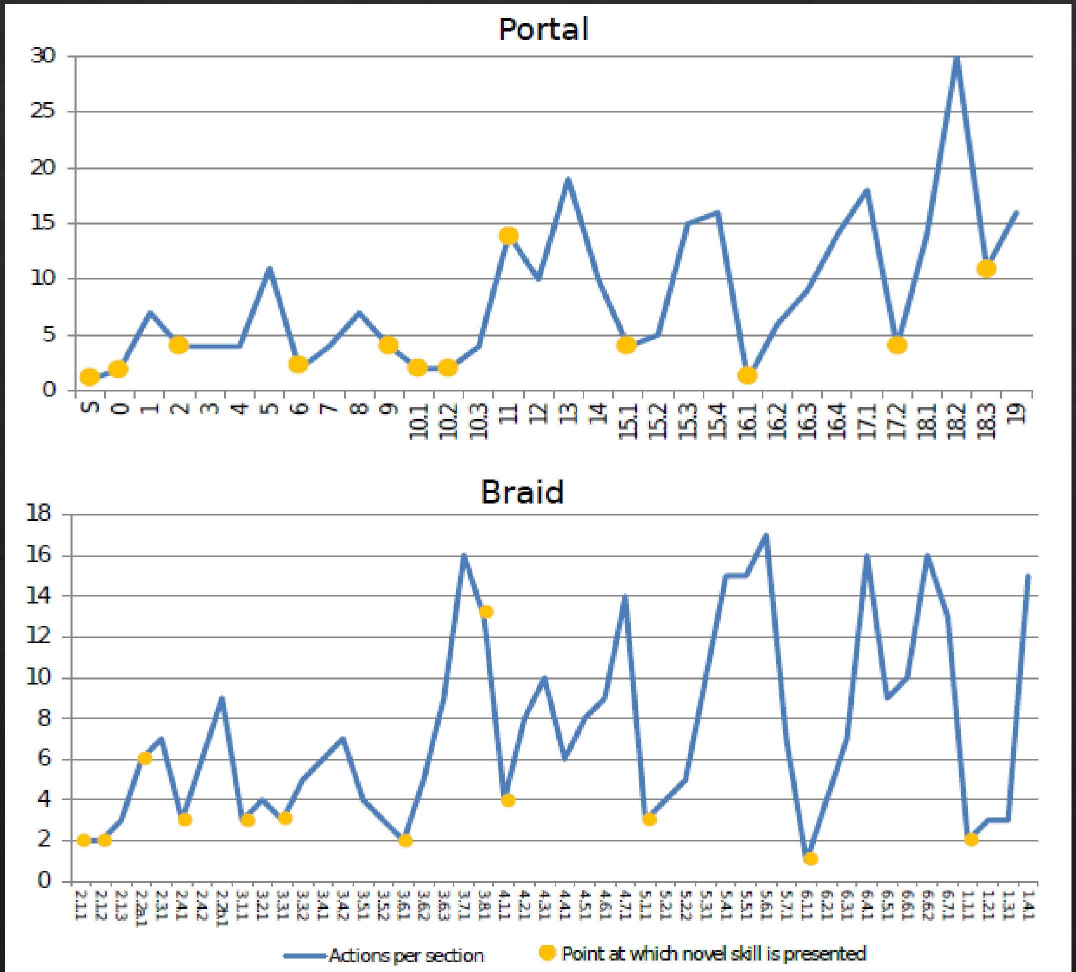
Inferring and Comparing Game Difficulty Curves using Player-vs-Level Match Data

Anurag Sarkar and **Seth Cooper**

Northeastern University

Difficulty Curve

- Defines how a game's difficulty changes over the course of gameplay

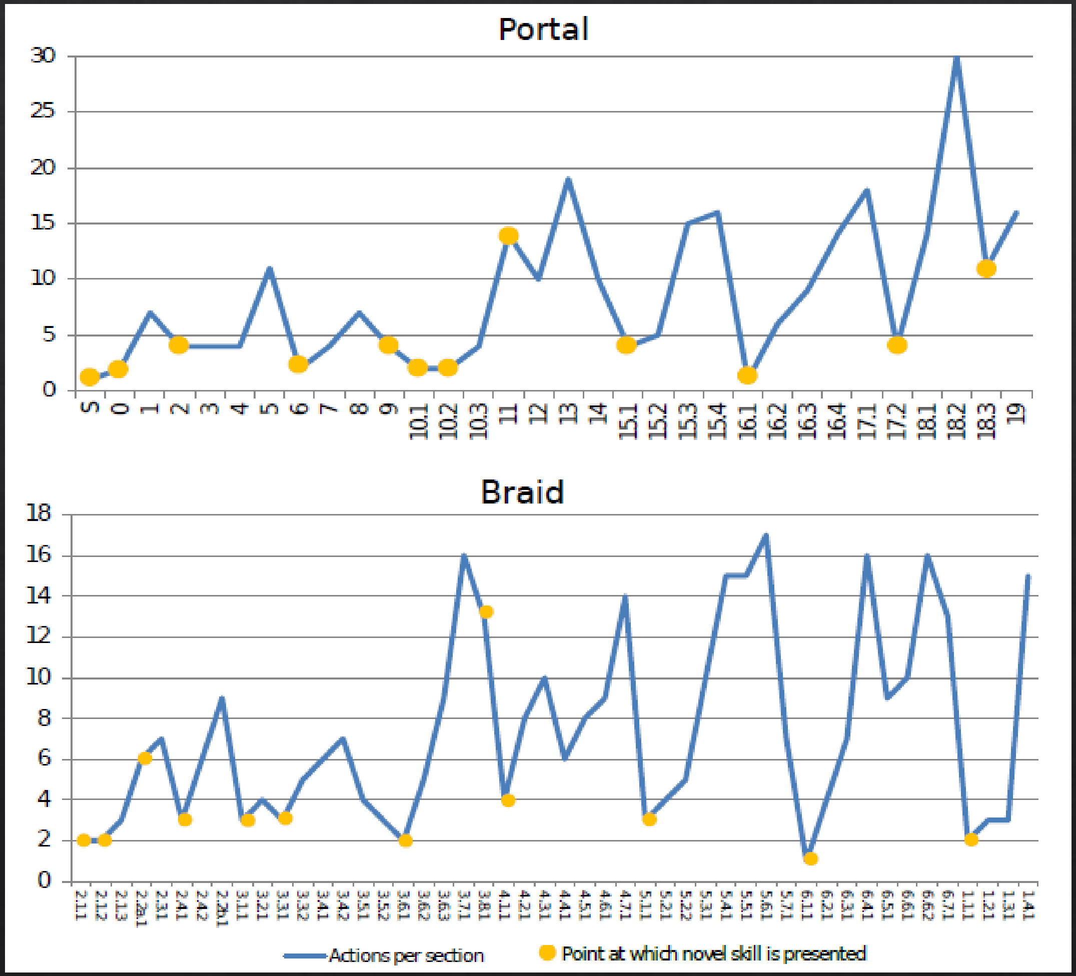


Linehan et al., 2014

Difficulty Curve

- Defines how a game's difficulty changes over the course of gameplay

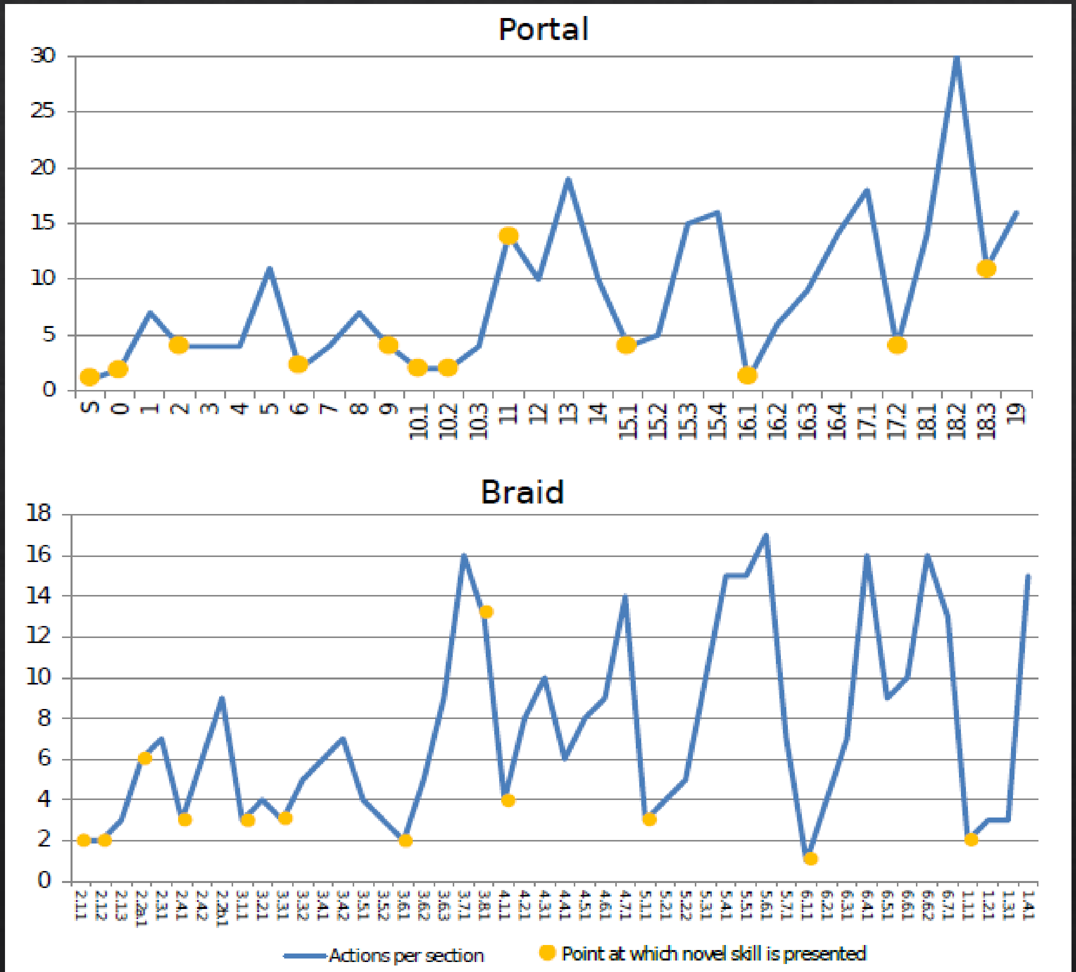
- Curves can be viewed as functions mapping from progression to difficulty



Linehan et al., 2014

Difficulty Curve

- Defines how a game's difficulty changes over the course of gameplay
- Curves can be viewed as functions mapping from progression to difficulty
- Traditional methods of defining curves involve manual refinement through iterative playtesting

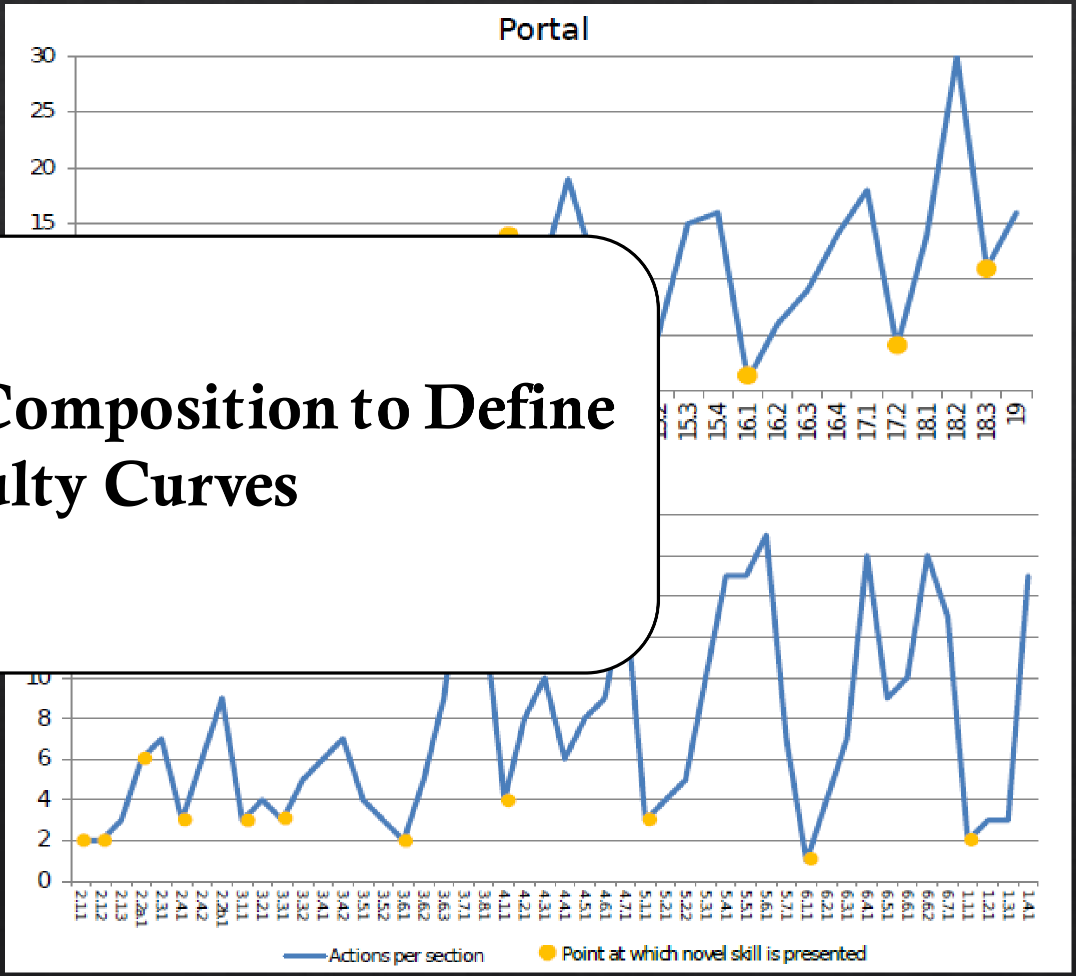


Linehan et al., 2014

Difficulty Curve

- Defines how a game's difficulty changes over the course of gameplay
- Curves can
  from progre
- Traditional
  involve manual refinement through iterative playtesting

PRIOR WORK: Used Function Composition to Define and Transform Difficulty Curves



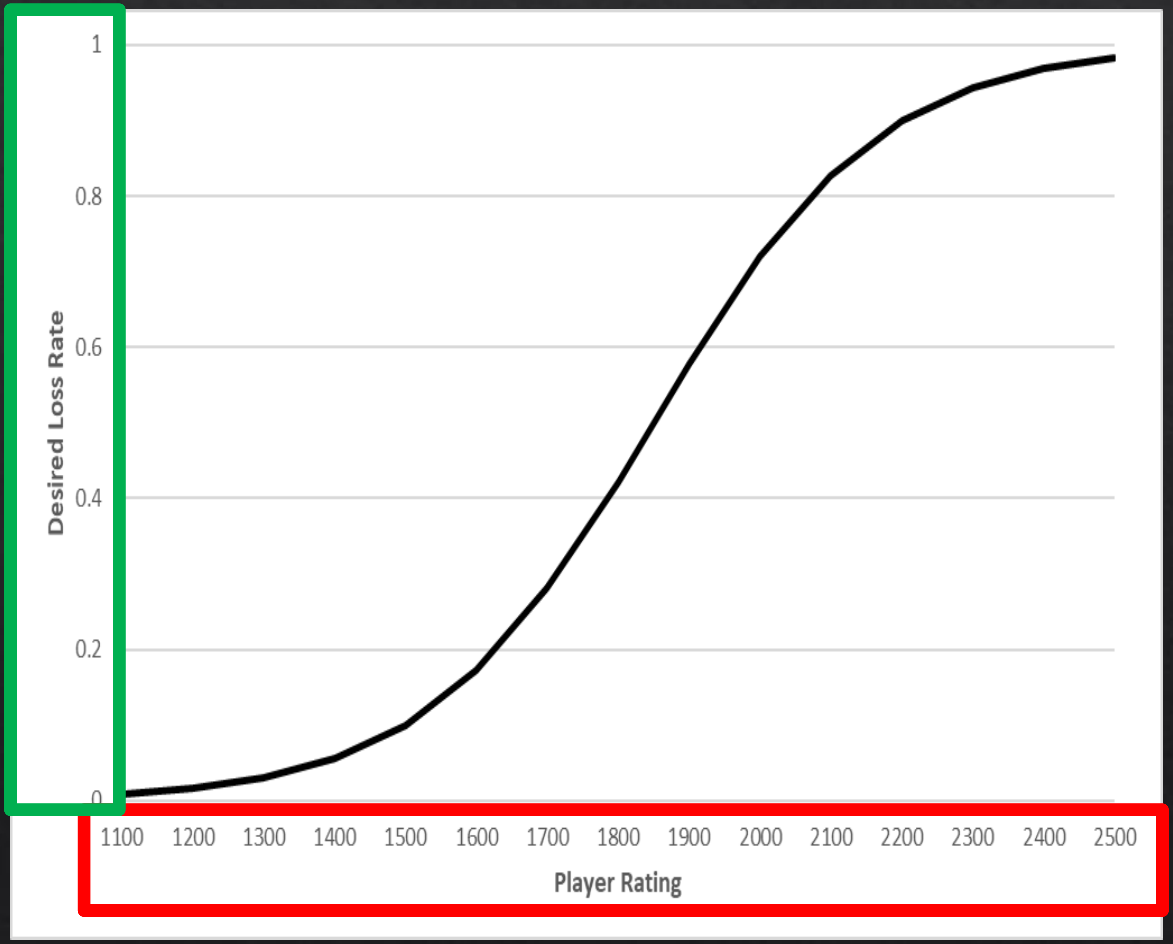
Linehan et al., 2014

Difficulty Curves and Function Composition

Difficulty curve is a function mapping player skill (Glicko-2 rating) to difficulty (desired loss rate)

Baseline Curve	Description
$\boxed{f(x)} = \boxed{\frac{1}{1+e^{\alpha(\beta-x)}}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c

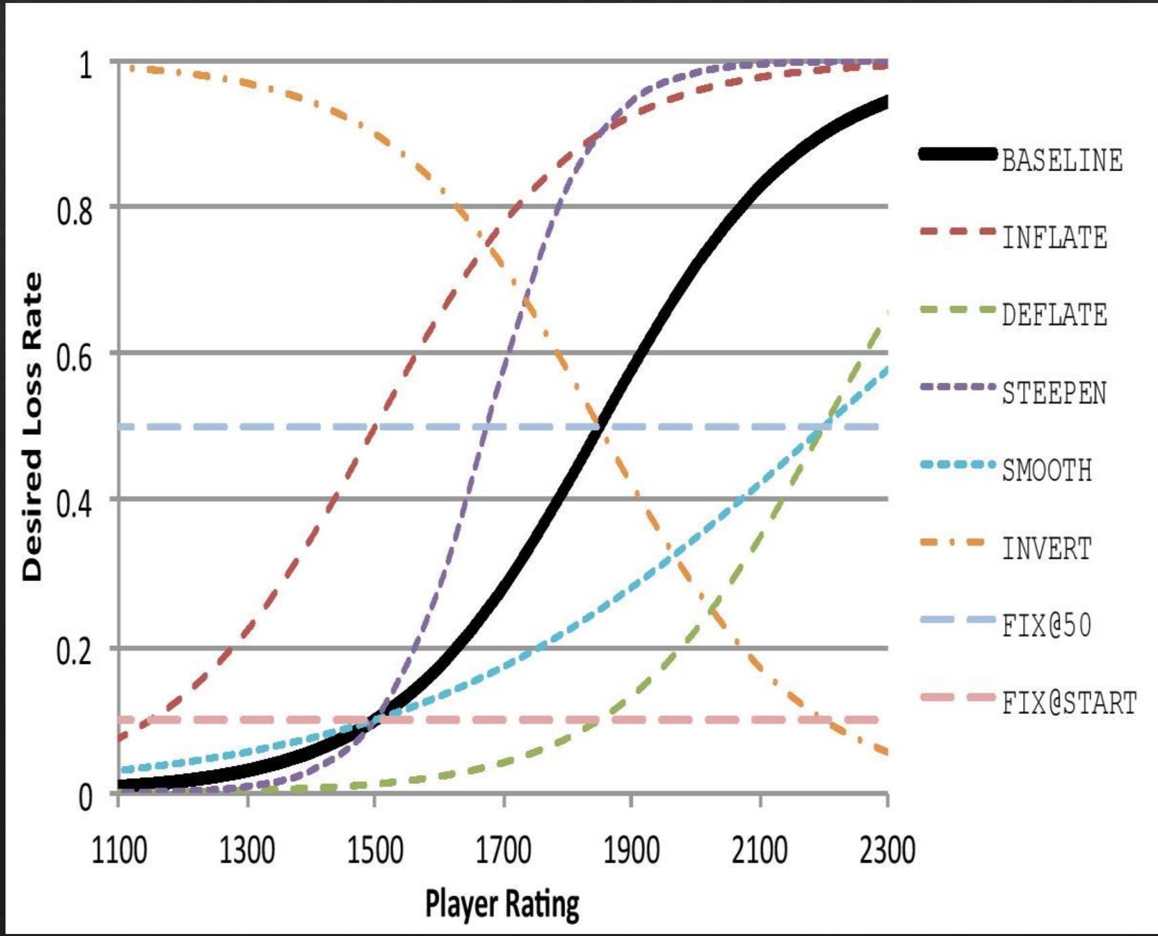


Sarkar and Cooper, "Transforming Game Difficulty Curves using Function Composition", CHI 2019

Difficulty Curves and Function Composition

Difficulty curve is a function mapping player skill (Glicko-2 rating) to difficulty (desired loss rate)

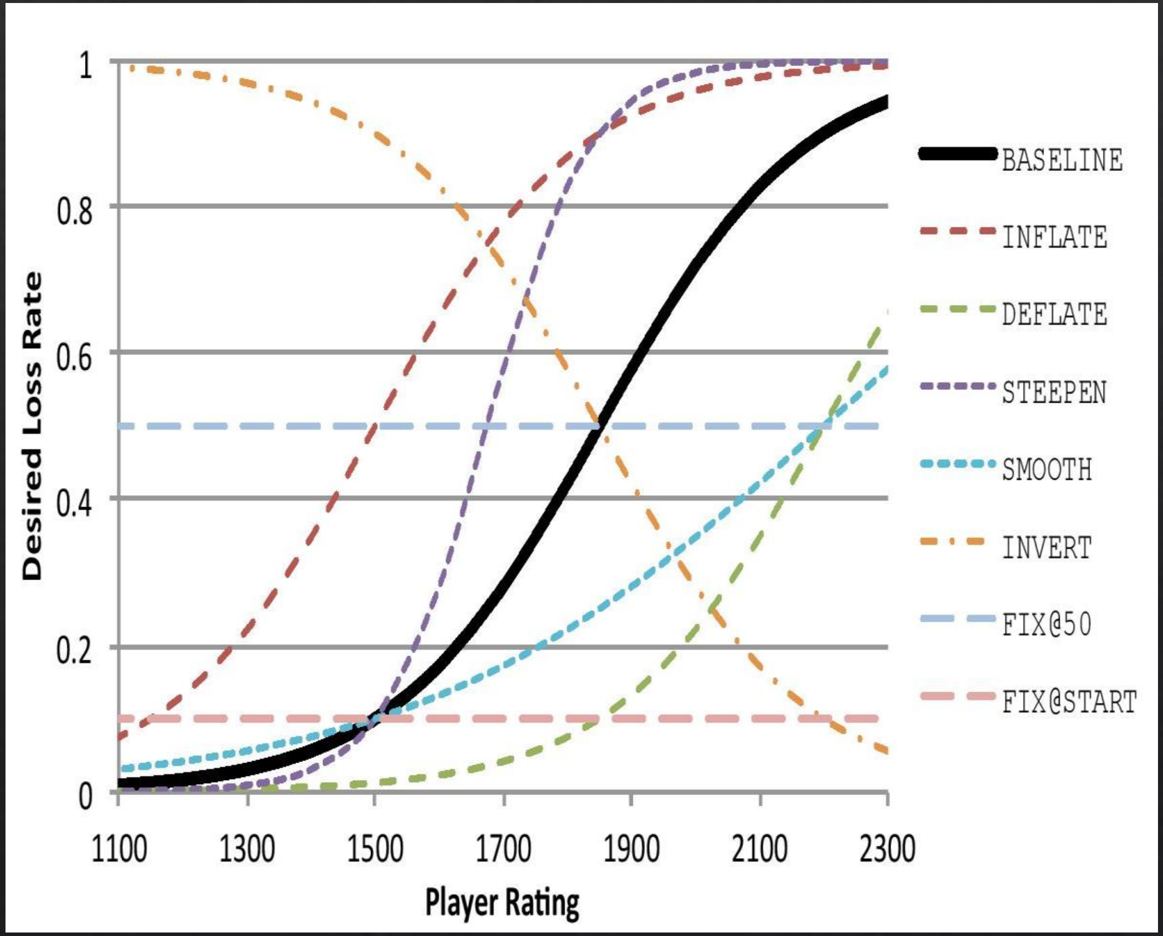
Baseline Curve	Description
$f(x) = \frac{1}{1+e^{\alpha(\beta-x)}}$	Logistic curve
Transformation Functions	**Description**
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c



Sarkar and Cooper, "Transforming Game Difficulty Curves using Function Composition", CHI 2019

Difficulty Curves and Function Composition

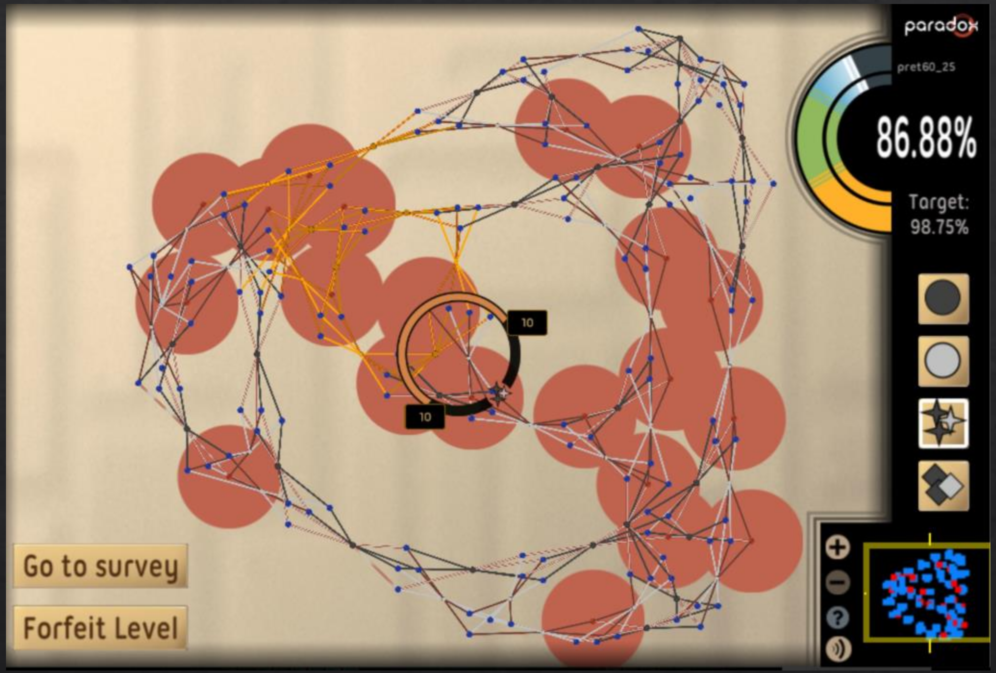
- A formal approach to transforming a game's difficulty curve
- Modified baseline curve to generate new curves and precisely defined transformations
- Transformed curves impacted gameplay and some improved engagement



Sarkar and Cooper, "Transforming Game Difficulty Curves using Function Composition", CHI 2019

Drawback

- Only used with a single game *(Paradox)*
- Curves and transformations defined with respect to *Paradox's* DDA system and Glicko-2 ratings



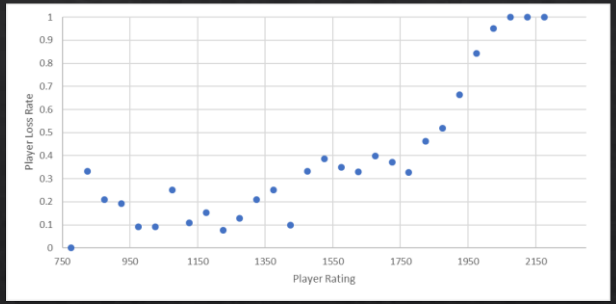
This Work

- Extends prior work to infer difficulty curves in a game-independent manner
- Uses same formulation for curves as prior work; enables use of function composition to compare curves from different games
- Applicable to games with either static or dynamic difficulty
- Introduces use of *phantom matches* (traditional playback does not work)

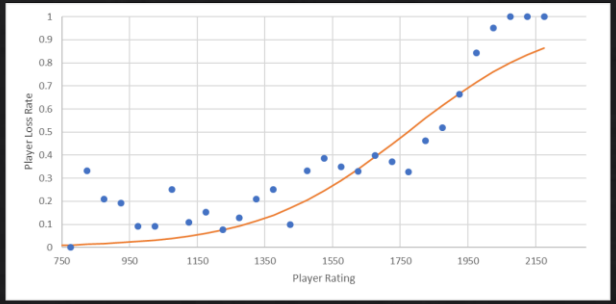
Approach

Timestamp	PlayerID	Level	Score
1535387920891	pl_1	hole6	1
1535387924221	pl_2	hole6	1
1535387944903	pl_3	hole6	1
1535387944959	pl_2	hole10	1
1535387945548	pl_1	hole10	1
1535387967345	pl_3	hole10	1
1535388008835	pl_2	flat50-50	0
1535388014748	pl_2	gen_tree_sa	1
1535388038068	pl_2	flat30-10	0
1535388046404	pl_1	flat50-50	0

- Collecting gameplay data



- Sampling from player skill to difficulty
  - Playback
  - Phantom match generation



- Fitting curves to sampled data

Gameplay Data

- Match data with instances of players playing levels treated as PvL matches
- Each entry consists of
  - Timestamp
  - Player ID
  - Level ID
  - Player win/loss (1/0)

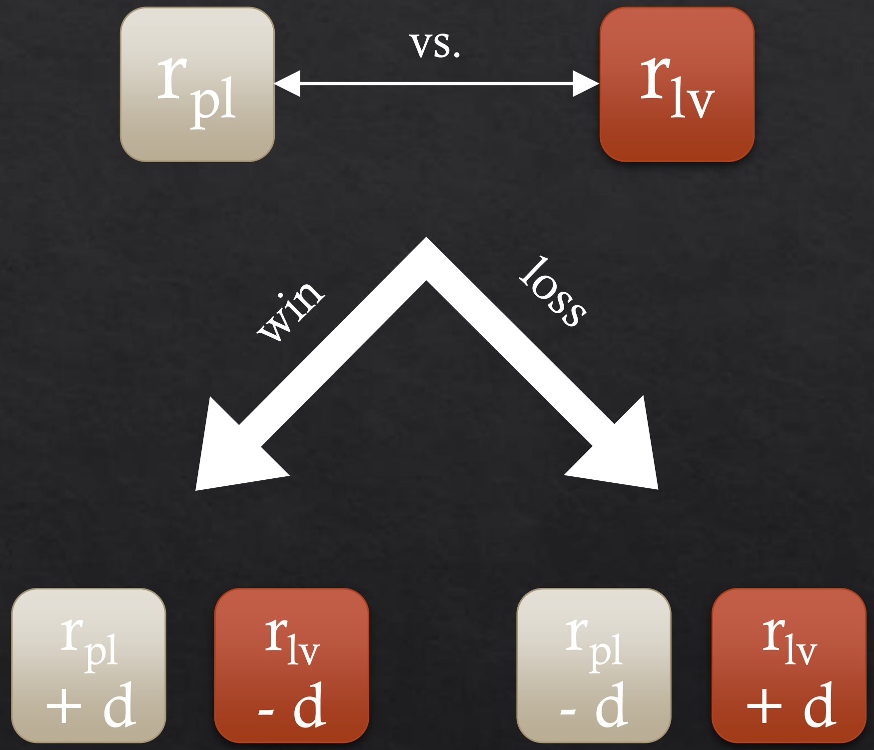
Timestamp	PlayerID	Level	Score
1535387920891	pl_1	hole6	1
1535387924221	pl_2	hole6	1
1535387944903	pl_3	hole6	1
1535387944959	pl_2	hole10	1
1535387945548	pl_1	hole10	1
1535387967345	pl_3	hole10	1
1535388008835	pl_2	flat50-50	0
1535388014748	pl_2	gen_tree_sa	1
1535388038068	pl_2	flat30-10	0
1535388046404	pl_1	flat50-50	0

Sampling from Player Skill to Difficulty

- Want to determine game difficulty curve from this data i.e. fit curves to it
- In our formulation, curves are functions mapping player skill to difficulty
- To fit curves, we sample this mapping
  - Player skill → Glicko-2 rating
  - Difficulty → Player's loss rate
- Fitting curves involves playback and phantom match generation

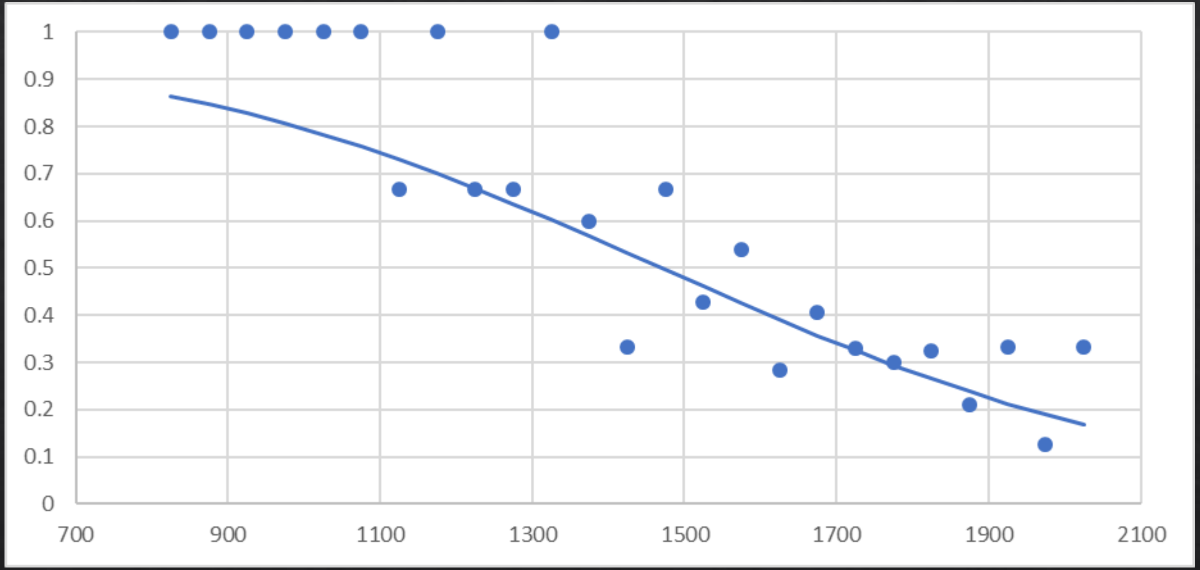
Playback

- Each player and level assigned Glicko-2 ratings (init=1500)
  - Player rating → Skill
  - Level rating → Difficulty
- Compare ratings to compute player's chance of losing level i.e. level difficulty for that player
- Ratings updated based on PvL outcomes
- Each match creates a sample of the game's difficulty curve by recording current player rating and if player won or lost
- Samples grouped into bins by rating and the mean player loss rate for each bin is computed



Survivorship

- In match data, harder levels mostly have matches vs. high skill players
- Only skilled players survive past easy and moderately difficult levels
    - Match up with harder levels in the game
        - Harder levels end up with low ratings



Solution: Phantom Matches

- We create a *phantom match* for each PvL pairing that did not actually occur during gameplay

- For each such pairing between player P and level L, to determine result of phantom match:
  - We note the lowest rated player X that beat level L
  - If P's final rating >= X's rating, then P wins
  - Else P loses

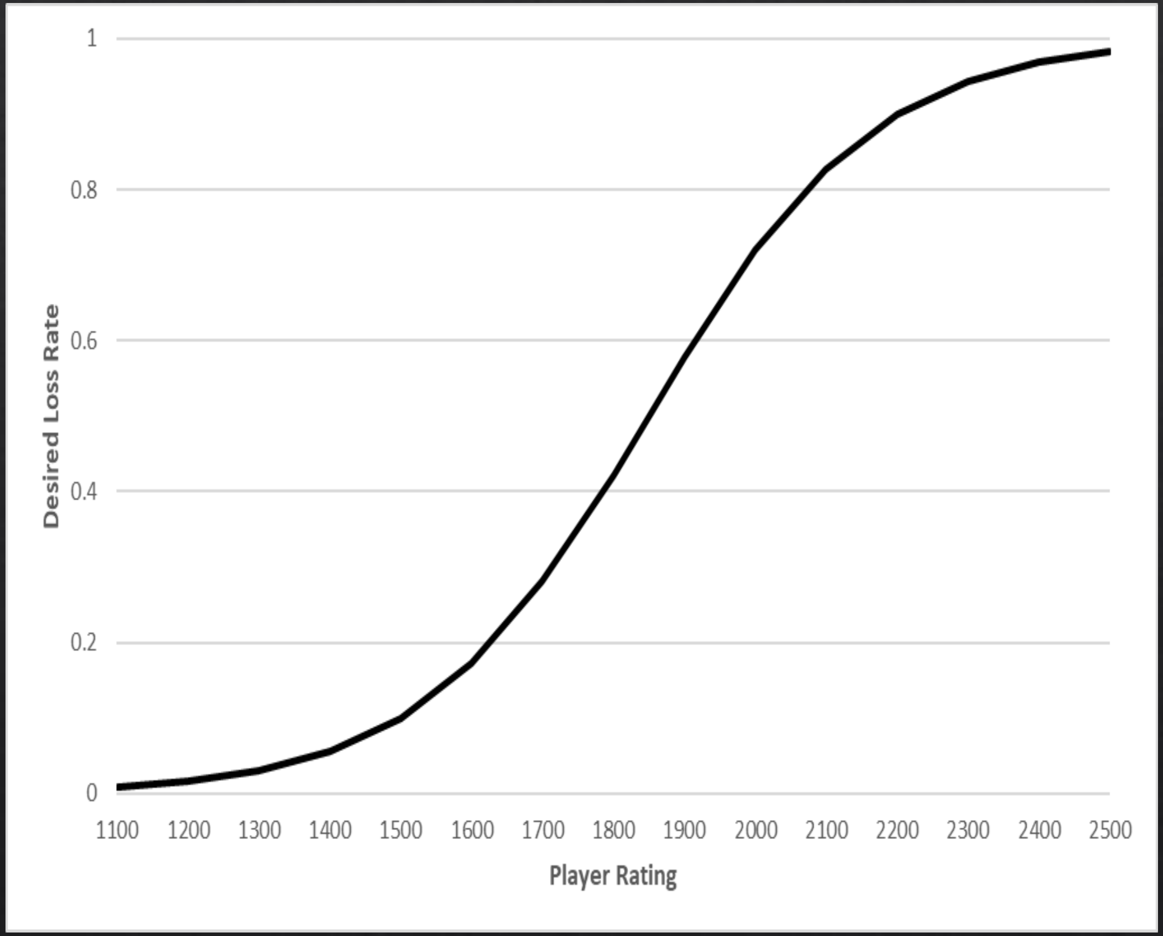
- Phantom matches let harder levels get back wins against low skill players who dropped out

- Combined match data = Real matches + Phantom matches

Fitting Curves to Sampled Data

- To fit a curve to the data, we used a logistic function mapping player rating to loss rate
- Player's loss rate measures difficulty as it determines how hard the next match will be
- Curve taken from prior work in Paradox:

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

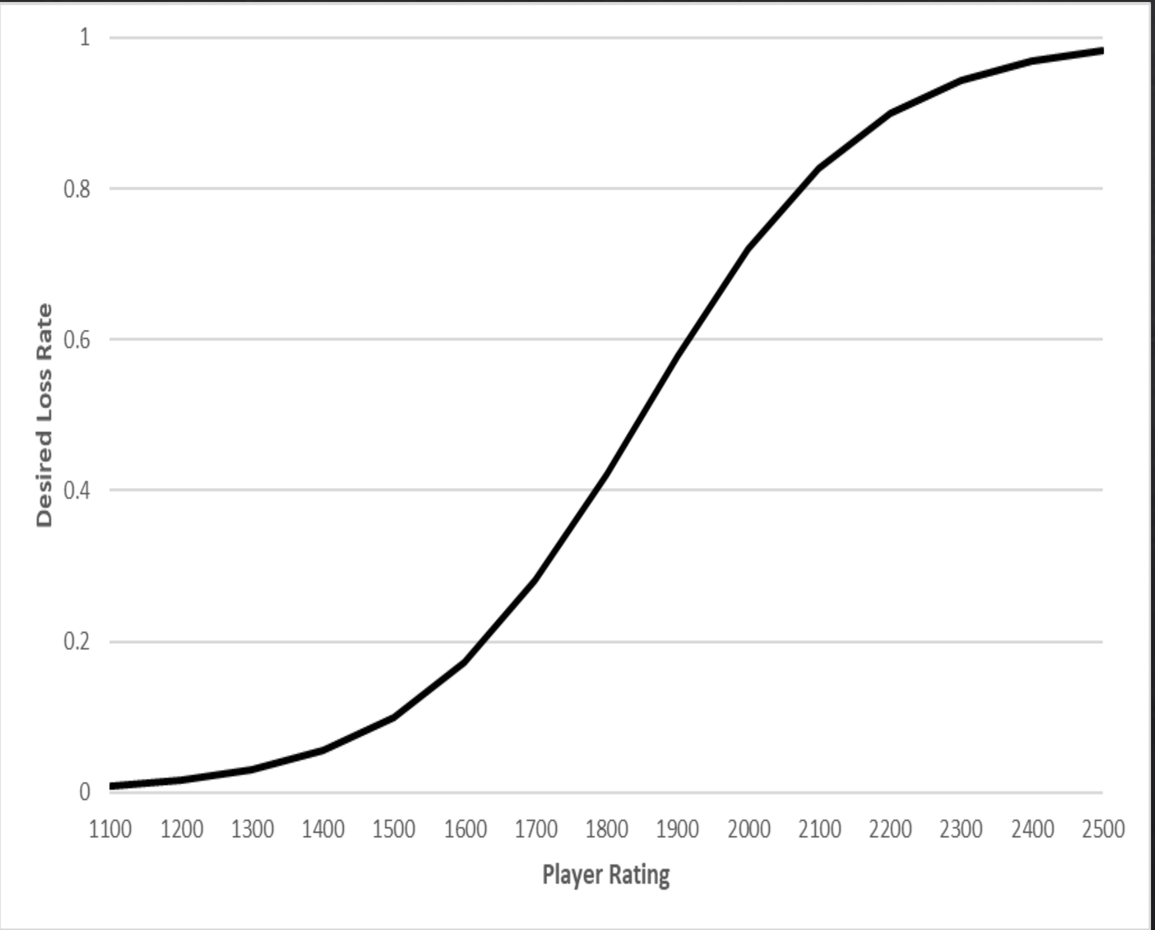


Function Composition

Baseline Curve	Description
$f(x) = \frac{1}{1+e^{\alpha(\beta-x)}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c

Curve Name	Function	Description
BASELINE	f	baseline curve
INFLATE	$f \circ t_{r_d}$	inflate difficulty via shifting curve left by a constant
DEFLATE	$f \circ t_{-r_d}$	deflate difficulty via shifting curve right by a constant
STEEPEN	$f \circ s_{2,r_l}$	steepen difficulty by increasing curve's rate of change
SMOOTH	$f \circ s_{0.5,r_l}$	smooth difficulty by decreasing curve rate's rate of change

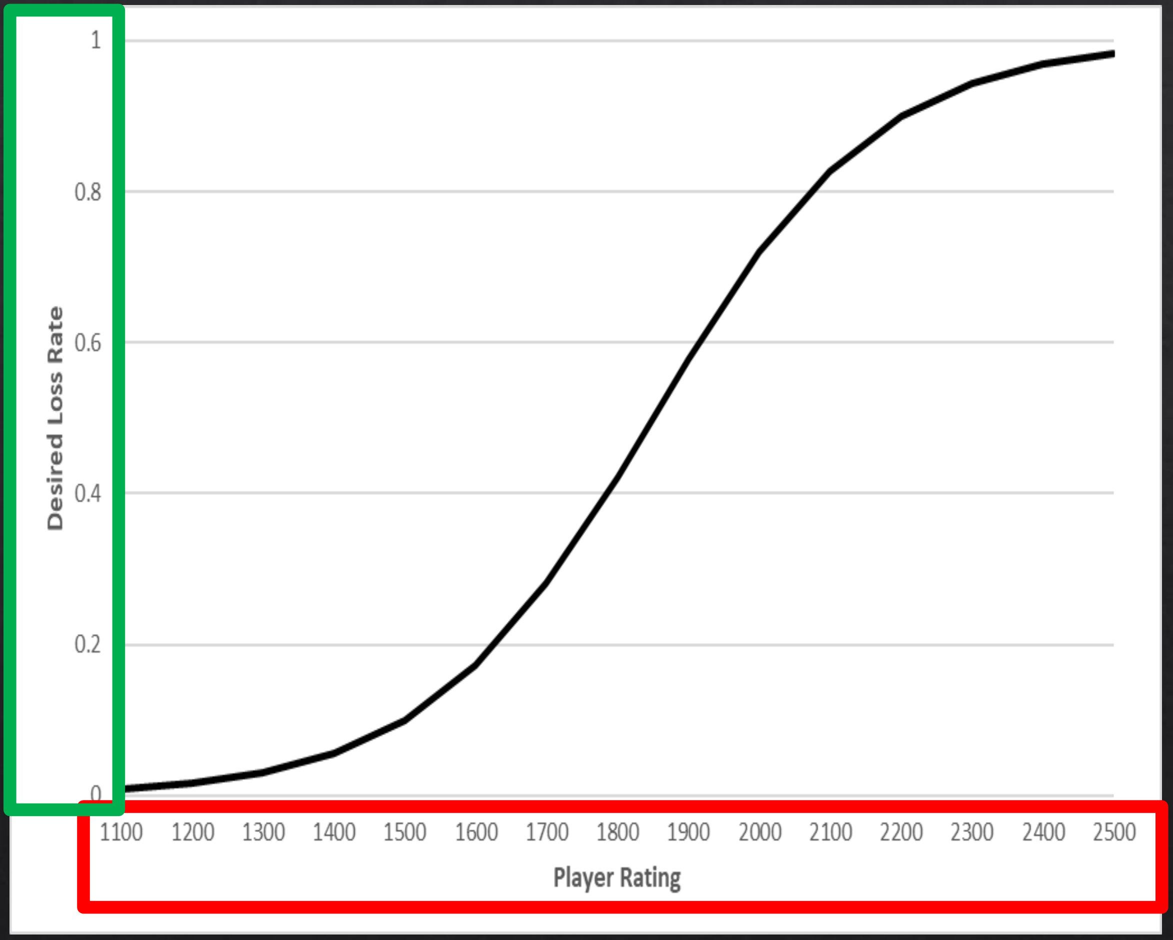


Function Composition

Baseline Curve	Description
$\boxed{f(x)} = \boxed{\frac{1}{1+e^{\alpha(\beta - x)}}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c

Curve Name	Function	Description
BASELINE	f	baseline curve
INFLATE	$f \circ t_{r_d}$	inflate difficulty via shifting curve left by a constant
DEFLATE	$f \circ t_{-r_d}$	deflate difficulty via shifting curve right by a constant
STEEPEN	$f \circ s_{2,r_l}$	steepen difficulty by increasing curve's rate of change
SMOOTH	$f \circ s_{0.5,r_l}$	smooth difficulty by decreasing curve rate's rate of change

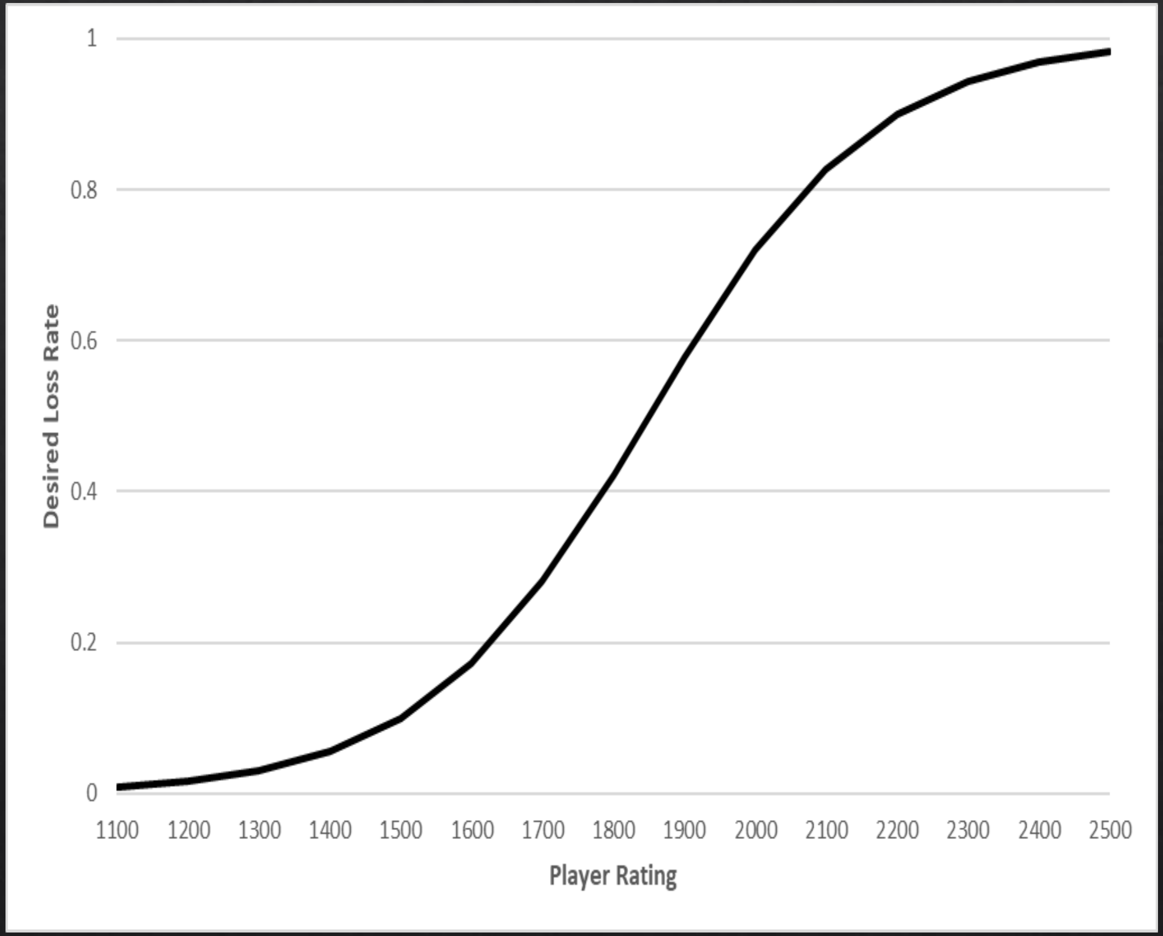


Function Composition

Baseline Curve	Description
$f(x) = \frac{1}{1+e^{\alpha(\beta-x)}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c

Curve Name	Function	Description
BASELINE	f	baseline curve
INFLATE	$f \circ t_{r_d}$	inflate difficulty via shifting curve left by a constant
DEFLATE	$f \circ t_{-r_d}$	deflate difficulty via shifting curve right by a constant
STEEPEN	$f \circ s_{2,r_l}$	steepen difficulty by increasing curve's rate of change
SMOOTH	$f \circ s_{0.5,r_l}$	smooth difficulty by decreasing curve rate's rate of change

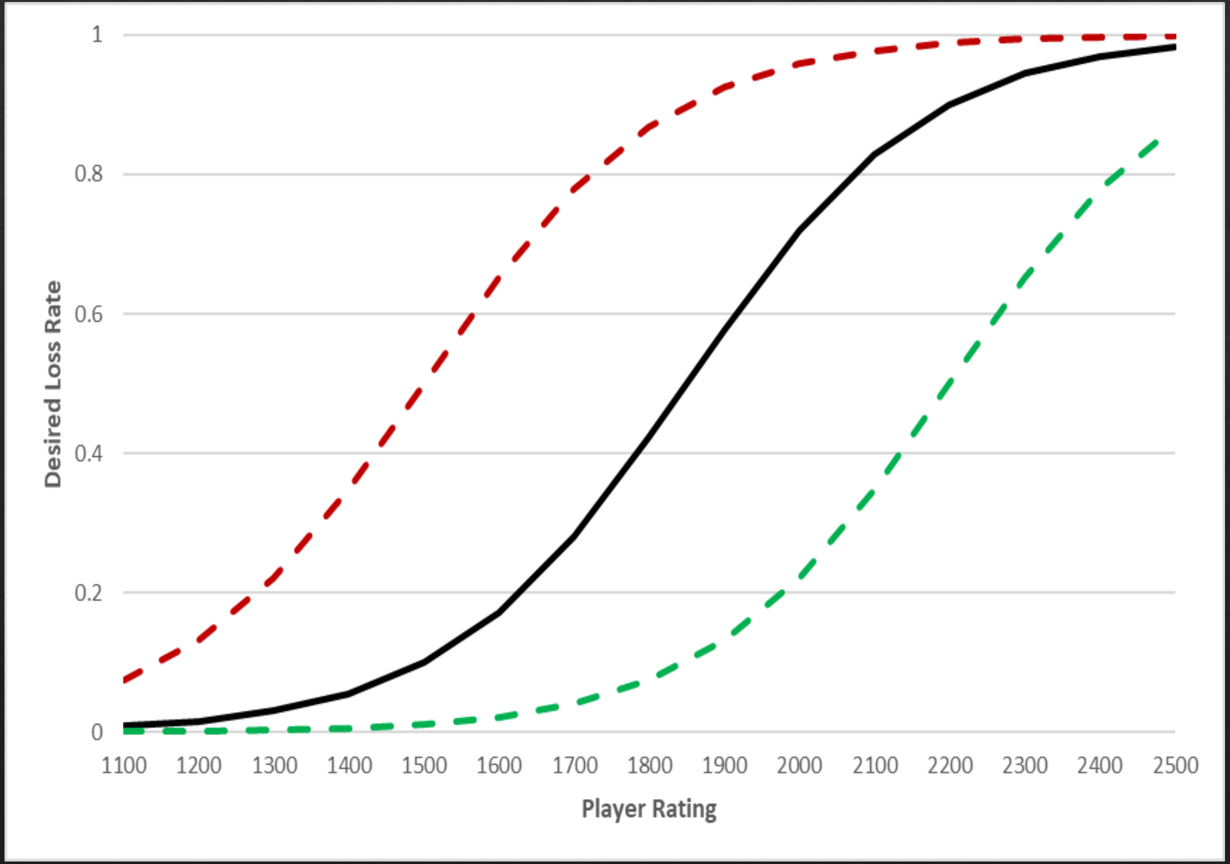


Function Composition

Baseline Curve	Description
$f(x) = \frac{1}{1+e^{\alpha(\beta-x)}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c

Curve Name	Function	Description
BASELINE	f	baseline curve
INFLATE	$f \circ t_{r_d}$	inflate difficulty via shifting curve left by a constant
DEFLATE	$f \circ t_{-r_d}$	deflate difficulty via shifting curve right by a constant
STEEPEN	$f \circ s_{2,r_t}$	steepen difficulty by increasing curve's rate of change
SMOOTH	$f \circ s_{0.5,r_t}$	smooth difficulty by decreasing curve rate's rate of change

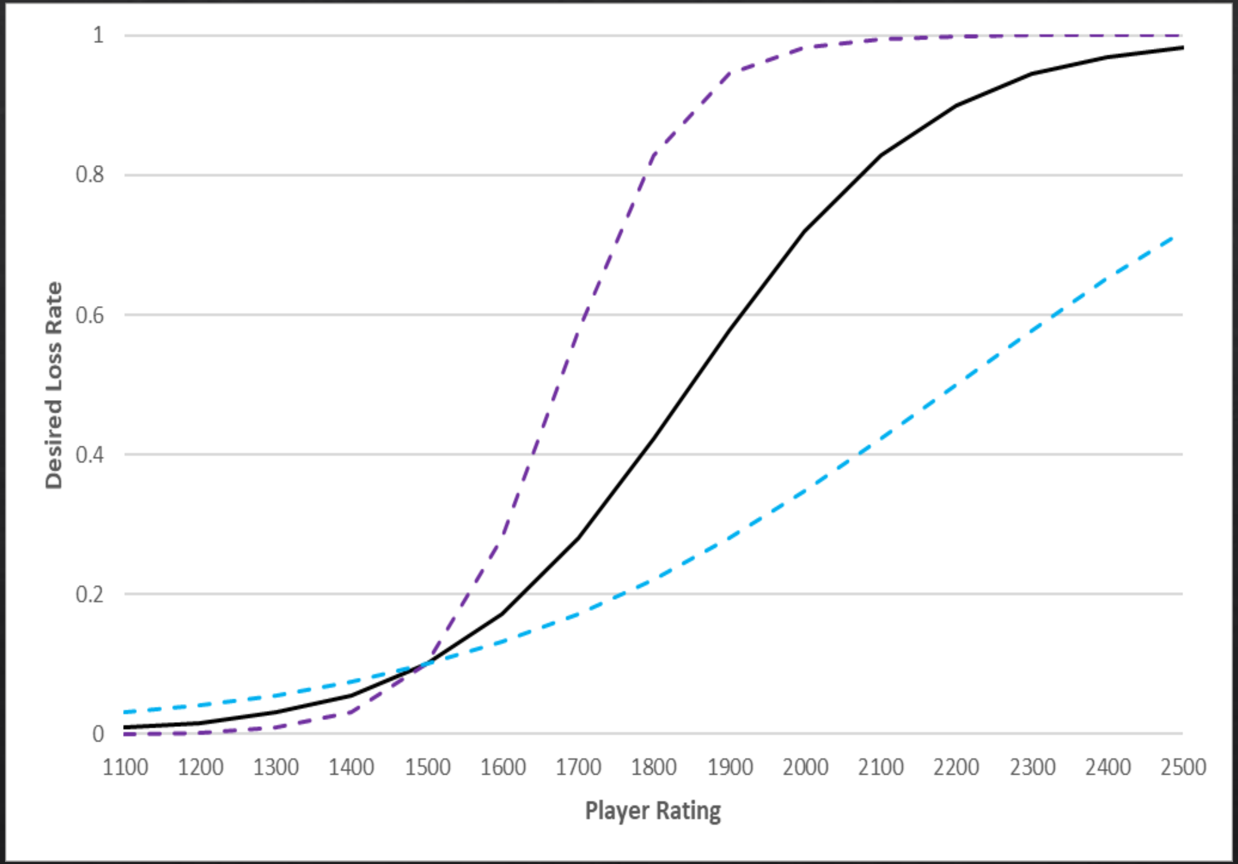


Function Composition

Baseline Curve	Description
$f(x) = \frac{1}{1+e^{\alpha(\beta - x)}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x - c) + c$	Scale by σ around c

Curve Name	Function	Description
BASELINE	f	baseline curve
INFLATE	$f \circ t_{r_d}$	inflate difficulty via shifting curve left by a constant
DEFLATE	$f \circ t_{-r_d}$	deflate difficulty via shifting curve right by a constant
STEEPEN	$f \circ s_{2,r_l}$	steepen difficulty by increasing curve's rate of change
SMOOTH	$f \circ s_{0.5,r_l}$	smooth difficulty by decreasing curve rate's rate of change



Function Composition

Baseline Curve	Description
$f(x) = \frac{1}{1+e^{\alpha(\beta-x)}}$	Logistic curve

Transformation Functions	Description
$t_\delta(x) = x + \delta$	Translate by δ
$s_{\sigma,c}(x) = \sigma(x-c) + c$	Scale by σ around c

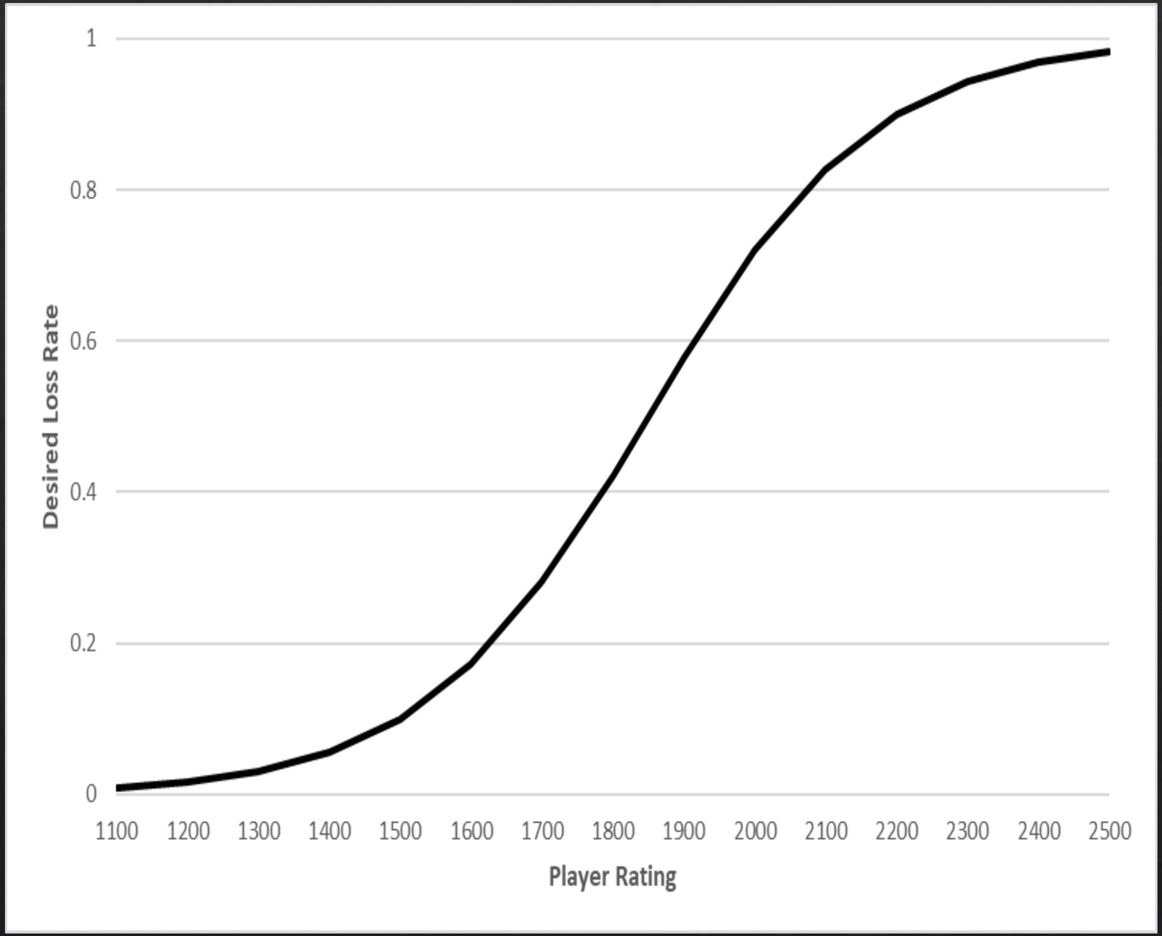
Curve Name	Function	Description
BASELINE	f	baseline curve
INFLATE	$f \circ t_{r_d}$	inflate difficulty via shifting curve left by a constant
DEFLATE	$f \circ t_{-r_d}$	deflate difficulty via shifting curve right by a constant
STEEPEN	$f \circ s_{2,r_l}$	steepen difficulty by increasing curve's rate of change
SMOOTH	$f \circ s_{0.5,r_l}$	smooth difficulty by decreasing curve rate's rate of change

- By composing functions as $f \circ s_{\sigma,c} \circ t_\delta$, we get curves parameterized by δ and σ
- Fit curves to data by optimizing δ and σ to minimize RMSE between curve value at bin centers and mean player loss rate

Phantom Match Validation

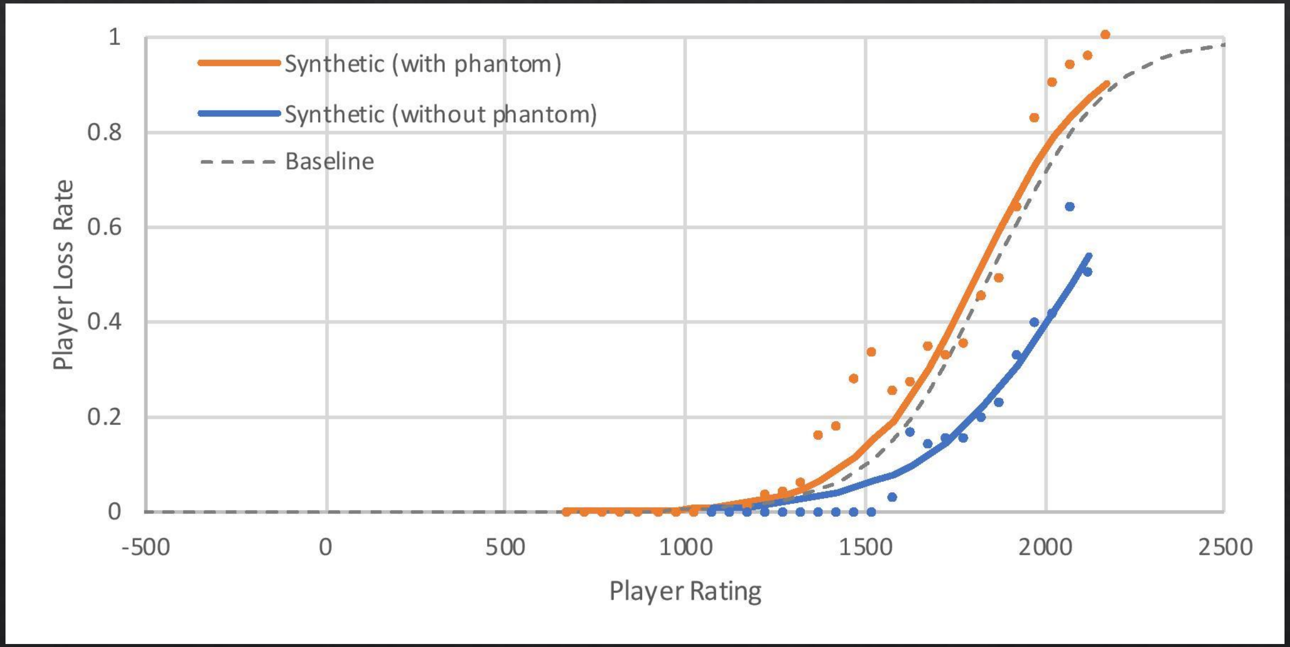
- For validation, we used
  - Synthetically generated dataset
  - Dataset from past trials using *Paradox*

- In both cases, we know the underlying baseline difficulty curve ($\delta = 0$, $\sigma = 1$)



Synthetic Data

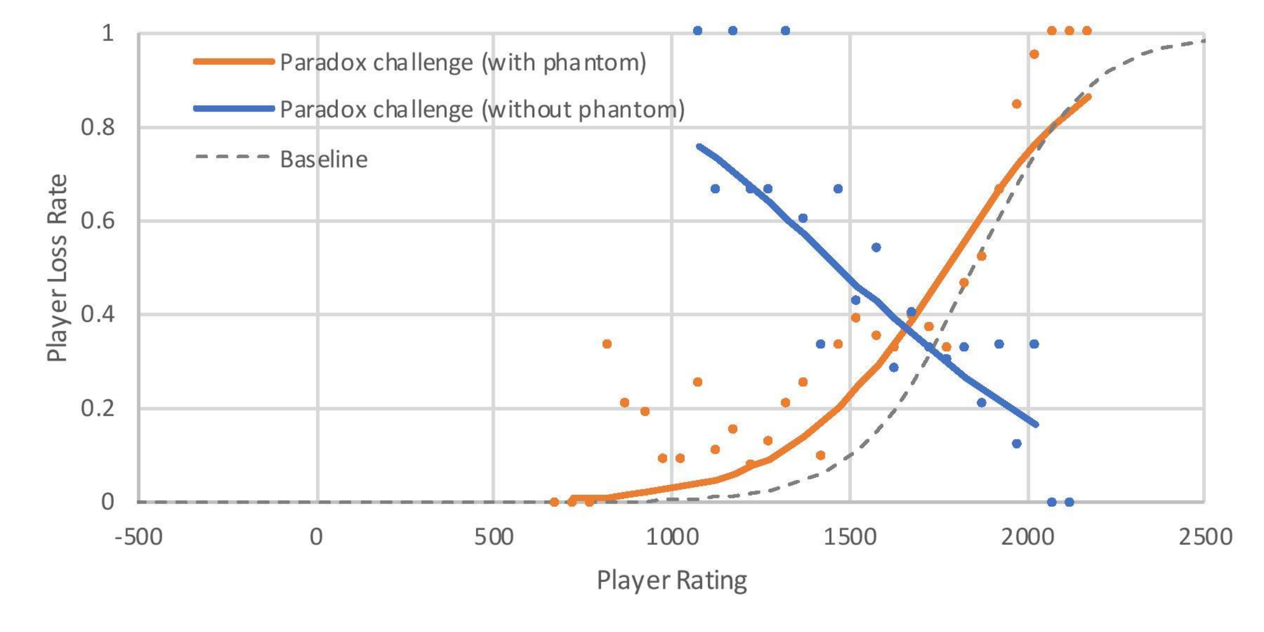
- 50 generated players rated randomly from 900 to 2100
- 61 generated levels rated from 0 to 3000 in increments of 50
- To generate synthetic data
  - Randomly select player
  - Determine best level to serve
  - Simulate match result using the rating system
  - If player loses, stops playing via a drop rate
  - Continue simulation until no players remain



Paradox

- Gameplay data from challenge portion of *Paradox* gathered from prior work

- Used baseline curve to perform DDA, so applying phantom matches should help recover this curve



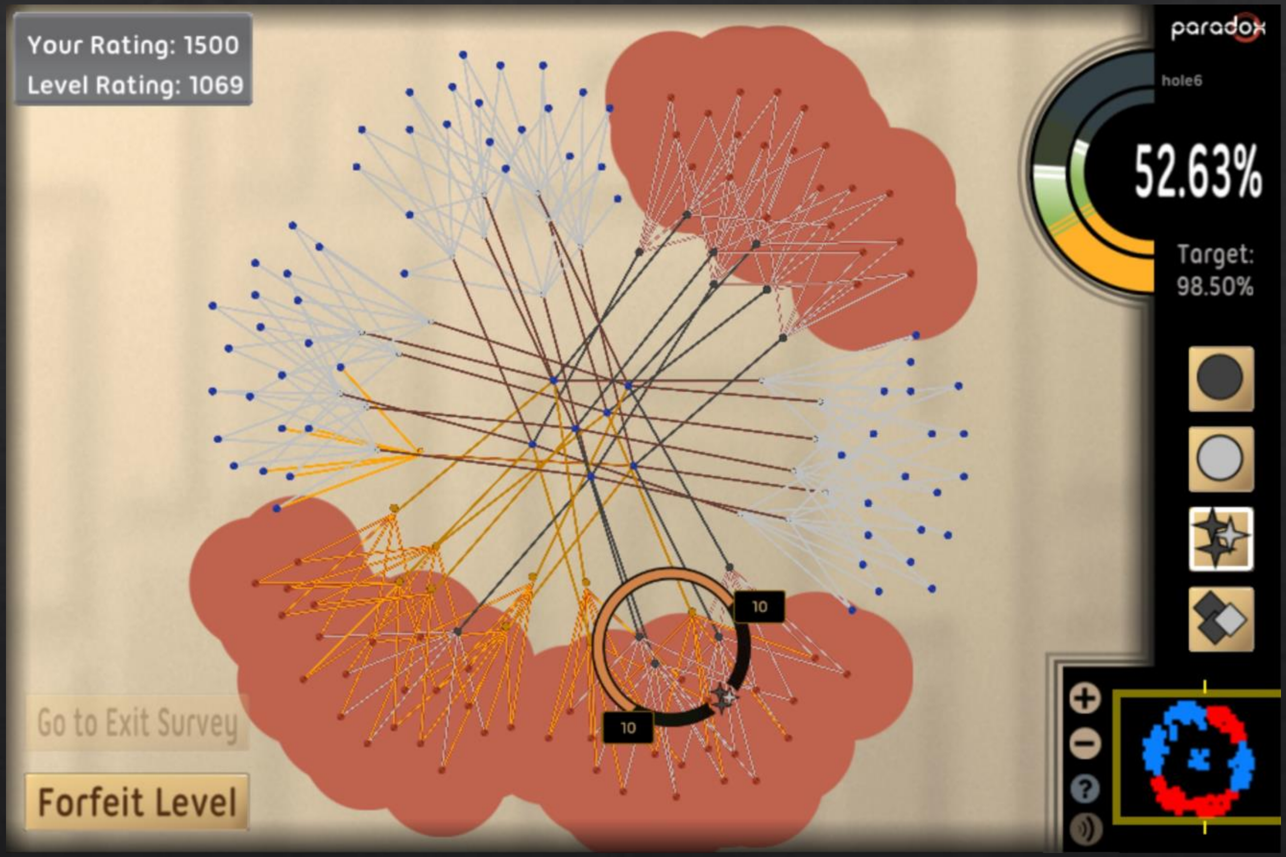
Games

- Used gameplay data from 4 games to apply our approach
  - *Paradox*
  - *Iowa James*
  - *Signaligner*
  - *Foldit*

- Existing dataset for first three gathered through Amazon Mechanical Turk
- Existing *Foldit* data gathered through website https://fold.it

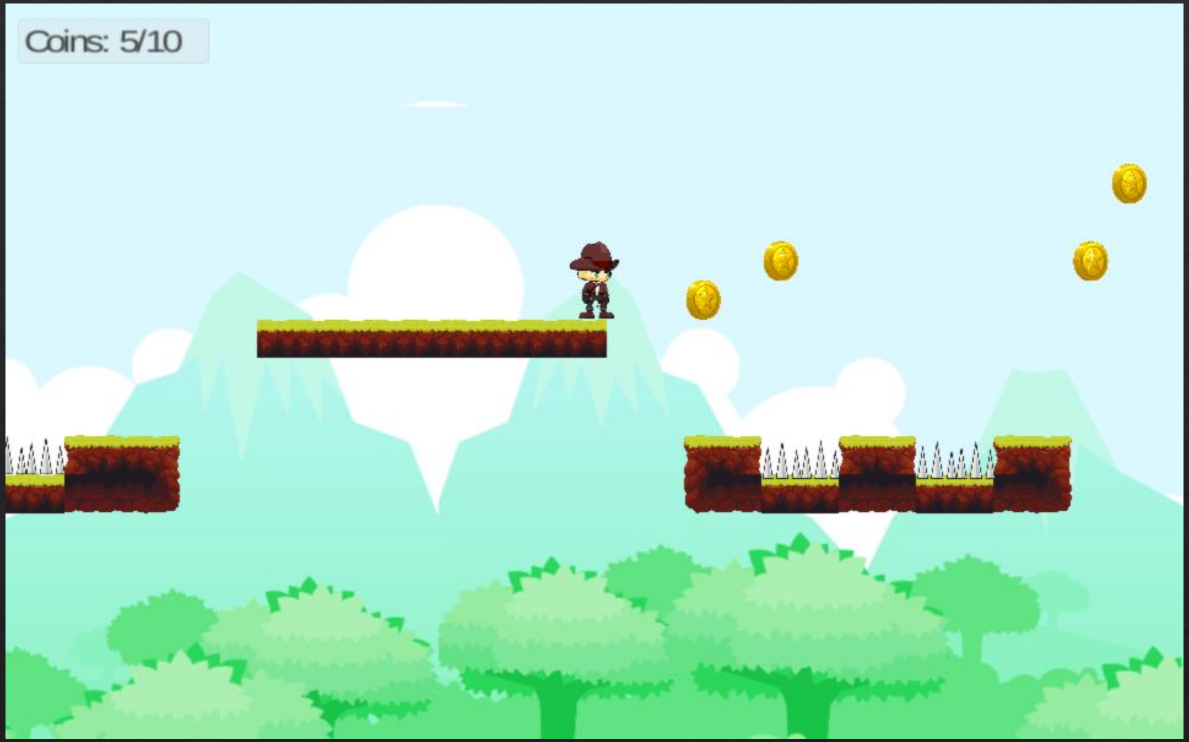
Paradox

- 2D human computation puzzle game
- Each level is a MAX-SAT problem with a target number of constraints to be satisfied
- Players assign values to variables to solve constraints
- Score: percentage of satisfied constraints
- Goal: complete level by reaching target score
- Player wins vs. a level by completing it
- 8 tutorial levels → fixed order
- 50 challenge levels → dynamic order



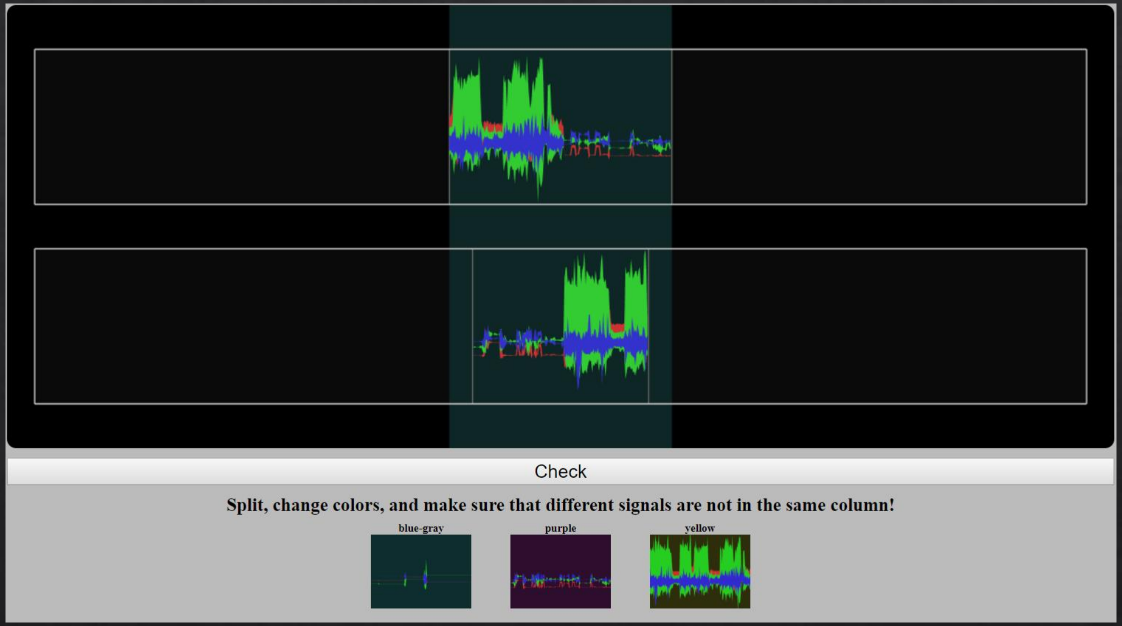
Iowa James

- Basic platformer with 14 levels following an increasing difficulty ordering
- Each level has hazards that player must avoid
- Goal: reach treasure chest at the end of the level
- Player wins vs. a level by reaching the chest, regardless of number of deaths
- Player loses by quitting the level without reaching the chest



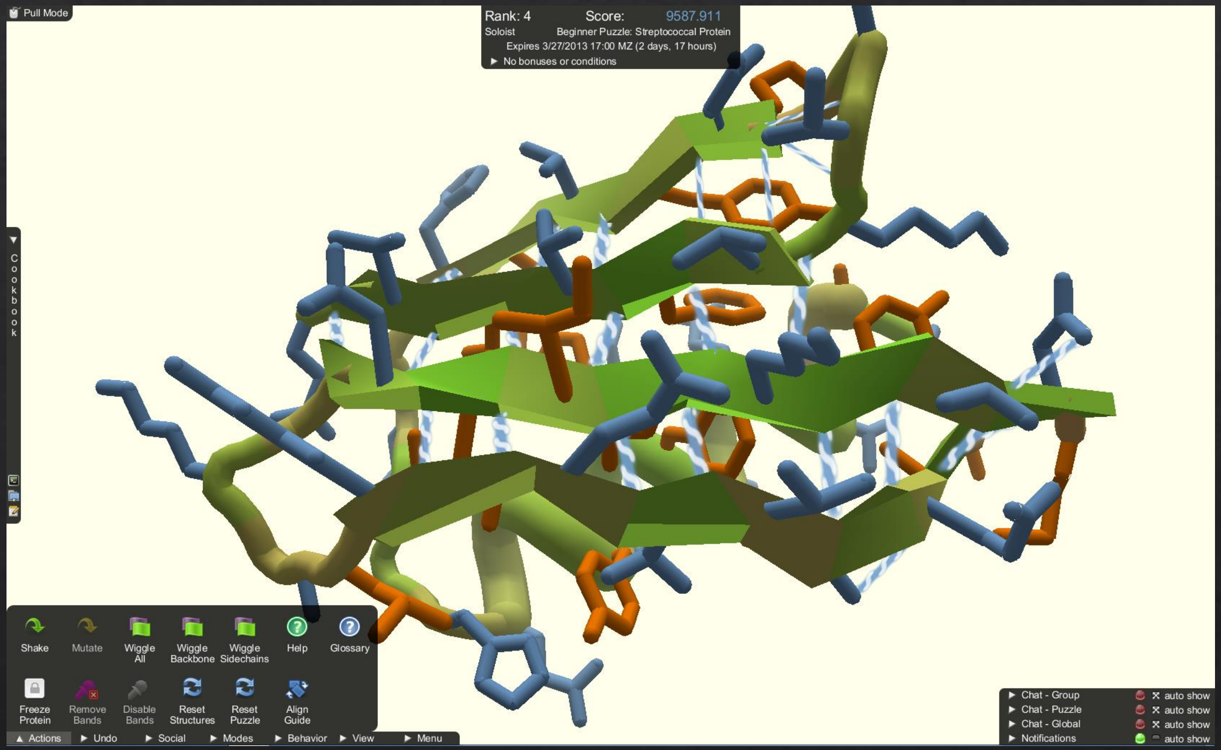
Signaligner

- 2D human computation puzzle game
- Players annotate raw accelerometer data with activity labels
- Group together similar looking data signal blocks by splitting, merging or aligning
- 4 tutorial levels and 1 of 7 possible challenge levels
  - Tutorial → multiple attempts to submit correct answer
  - Challenge → one attempt; players win if they submit correct answer



Foldit

- Human computation puzzle game based on protein folding
- Players interactively fold and pack protein structures
- 37 tutorial levels were used for this analysis
- Score: energy of the current fold
- Players win a level by reaching the target score
- Tutorial progression is same for all players but players have choices at branching points and can replay previous levels

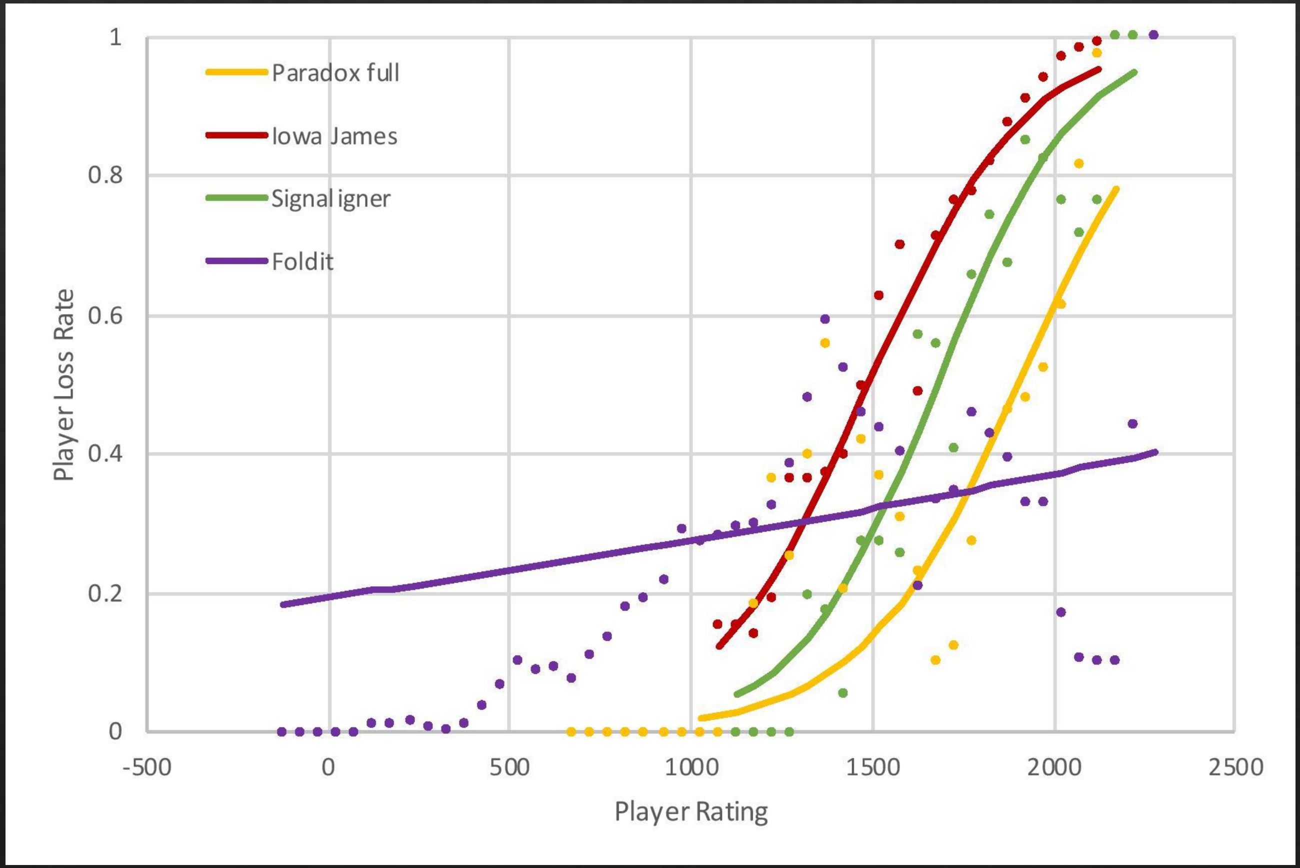


Curve Comparisons and Transformations

- Using our approach we fit difficulty curves on data for all 4 games

- Function composition-based terminology
  - *Foldit* has the smoothest curve
  - Other 3 games have steeper curves
  - Of these 3, *Iowa James* has an inflated curve compared to *Signaligner* and *Paradox*
  - *Paradox* has the most deflated curve

- *Paradox* and *Foldit* curves have the highest error
  - Single curve does not fit data well and multiple curves may be needed



Conclusion and Future Work

- A method of inferring a game’s difficulty curve using gameplay data
- Applicable to fixed/dynamic/hybrid difficulty progressions
- Enables comparing difficulty curves across games using a common, precise vocabulary

Conclusion and Future Work

- A method of inferring a game's difficulty curve using gameplay data
- Applicable to fixed/dynamic/hybrid difficulty progressions
- Enables comparing difficulty curves across games using a common, precise vocabulary

- For future work, fit multiple curves (e.g. sawtooth pattern) to each game
- Can help infer separate curves for segments with different difficulty
- Use continuous PvL outcomes rather than binary win/loss

Conclusion and Future Work

- A method of inferring a game's difficulty curve using gameplay data
- Applicable to fixed/dynamic/hybrid difficulty progressions
- Enables comparing difficulty curves across games using a common, precise vocabulary

- For future work, fit multiple curves (e.g. sawtooth pattern) to each game
- Can help infer separate curves for segments with different difficulty
- Use continuous PvL outcomes rather than binary win/loss

Contact

Anurag Sarkar
Northeastern University
sarkar.an@husky.neu.edu

Acknowledgments

This work was supported by the **National Science Foundation** under grant nos. 1652537 and 1629879 and also 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