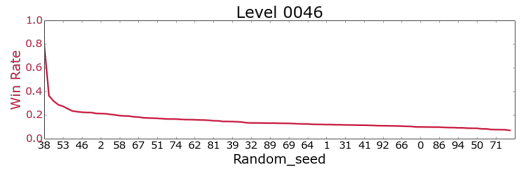

Figure 5: Difficulties of various random seeds at a level of the match-three game. Difficulty is measured by the win rate of a certain seed, i.e., the percentage out of all trials with this seed are actually wins. The variance of difficulties across seeds is large. We can see that the easiest seed (leftmost, seed 38) shows a win rate up to 0.75. In contrast, the hardest seed (rightmost, seed 71) has a win rate as low as 0.15.

Before adopting DDA, we must ask: can DDA help this game? To answer this question, we must convince ourselves of a causal link between the difficulty and engagement. First, we should determine if game difficulty is affecting the player engagement in the game. Second, appropriate levers should exist to effectively adjust the game difficulty. We will examine these two prerequisites in the following sections.

5.1 Difficulty and Engagement

To evaluate the impact of difficulty on player engagement, we compared the retained population with level difficulties (see Fig. 4). Retained population (red line) at a level is the number of players who have achieved this level as the highest one. There are players churned at each level, thus the retained population decreases as the level increases. The difficulty (blue line) is measured by the average number of trials that are needed to win this level. The more trials it takes, the more difficult this level is. Dividing all levels into three ranges: lower range ($\leq$20), middle range (21-80) and high range (>80), we can see that the correlation between retained population and difficulty varies. In the lower range of levels, the population naturally decays regardless of the low difficulty of these levels. Especially at level 21, there is a slight increase in difficulty, and the number of players drops significantly. In the high range of levels, the population becomes flat and decays slowly, despite that these high levels are very difficult. In contrast, in the middle range, difficulty spikes are highly correlated with the steep drops in retained population. This observation supports the hypothesis that appropriate difficulty adjustment has the potential to enhance player engagement for this game.

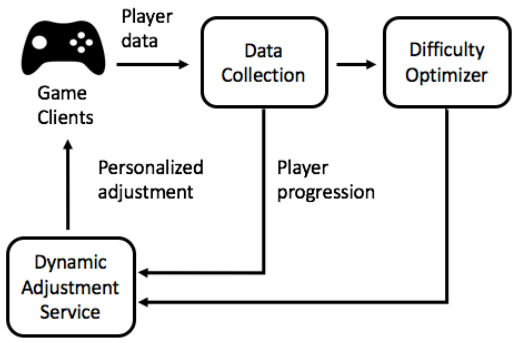

Figure 6: Schematic diagram of the Dynamic Difficulty Adjustment system.

5.2 Difficulty Lever

DDA adjusts win rates at different states in the progression graph through a difficulty lever. An effective difficulty lever needs to satisfy two key constraints. First, adjusting this lever should make the game easier or harder. Second, adjusting this lever should be invisible to players (as reviewed in [10]). For example, although we can simply change the "goal" or "mission" to lower game difficulty, players can easily observe it in retrials. As a consequence, such changes undermine the players' sense of accomplishment even when they finally win with the help of DDA.

Fortunately, the case study game provides an effective difficulty lever: the random seed of board initialization. At the beginning of each round, the board is initialized from a random seed, which is indexed by a integer from 0 to 99. After evaluating the average win rate of each seed in gameplay data, we find a wide range of difficulties. Fig. 5 shows an example of difficulties versus seeds at one level. The seeds are ordered by their observed win ratios. We can see that the easiest seed (leftmost) has a win rate as high as 0.75, while the hardest seed (rightmost) has a win rate as low as 0.15. The player who plays the hardest seeds will take 5x more trials to pass this level than those playing the easiest seeds. This variance can be explained by the game mechanism. For example, some initial boards have many items of the same color close to each other, making it significantly easier to match items than boards with more pathological scattering. By carefully selecting the seed according the mapping in Fig. 5, we can control the game difficulty for players on this level. The hardest and easiest seeds provide the lower and upper bounds of win rates, i.e., w^{low} and w^{up} in Eqn. 6.

5.3 Dynamic Adjustment System

We developed a real-time system to serve dynamic difficulty adjustments in an EA match-three game. Fig. 6 describes the workflow of the DDA system. At the beginning of each game round, the game clients send a request to the DDA service. The dynamic adjustment service determines the most suitable difficulty for the current player state, $s_{k,t}$

based on the player progression and difficulty optimization results, $w^{\star}_{k,t}$. The optimal win rates are computed offline as discussed in Section 4.2. Since we use random seeds as the difficulty lever, the dynamic adjustment service then applies the mapping from win rates to the random seeds showed in Fig. 5 and return it to the game client. In practice, we randomly select one seed from the top five candidate seeds to prevent from repeatedly playing only one seed. The game was first released in a period without DDA (soft launch phase), allowing the measurement of win rate for each random seed. After DDA is started, we continued collecting player data to improve the random seed mapping, churn probabilities, and difficulty optimization.

5.4 Experimental Results

To measure the effectiveness of our technique, we conducted A/B experiments that use multiple variants as control and treatment groups. The control group randomly recommends seeds out of all possibilities, an action corresponding to the default game behavior. The treatment group recommends the seeds based on our DDA optimization. We calculated all parameters for optimization, such as the churn surface and win rates of seeds, from real game play data. We kept track of these parameters and updated them when significant changes were observed.

The experiment started two weeks after the global release of the game. We conducted the experiment in three phases, where the proportion of the treatment group increased gradually from 10% to 40%, and finally 70% (the proportion of the control group decreases from 90%, 60% to 30%). The first two phases each lasted about one month. The third phase has been live for about four months and is ongoing. We compared core engagement metrics between the control and the treatment groups to evaluate the effectiveness of our DDA scheme. The results are daily averages normalized according to the proportion of its group, so that groups with different proportions could be compared. Phase III has not been terminated in order to collect churn probabilities and evaluate performance.

Table 1 shows the increase of the number of rounds played, suggesting that the DDA is optimizing its objective metric. In each phase of increasing treatment populations, all treatment groups exhibits statistically significant improvement (p-value < 0.05). Table 2 shows the impact of DDA on another engagement metric, total gameplay duration. Though this metric is not the explicit optimization objective of DDA, we wanted to test an alternative hypothesis that players played more rounds in the same amount of time. Our DDA treatment group shows significant improvement on this engagement metric as well. Similarly, performance increased as more data was collected in the second phase; then stayed stationary when the gameplay data became accurate and stable in the latest phase. Note that we see slightly different performances between iOS and Android platforms, though, in the same order. This resonates with the common observation in mobile game development that user behaviors between platforms often differ from each other [1, 14], so that separate modeling and treatment are preferred.

We observed the lowest performance in Phase I, when DDA is completely based on the model we learned from limited data - soft launch data and first two weeks in worldwide release. The soft launch data is only partially useful as some game design is changed before worldwide release. After the release, we continued to collect more data to form more accurate parameters for DDA optimization. This explains the further improved performance in Phase II over Phase I. In Phase III, the performance has been observed stable for more than three months. The amount of boost is relatively smaller than that of Phase II because there are fewer new players in this phase and DDA has higher impacts on early stage players (see Fig. 4).

Phase	Platform	Default	DDA	Delta	Gain
I	iOS	1,118,237	1,167,616	+49,379	+4.4%
	Android	789,640	825,182	+35,543	+4.5%
II	iOS	855,267	915,334	+60,067	+7.0%
	Android	1,137,479	1,228,473	+90,995	+7.9%
III	iOS	711,749	763,508	+51,759	+7.2%
	Android	1,285,448	1,366,820	+81,373	+6.3%

Table 1: Total numbers of rounds played daily in the default (control) group versus in the DDA treated group. Delta is the absolute increase by DDA and Gain is the relative increase.

Phase	Platform	Default	DDA	Delta	Gain
I	iOS	3,684,082	3,847,516	+163,435	+4.4%
	Android	2,686,781	2,814,953	+128,172	+4.8%
II	iOS	2,916,570	3,148,722	+232,152	+7.9%
	Android	3,787,414	4,129,762	+342,348	+9.0%
III	iOS	2,582,809	2,788,690	+205,881	+8.0%
	Android	4,619,907	4,956,680	+336,773	+7.3%

Table 2: Total durations of gameplay time (in minutes) daily in the control group versus in the DDA treated group.

Last but not least, we also compared the impact on monetization between the control and the treatment groups. This comparison is critical as a monetization objective might contradict engagement maximization. Our DDA treatment group had a neutral impact on monetization. No statistically significant difference on in-game transaction revenues was observed between two groups. This is probably caused by the mechanism of our optimization framework: it "saves" players of high churn risks, who are less likely to spend.

6. CONCLUSION

In this paper, we presented a framework that approaches dynamic difficulty adjustment (DDA) as an optimization problem. While existing DDA systems adapt game difficulty in a greedy manner for local benefit, our method maximizes the player engagement throughout the entire game. We modeled the player progression as a probabilistic graph, with which engagement maximization becomes a well-defined objective and we proposed an efficient dynamic programming method to solve it. We evaluated an implementation of the DDA technique in a mobile game by EA. The DDA treated group shows significant increases in core engagement metrics, such as a total number of rounds played and gameplay duration, while it is revenue neutral compared to the control group with no DDA enabled.

In the near future, we will extend the framework to a wide range of game genres. The key to successfully applying DDA to other genres is the construction of an adequate progression model. For level-based games, the states can be naturally identified by two major dimensions: level and trial. For more complicated games with non-linear or multi-

ple progressions, such as role-playing games (RPG), we can also define the states with different dimensions. The graph might be more complicated as it includes more states and connections. Solving the optimal difficulty associated with each state, which maximizes player engagement, however, remains a well-defined graph optimization problem.

Another problem worth investigating is the cold-start stage when the graph parameters (such as seed difficulty and churn risks) needed for optimization are not known yet. In the current system, we learned these parameters in soft launch and the first few weeks after release before starting treatment (see Section 5.4). A possible direction is to conduct reinforcement learning at this early state, by defining some short-term awards that help quickly establish greedy policies. Once sufficient data is collected, we can then shift to the supervised optimization framework.

Last but not least, we would like to explore generic dynamic player experience. Beyond difficulty adjustment, how can we provide a dynamic and intelligent game experience personalized for individual players? It includes, but not limited to, dynamic tutorials, dynamic messages, dynamic awards, dynamic advertisement and etc. We believe that the progression model and corresponding optimization framework will become key to these generic solutions.

7. REFERENCES

[1] Z. Benenson, F. Gassmann, and L. Reinfelder. Android and ios users' differences concerning security and privacy. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 817–822, New York, NY, USA, 2013. ACM.

[2] R. Bernhaupt. User experience evaluation in entertainment. In *Evaluating User Experience in Games*, pages 3–7. Springer, 2010.

[3] M. Csikszentmihalyi and I. S. Csikszentmihalyi. *Optimal experience: Psychological studies of flow in consciousness*. Cambridge university press, 1992.

[4] P. Demasi and J. d. O. Adriano. On-line coevolution for action games. *International Journal of Intelligent Games & Simulation*, 2(2), 2003.

[5] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. Predicting player churn in the wild. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[6] J. Hagelback and S. J. Johansson. Measuring player experience on runtime dynamic difficulty scaling in an rts game. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 46–52. IEEE, 2009.

[7] G. Hawkins, K. Nesbitt, and S. Brown. Dynamic difficulty balancing for cautious players and risk takers. *Int. J. Comput. Games Technol.*, 2012:3:3–3:3, Jan. 2012.

[8] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.

[9] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 11. ACM, 2010.

[10] S. Karpinskyj, F. Zambetta, and L. Cavedon. Video game personalisation techniques: A comprehensive survey. *Entertainment Computing*, 5(4):211–218, 2014.

[11] O. Missura and T. Gärtner. Predicting dynamic difficulty. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2007–2015. Curran Associates, Inc., 2011.

[12] J. Runge, P. Gao, F. Garcin, and B. Faltings. Churn prediction for high-value players in casual social games. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[13] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma. Difficulty scaling of game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37, 2004.

[14] C. Zhou, Y. Guo, Y. Chen, X. Nie, and W. Zhu. Characterizing user watching behavior and video quality in mobile devices. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6, Aug 2014.

[15] A. Zook, S. Lee-Urban, M. R. Drinkwater, and M. O. Riedl. Skill-based mission generation: A data-driven temporal player modeling approach. In *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, PCG'12, pages 6:1–6:8, New York, NY, USA, 2012. ACM.

[16] A. Zook and M. O. Riedl. A temporal data-driven player model for dynamic difficulty adjustment. In *Artificial Intelligence and Interactive Digital Entertainment*, 2012.