Inferring and Comparing Game Difficulty Curves using Player-vs-Level Match Data

Anurag Sarkar
Northeastern University
Boston, USA
sarkar.an@husky.neu.edu

Seth Cooper
Northeastern University
Boston, USA
se.cooper@northeastern.edu

***Abstract*—Prior work has focused on formalizing difficulty curves by using function composition to give precise definitions to curves and their transformations. However, the proposed framework was demonstrated using a single game, and the curves and transformations were defined with respect to the game's ratings-based dynamic difficulty system. In this work, we infer difficulty curves from gameplay data using a method that is based on the aforementioned difficulty system but that can also be generalized to other games for which information on player-vs-level win/loss outcomes is available. Moreover, since this method uses the same difficulty mechanism as past work, it lets us similarly leverage function composition to compare difficulty curves across games, having either a fixed or dynamic level ordering, using a clearly defined vocabulary. We use four different games to demonstrate our method, which relies on an adjustment to traditional playback of ratings-based match data, which we also present in this work.**

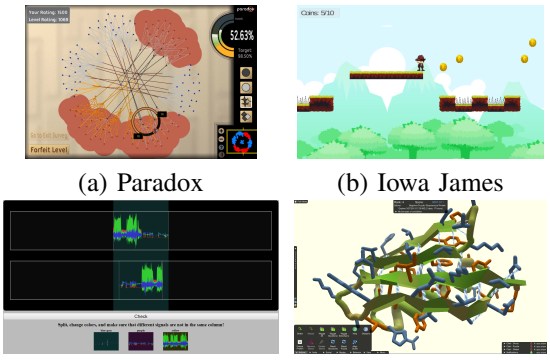
***Index Terms*—difficulty curve; rating systems; gameplay data**

I. Introduction

Traditional methods of defining optimal difficulty curves for games typically rely on an iterative process of manual refinement via several rounds of playtesting [1]. Such methods lack precise ways of capturing what it means to modify curves to affect difficulty. Past work [2] addressed this by using function composition to model curves and their transformations in order to move towards a formal way of examining difficulty curves and offer a precise vocabulary of discussing changes to curves in order to trade-off different design goals. However, the curves used were defined in terms of the ratings-based dynamic difficulty system of the specific game, *Paradox*.

In this paper, we extend this prior work to infer game difficulty curves from player-vs-level (PvL) match data in a game-independent manner. Specifically, we use the Glicko-2 rating system [3] to playback PvL match data from four different games and infer their difficulty curves using the formulation employed in the prior work. As in that work, this lets us use function composition to discuss and compare curves across games using a precise vocabulary by fitting parameterized versions of these curves to gameplay data. Further, use of rating systems via match playback lets us apply the approach to games with both static and dynamic difficulty.

We also applied an approach to address a survivorship issue during playback that caused harder levels of the games to appear easier, as only more advanced players made it to those levels. Our approach addresses this by generating *phantom matches* between players and the levels they don't end up playing. This adjusted playback lets us use rating systems to elicit difficulty curves using data about player-vs-level win/loss outcomes, for any game for which such data is available, regardless of whether that game uses a ratings-based difficulty system, as well as independent of whether the game uses a fixed or dynamic difficulty progression. This work contributes a generic method to infer difficulty curves of games from gameplay data, and a technique for comparing those curves.



(a) Paradox (b) Iowa James

(c) Signaligner (d) Foldit

Fig. 1. Screenshots from the four games used in this work.

II. Background

A. Difficulty Balancing using Rating Systems

Use of rating systems like Elo [4] and Glicko-2 [3] for balancing difficulty was motivated by the specific constraints to difficulty adjustment in human computation games (HCGs) where levels model real problems and are not readily modifiable [5]. In such usage, players and levels are assigned ratings indicating skill and difficulty respectively. Ratings are updated during gameplay based on outcomes of player attempts at levels. This enables dynamic difficulty adjustment (DDA) by allowing the use of matchmaking to match players with levels compatible with their skill. Prior work has demonstrated the effectiveness of ratings-based systems for DDA [6] and difficulty curve refinement [2] in HCGs. Here, since ratings are used to determine the difficulty progression, a game's difficulty curve can be defined in terms of the rating system rather than

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

in terms specific to the game. Thus, ratings-based curves could be used to compare difficulty across games using a common vocabulary if difficulty is framed in terms of the rating system.

B. Difficulty Curves and Function Composition

Recent work [2] looked at formalizing the definition, refinement and transformation of difficulty curves using function composition. That approach applied basic translation and scaling functions to the game's existing difficulty curve to generate new curves, offering a precise way of defining and transforming curves and discussing comparisons and relationships between different curves and their effects on gameplay. However, that work only looked at *Paradox* with the methodology not demonstrated as generalizable to other games where the difficulty progression and curve are not defined in terms of a rating system. The method presented in this current work aims to fit rating system-based difficulty curves on existing gameplay data for several games in order to compare and discuss curves across games using this function composition-based formulation.

III. Approach

In our approach, first, gameplay data is collected for a game. Next, the data is used to create data points describing a mapping from player skill to difficulty for that game. Finally, a parameterized curve is fit to the data points. These steps are described below.

A. Collecting Gameplay Data

Gameplay data for our method consisted of match data for each game where each instance of a player playing a level was treated as a match. Each match in the data is described by an entry consisting of the timestamp of the match, the player, the level and the outcome, i.e. if the player won or lost the level.

B. Sampling from Player Skill to Difficulty

Difficulty curves can be viewed as functions that map from player skill to difficulty. Thus, to fit curves onto match data for a game, we sample the mapping from player skill to game difficulty. As in prior work, we use Glicko-2 ratings to represent player skill and the player's loss rate (i.e. how often a player will lose a match) to represent difficulty. Sampling this mapping allows us to use a ratings-based difficulty framework for the purpose of leveraging function composition for inferring and comparing curves.

1) Playback: The gathered match data for a game is played back using the pyglicko2 Python implementation [7] of the Glicko-2 system with each player and level starting with a default rating of 1500. Playback generates ratings for players and levels based on PvL outcomes which can be used to determine a player's probability of losing a level. This in turn can be viewed as the difficulty of that level for that player. A level's rating is thus an estimate of the player skill required to win against that level. In other words, the lower the player's rating is compared to that of the level, the harder that level should be for them. Thus, playing back match data for a game should ideally allow inferring the game's difficulty curve through the lens of player and level ratings. During playback, each match in the gameplay data is used to create a sample of the game's difficulty curve by recording the current estimate of the player's skill before the match and whether the player won or lost that match. All of these samples (including those from the phantom matches described below) are then grouped in to bins by rating and the average player loss rate for each bin is computed. We used bin sizes of 50 Glicko-2 rating units.

2) Phantom Matches: In both fixed and dynamic difficulty progressions however, only the more skilled players usually reach or get matched up with the harder levels. Thus, in match data, this gets reflected as the harder levels being involved with matches only with these skilled players who are more likely than the average player to win against these levels and cause their ratings to go down. Hence, regular playback of match data for games with a fixed or dynamic difficulty progression may suffer from *survivorship* issues—only the skilled players that survive past the easy and moderately difficult levels end up playing the hardest levels in the game thus causing the latter to end up with ratings that are not indicative of their true difficulty. Since this survivorship issue is caused by harder levels being played only by highly skilled players, which skews the ratings for the hard levels, to fix this problem, we create a *phantom match* for each PvL pairing that did not take place in the actual gameplay, i.e. a PvL instance for which there was no entry in the match data. For each such phantom match between player p and level l, we determined its result as follows:

- During playback, note lowest rated player x that beat l
- If p's final rating $\geq x$'s rating, p wins, else p loses

The phantom matches thus allow harder levels to 'get back' wins against lower skilled players who dropped out before reaching these levels, or never got good enough to match up against them. Hence, the combined match data for each game consisted of the real matches plus these phantom matches.

C. Fitting Curves to Sampled Data

Once we have the binned samples from player rating to loss rate, we can fit a curve to these data points. In this work we use a logistic function, previously used for DDA in *Paradox* [6], that maps from player rating to the probability of the player losing at that rating (i.e. the loss rate). This rate is a measure of difficulty as it determines how hard the next match will be for that player. The logistic function represents a smoothly changing difficulty curve. The baseline difficulty curve was taken from prior work in *Paradox* [2] and is given by:

$$f(x) = \frac{1}{1 + e^{\alpha(\beta - x)}} \tag{1}$$

where x is the player's rating and α and β are constants set to $\sim$0.006 and 1850 respectively so that players with a starting rating of $x = 1500$ have a low 10% chance of losing.

Additionally, for function composition, we used two transformation functions: $t_\delta(x) = x + \delta$, which *deflates* or *inflates* the curve by translating the player rating by δ; and $s_{\sigma,c}(x) = \sigma(x - c) + c$, which *smooths* or *steepens* the curve by scaling the player rating by σ around c (we used a constant 1500 for c). Composing one or both of these with $f(x)$ generates different curves. Details of such transformations are given in

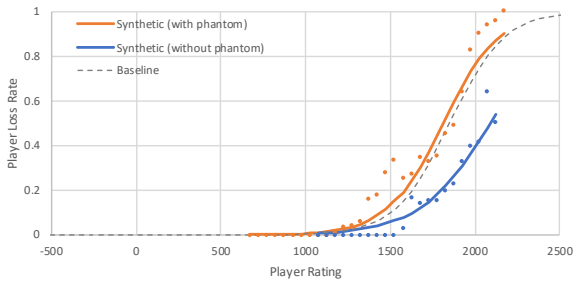


Fig. 2. Synthetic data with and without phantom matches. Lines are the fit curves and points are binned samples.

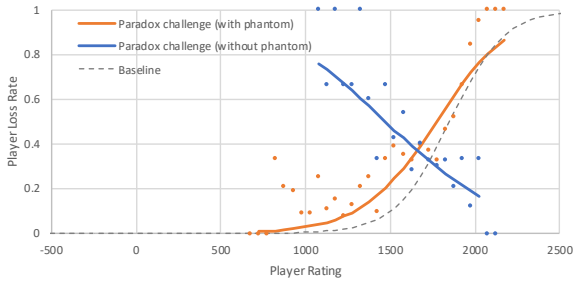


Fig. 3. Data from Paradox challenge levels with and without phantom matches. Lines are fit curves and points are binned samples.

[2]. By composing the functions as $f \circ t_\delta \circ s_\sigma$, we get difficulty curves parameterized by δ and σ. We then fit difficulty curves to the binned sample data by optimizing δ and σ to minimize the root mean squared error between the curve value at the center of each bin and the mean player loss rate in the bin.

IV. Validating Phantom Matches

For validating our phantom match methodology, we used a synthetically generated dataset and data from the challenge (i.e. non-tutorial) segment of *Paradox*. In both cases, we know that the underlying baseline difficulty curve ($\delta = 0$, $\sigma = 1$) was used to generate matches, and thus we should recover parameter values close to those when fitting the curve.

A. Synthetic Data

To create the synthetic dataset, we generated 50 players with uniformly random Glicko-2 ratings between 900 and 2100 and 61 levels with ratings ranging from 0 to 3000 in increments of 50. Note that these were their *true* ratings indicating their skill and difficulty, but for matchmaking, each player and level was still given an *estimated* starting rating of 1500. To simulate matches, we randomly selected a player, used the estimated ratings to determine which level to serve, but used the true ratings to simulate whether the player won or lost against that level. To simulate players stopping playing, we set a drop rate proportional to their true rating, so that lower-skilled players were more likely to stop playing. If a player lost a match, they stopped playing based on their drop rate. This models how in actual gameplay, players who are performing poorly may be more likely to stop playing than those who are doing well. We continued simulating matches in this manner until there were no remaining players. This process was run five times and all the matches combined. The resulting binned samples and fit curves are shown in Figure 2 and curve data are given in Table I. Notably, using the phantom matches produces a curve fit very close to the baseline, while without the phantom matches, the game appears easier for higher-rated players.

B. Paradox

For validation, we also used gameplay data from the challenge portion of *Paradox* gathered from prior work. Here, we would again expect the curve that fits to this data to be similar to the baseline since the latter was used for DDA-based matchmaking in the game's challenge section. The resulting binned samples and fit curves are shown in Figure 3 and curve data are given in Table I. Without the phantom matches, the curve appears *inverted*; using the phantom matches however produces a curve fit much closer to the underlying baseline.

V. Application

A. Games

To apply our approach, we used gameplay data from four games: three puzzle human computation games (HCGs)—*Paradox*, *Signaligner* and *Foldit*—each with varying mechanics and gameplay, and one platformer—*Iowa James*. Screenshots of each game are shown in Figure 1. We gathered match data from existing gameplay data collected previously through Amazon Mechanical Turk for *Paradox*, *Signaligner* and *Iowa James* and through the version of *Foldit* available on its website (https://fold.it). Each game is described below.

Paradox is a 2D puzzle HCG where levels represent boolean maximum satisfiability problems. Players try to complete levels by satisfying a target number of constraints via assigning boolean values to variables using various tools. For gameplay match data, a player completing a level is a win for the player. If a player fails to reach the target score for a level before moving on to the next one, it is a loss for the player. The tutorial levels follow a fixed order while the challenge levels follow a dynamic difficulty ordering based on Glicko-2 ratings. Gameplay details can be found in prior research [6] that used the game. In this section of analysis, we used the full game consisting of 8 tutorial and 50 challenge levels.

Iowa James [8] is a platformer consisting of 14 levels following a fixed, increasing difficulty progression. Each level has a variety of hazards that the player must avoid in order to reach a treasure chest at the end, which lets the player move on to the next level. Players have unlimited lives during playthrough. For each level, if the player reaches the chest, regardless of the number of times they died, it is considered a win for the player. If the player quits without reaching the chest, it is taken as a loss for the player on that level.

Signaligner is a 2D puzzle HCG where players annotate raw accelerometer data with activity labels by splitting, merging, and aligning accelerometer data signal blocks in order to group together similar looking ones. Players can submit their answer when they think they have the blocks organized correctly. The game is relatively short with players given four tutorial levels where they can submit as many answers as they want until submitting a correct one, at which point they play one of seven

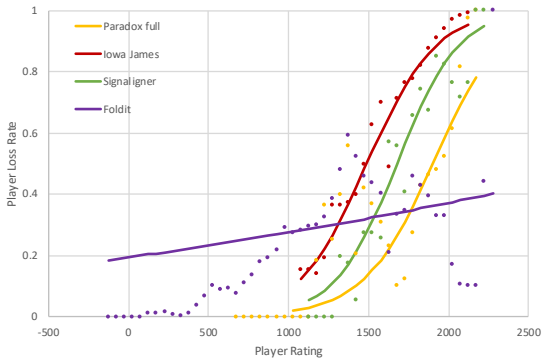


Fig. 4. Fit difficulty curves for each game. Points represent binned samples.

possible challenge levels with only one chance to answer. If they submit a correct answer, they win, else they lose.

Foldit [9] is an HCG based on protein folding and design, where players attempt to interactively fold and pack protein structures as efficiently as possible. The game contains a set of 37 tutorial levels meant to introduce the game, which we used in this analysis. The player's score in a level is based on the energy of their current fold, and players win a tutorial level by reaching a target score. The game's tutorial level progression is the same for all players, but players have choices at some branching points and can replay previous levels, via a level select screen.

B. Curve Comparisons and Transformations

Plots of the fit curves for each game, along with the binned samples, are shown in Figure 4. Curve data are given in Table I. Using the terminology from [2], *Foldit* has the *smoothest* difficulty with the other three games exhibiting *steeper* difficulty curves. Among these three, *Iowa James* has an *inflated* (i.e. higher) difficulty compared to *Signaligner* and *Paradox* and consequently, *Paradox* has a *deflated* (i.e. lower) difficulty compared to the other two.

Further, the curves for *Paradox* and *Foldit* have the highest RMSE as the binned samples go up and down suggesting that a single curve does not fit their data well and multiple curves might better represent their difficulty. Overall, the ability to compare difficulty curves of multiple games and discuss how they relate to each other in this precise and generic way demonstrates the utility of our approach.

VI. Conclusion and Future Work

We presented a method of inferring a game's difficulty curve using its gameplay data. Since the curve is defined as a function mapping from player skill to difficulty and is independent of the game's level ordering, the method is applicable whether the game has a fixed or dynamic difficulty progression. We consider several avenues for future work.

We fit a single curve to the data for each game, but a difficulty curve might more closely follow a sawtooth pattern, common in such curves [10], that is better captured using multiple piecewise curves. This would enable inferring separate curves for segments of a game that differ in difficulty. Additionally, the drop rate for generating synthetic data was set heuristically. Future work could look into empirically confirming this rate. We also used binary win/loss for PvL outcomes as we did not want to use a heuristically set scoring mechanism. Future work could thus explore if continuous outcomes can help in inferring more accurate difficulty curves. Lastly, future work can also explore further impact and any biases from using phantom matches.

TABLE I
Match counts and parameter values for each fit curve shown in Figures 2-4. *Matches* and *Phantom* refer to number of actual and phantom matches respectively.

Game	Matches	Phantom	RMSE	σ (scale)	δ (transl.)
Synthetic (w/ phantom)	484	1732	0.07	0.96	50
Synthetic (w/o phantom)	484	0	0.04	0.76	-98
Paradox Chal. (w/ phantom)	661	1404	0.11	0.73	152
Paradox Chal. (w/o phantom)	661	0	0.08	-0.46	336
Paradox	1125	1938	0.13	0.73	58
Iowa James	6276	54204	0.06	0.75	357
Signaligner	415	1202	0.09	0.83	203
Foldit	296011	236539	0.13	0.07	230

Acknowledgements

We thank all the players. This work was supported by the National Science Foundation under grant numbers 1652537 and 1629879. Components of research reported in this work were supported by the National Cancer Institute and the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under award numbers UH2CA203780 and UH2EB024407. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

[1] M.-V. Aponte, G. Levieux, and S. Natkin, "Measuring the level of difficulty in single player video games," *Entertainment Computing*, vol. 2, 2011.

[2] A. Sarkar and S. Cooper, "Transforming game difficulty curves using function composition," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019.

[3] M. E. Glickman, "Dynamic paired comparison models with stochastic variances," *Journal of Applied Statistics*, vol. 28, no. 6, pp. 673–689, Aug. 2001.

[4] A. E. Elo, *The rating of chessplayers, past and present*. Arco, 1978.

[5] S. Cooper, S. Deterding, and T. Tsapakos, "Player rating systems for balancing human computation games: testing the effect of bipartiteness," in *Proceedings of the 1st International Joint Conference of DiGRA and FDG*, 2016.

[6] A. Sarkar, M. Williams, S. Deterding, and S. Cooper, "Engagement effects of player rating system-based matchmaking for level ordering in human computation games," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 2017.

[7] R. Kirkman, "pyglicko2: a Python Implementation of the Glicko-2 algorithm," 2010. [Online]. Available: https://code.google.com/p/pyglicko2/

[8] A. Sarkar, V. Sriram, R. Padte, J. Cao, and S. Cooper, "Desire-path inspired procedural placement of coins in a platformer game," in *Proceedings of the Fifth Workshop on Experimental AI in Games*, 2018.

[9] B. Koepnick, J. Flatten, T. Husain, A. Ford, D.-A. Silva, M. J. Bick, A. Bauer, G. Liu, Y. Ishida, A. Boykov, R. D. Estep, S. Kleinfelter, T. Nrgrd-Solano, L. Wei, F. Players, G. T. Montelione, F. DiMaio, Z. Popovic, F. Khatib, S. Cooper, and D. Baker, "De novo protein design by citizen scientists," *Nature*, vol. 570, no. 7761, pp. 390–394, Jun. 2019.

[10] C. Linehan, G. Bellord, B. Kirman, Z. H. Morford, and B. Roche, "Learning curves: analysing pace and challenge in four successful puzzle games," in *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-human Interaction in Play*, 2014, pp. 181–190.