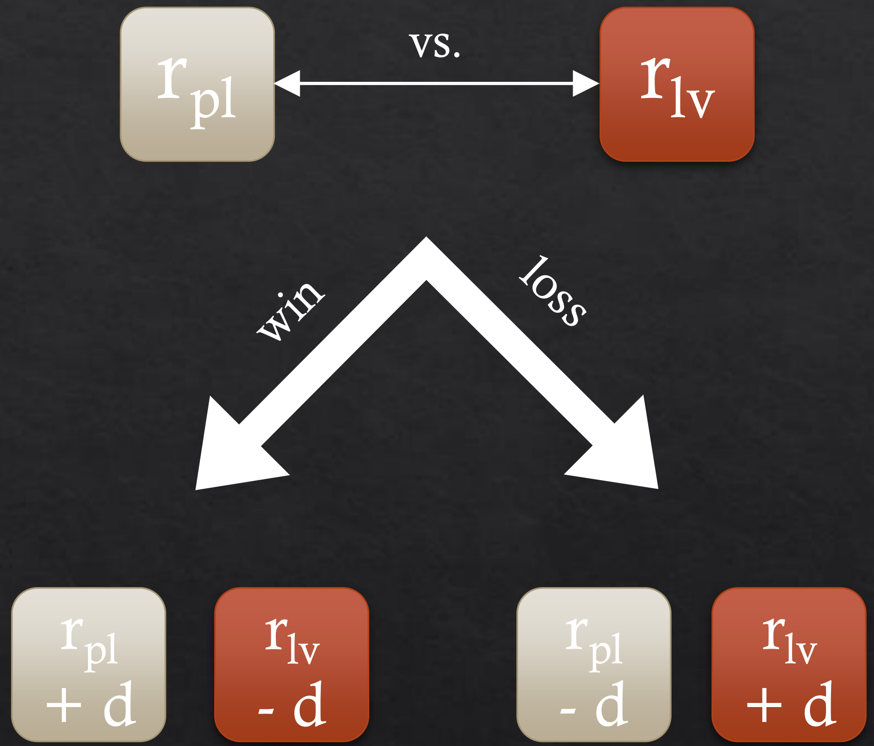
Using Rating Arrays to Estimate Score Distributions for Player-versus-Level Matchmaking

Anurag Sarkar and **Seth Cooper**

Northeastern University

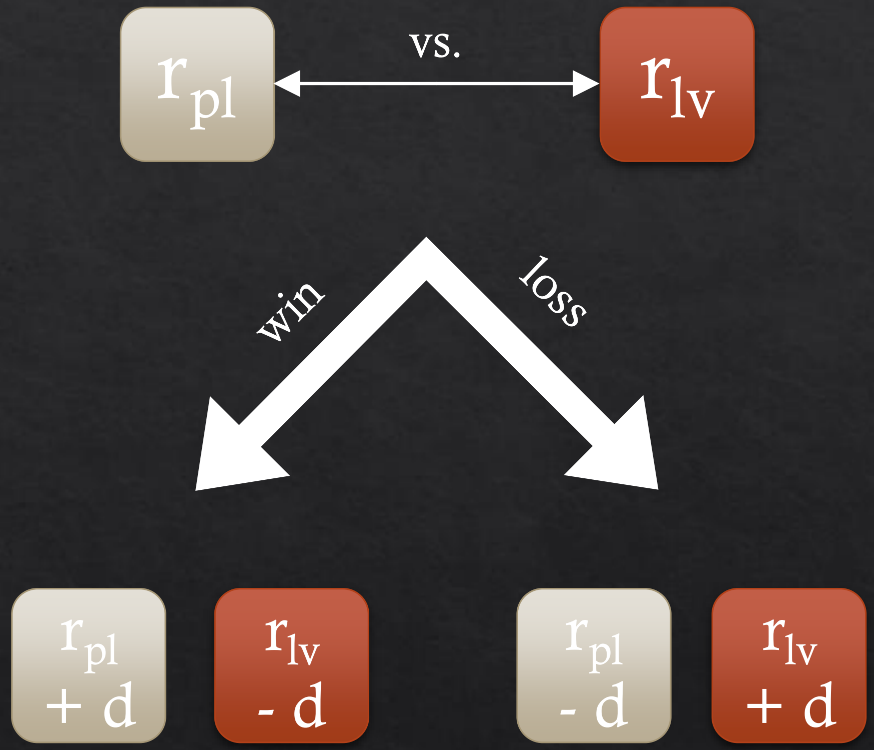
Player Rating Systems

- Assign skill-based ratings to players
- Produce fair matches by pairing players of similar skill
- Score prediction
- E.g. Elo, Glicko, Glicko-2, Microsoft TrueSkill



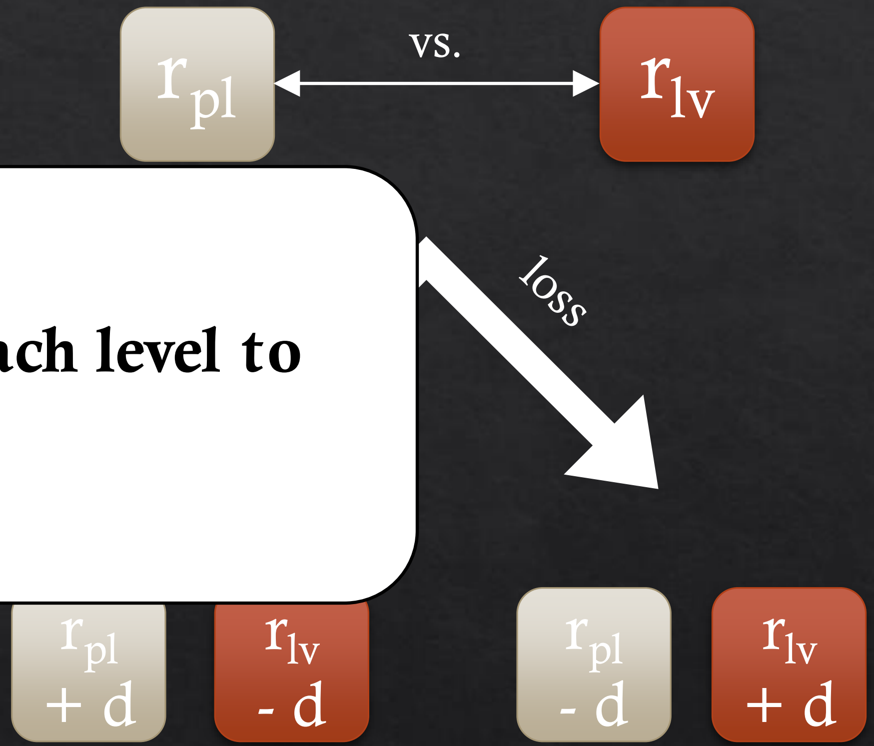
PvL Matchmaking

- Applied in the PvL domain for difficulty balancing
- 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 if player wins or loses vs. level



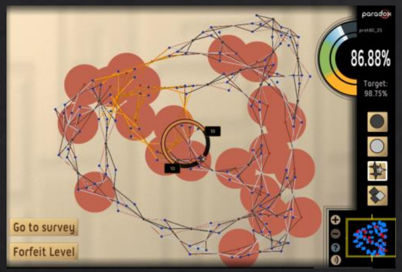
PvL Matchmaking

- Applied in the PvL domain for difficulty balancing
- Each player and level assigned Glicko-2 ratings (init=1500)
  - Player ratin
  - Level rating
- Compare rati
  i.e. level diffic
- Ratings updated based on if player wins or loses vs. level



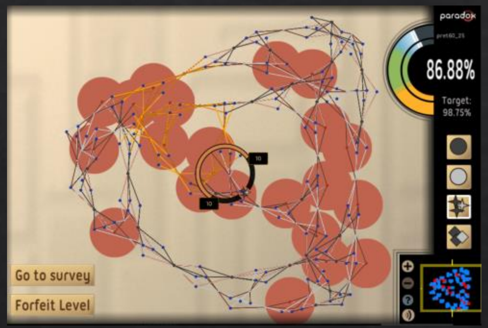
Requires fixing target score cutoff for each level to determine win/loss

Rating Arrays



1700

Single Level Rating

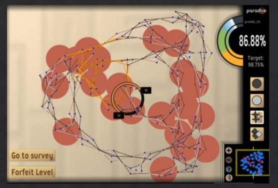


0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

Array of Level Ratings

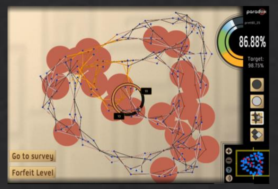
Rating Arrays

Single Level Rating



→ 1700

Array of Level Ratings

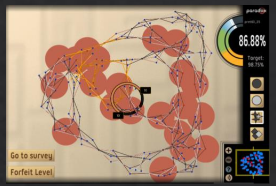


$$\left\{ \begin{array}{lcr} 0\% & \rightarrow & 305 \\ 10\% & \rightarrow & 929 \\ & \ldots & \\ 80\% & \rightarrow & 1860 \\ 90\% & \rightarrow & 2071 \end{array} \right\}$$

Single Rating	**Rating Array**
Matchmaking between players and levels	Matchmaking between players and (level, threshold) pairs

Rating Arrays

Single Level Rating



→ 1700

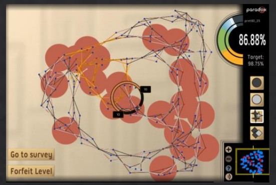
Array of Level Ratings

$$\left\{ \begin{array}{lcr} 0\% & \rightarrow & 305 \\ 10\% & \rightarrow & 929 \\ & \dots & \\ 80\% & \rightarrow & 1860 \\ 90\% & \rightarrow & 2071 \end{array} \right\}$$

Single Rating	**Rating Array**
Matchmaking between players and levels	Matchmaking between players and (level, threshold) pairs
Fixed thresholds for all players	Dynamic thresholds based on player skill

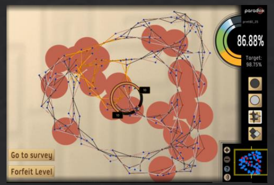
Rating Arrays

Single Level Rating



→ 1700

Array of Level Ratings

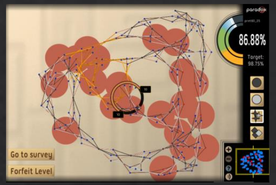


0%	→	305
10%	→	929
	…	
80%	→	1860
90%	→	2071

Single Rating	Rating Array
Matchmaking between players and levels	Matchmaking between players and (level, threshold) pairs
Fixed thresholds for all players	Dynamic thresholds based on player skill
Difficulty of completing a level	Difficulty of achieving specific scores on levels i.e. various stages of completion

Rating Arrays

Single Level Rating



→ 1700

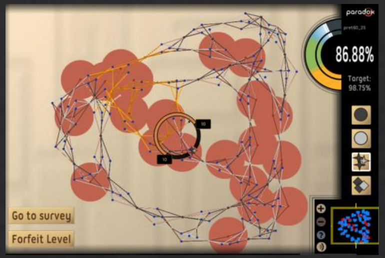
Array of Level Ratings

$$\left\{ \begin{array}{lcr} 0\% & \rightarrow & 305 \\ 10\% & \rightarrow & 929 \\ & \dots & \\ 80\% & \rightarrow & 1860 \\ 90\% & \rightarrow & 2071 \end{array} \right\}$$

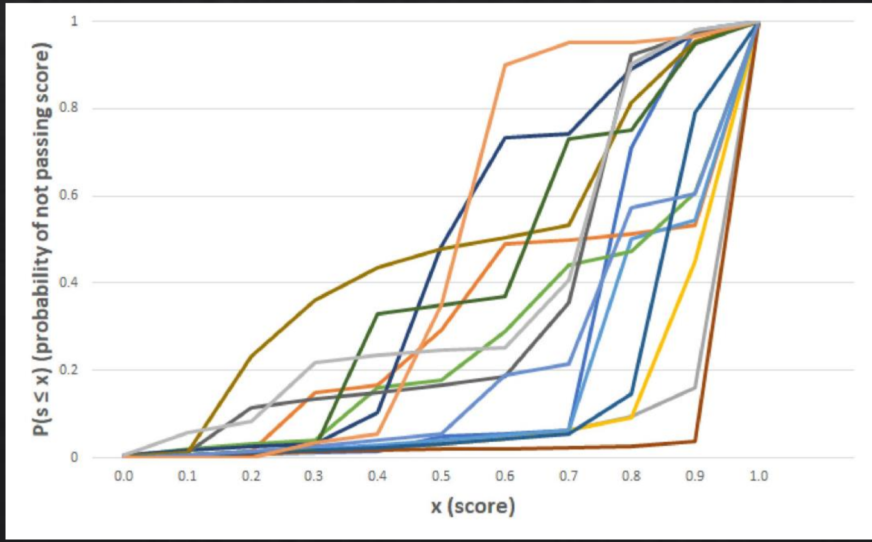
Single Rating	**Rating Array**
Matchmaking between players and levels	Matchmaking between players and (level, threshold) pairs
Fixed thresholds for all players	Dynamic thresholds based on player skill
Difficulty of completing a level	Difficulty of achieving specific scores on levels i.e. various stages of completion
Predict single scores or win/loss	Predict probability that player will achieve a certain score

Rating Arrays

- Enables modeling a CDF over possible scores

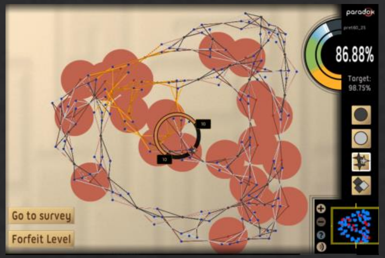


0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

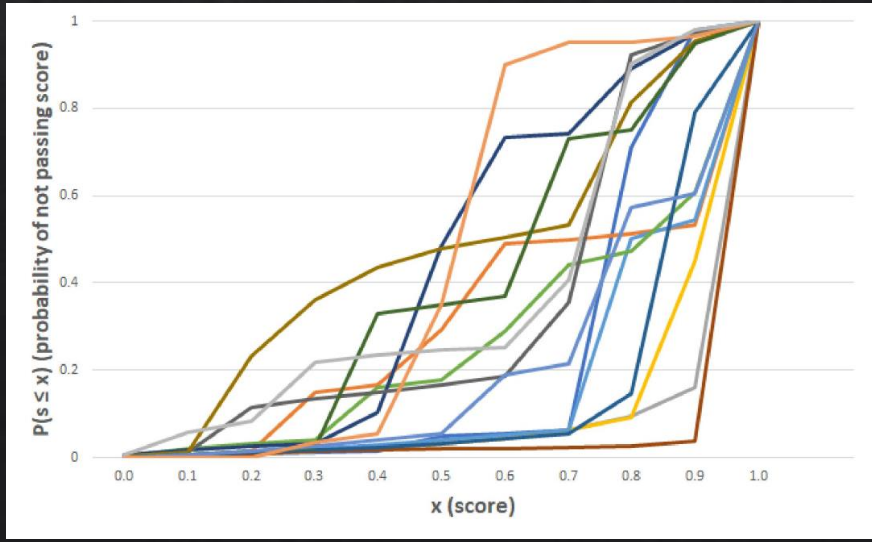


Rating Arrays

- Enables modeling a CDF over possible scores
- Allows predicting likelihood of player achieving new high score

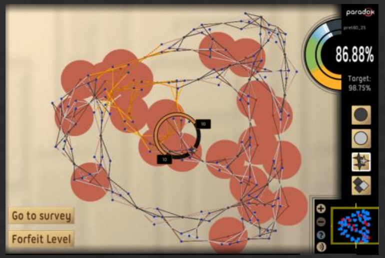


0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

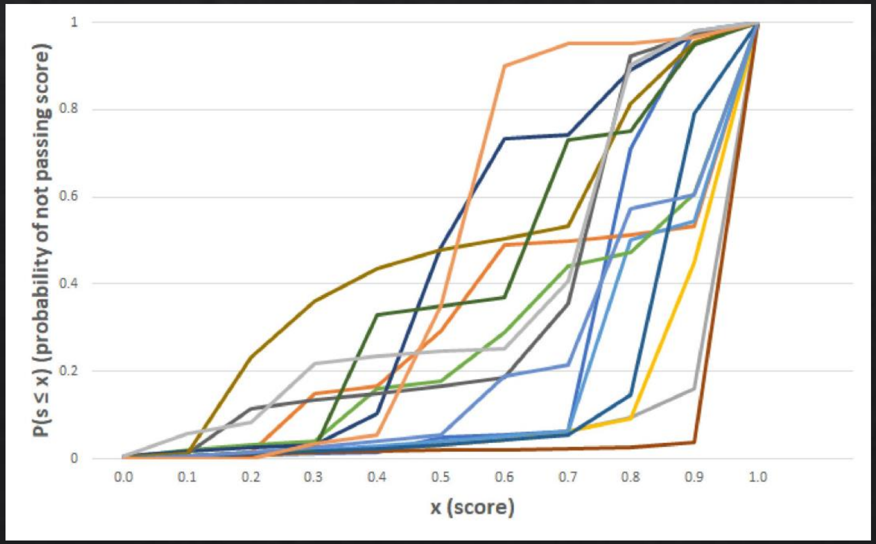


Rating Arrays

- Enables modeling a CDF over possible scores
- Allows predicting likelihood of player achieving new high score
- Useful in human computation games (HCGs) where high scores are new/better solutions

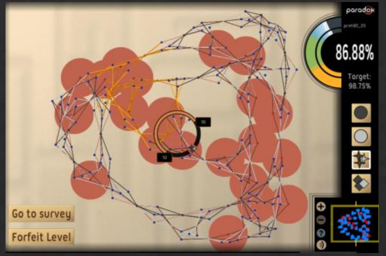


0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

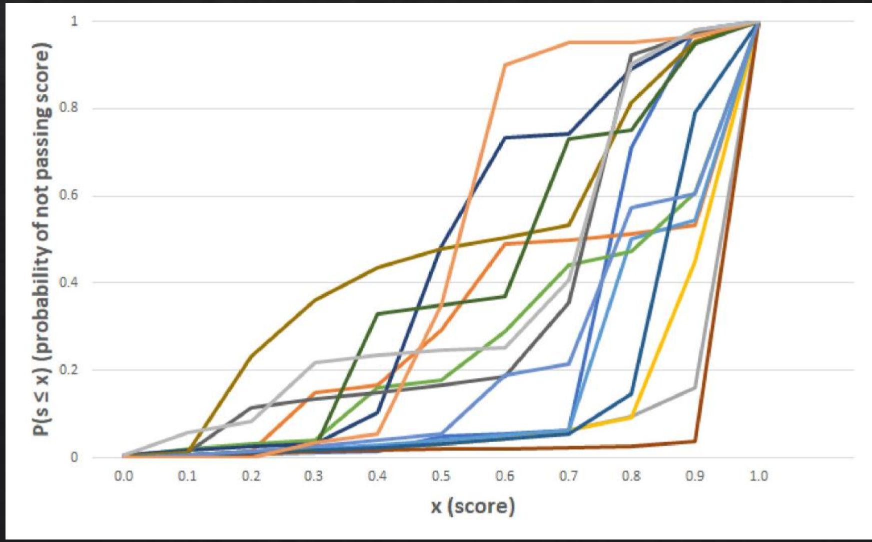


Rating Arrays

- Enables modeling a CDF over possible scores
- Allows predicting likelihood of player achieving new high score
- Useful in human computation games (HCGs) where high scores are new/better solutions
- Rating arrays + ratings-based matchmaking → identify players able to set new high scores while also performing DDA



0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071



Method: Initialization

- Glicko-2 rating system

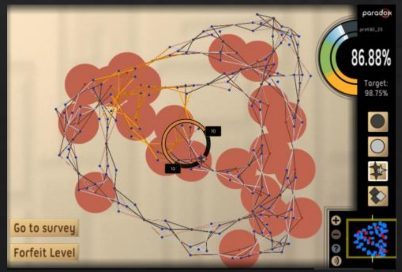
- Each player has a single rating (init=1500)
- Each level has an array of n ratings (n=10)

→ 1500

- Array indices represent thresholds (0% to 90%)
- Array values represent corresponding ratings

- Initialized rating array centered around 1500 using a smoothly increasing curve given by:

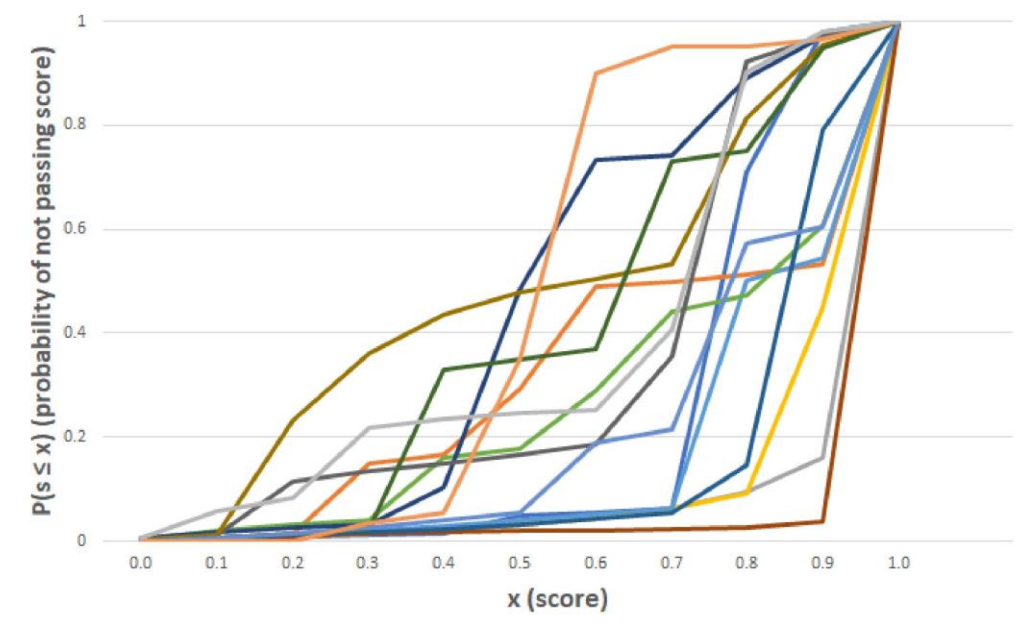
$$1500 - 260 \ln\left(\frac{1 - \text{threshold}}{\text{threshold}}\right)$$



0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

Method: CDF Computation

- For a PvL pairing, score CDF maps score to probability that player will not score higher on that level



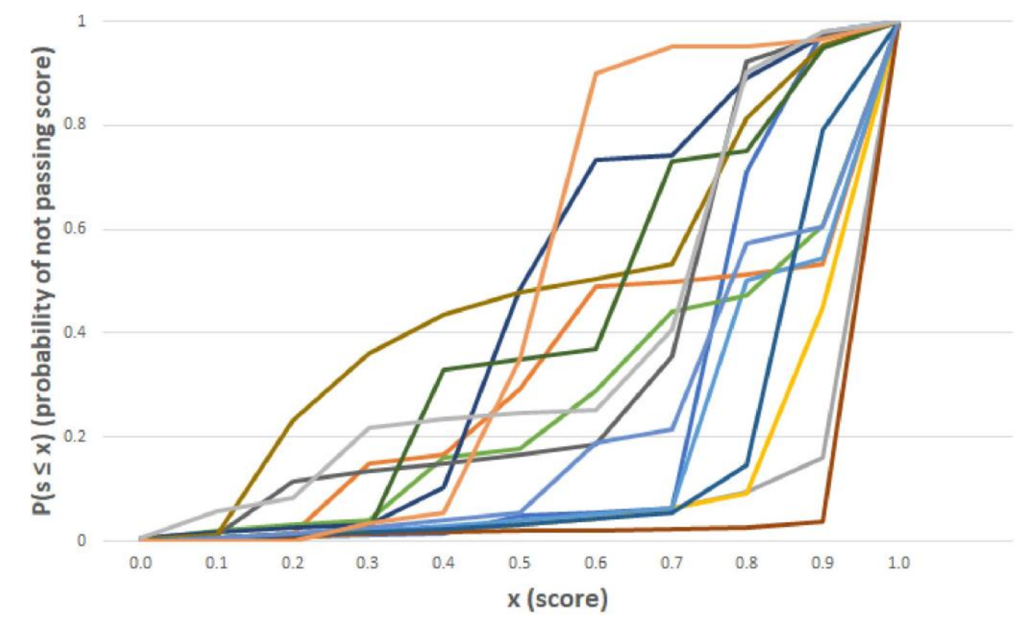
Example Player CDF

Method: CDF Computation

- For a PvL pairing, score CDF maps score to probability that player will not score higher on that level

- For a given player and threshold x, CDF of their score s on a level:

$$F_s(x) = P(s \leq x); P(s \leq 100) = 1$$



Example Player CDF

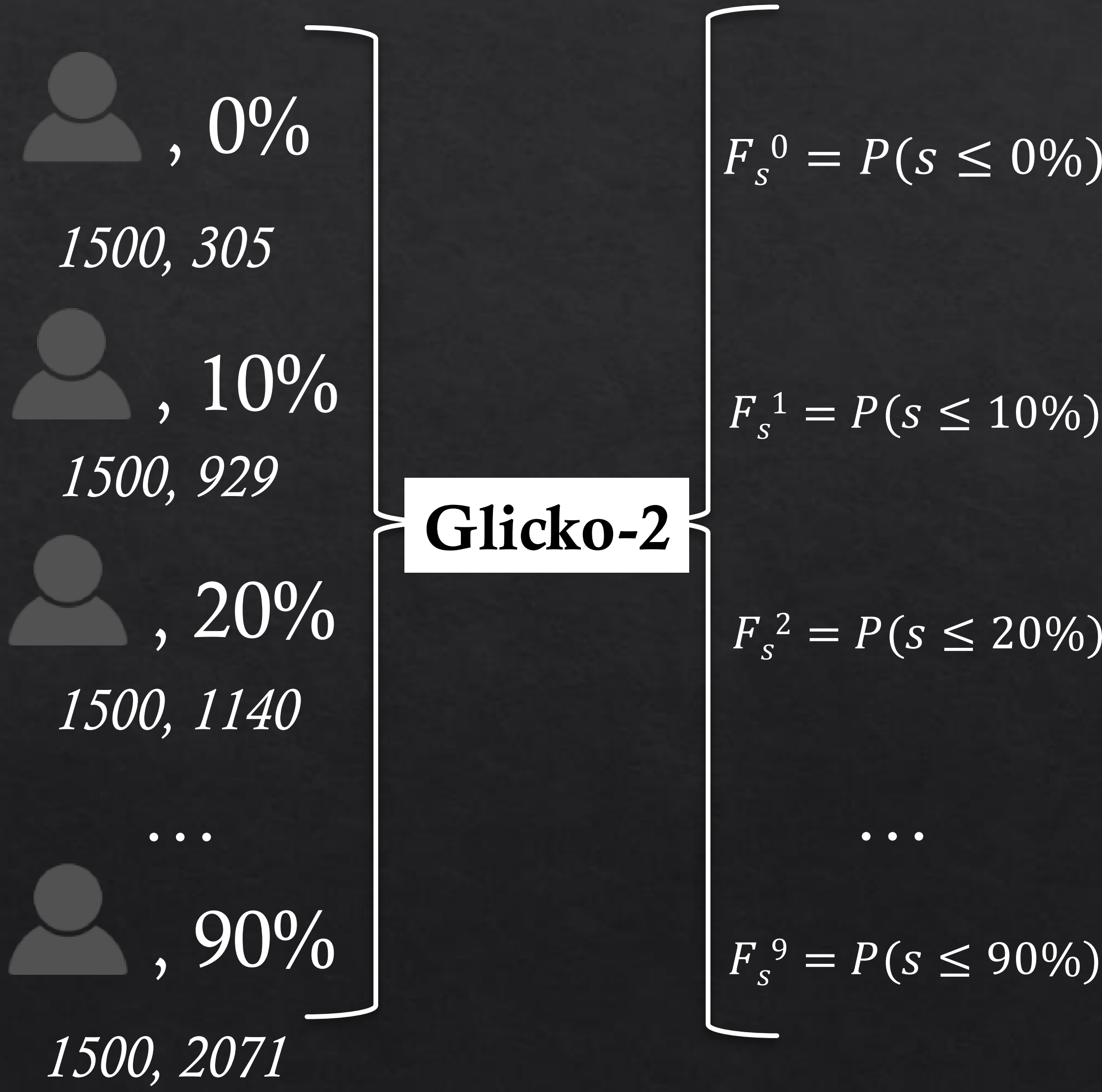
Method: CDF Computation

- For a PvL pairing, score CDF maps score to probability that player will not score higher on that level

- For a given player and threshold x, CDF of their score s on a level:

$$F_s(x) = P(s \leq x); P(s \leq 100) = 1$$

- Given player rating and level rating array, construct CDF with probabilities that player will not pass a threshold τ^t: $F_s{}^t = P(s \leq \tau^t)$

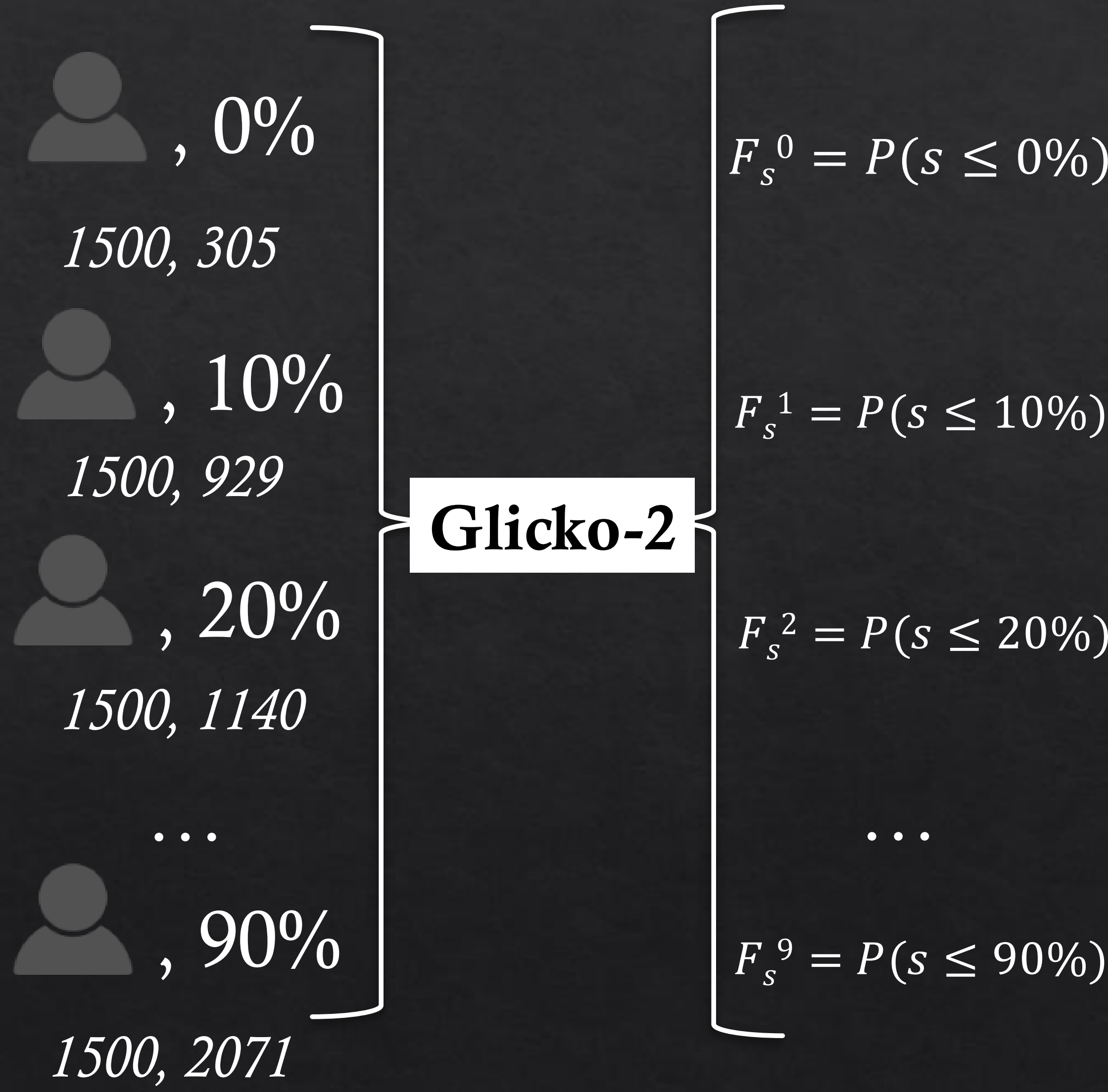


Method: CDF Computation

- For a PvL pairing, score CDF maps score to probability that player will not score higher on that level
- For a given player and threshold x, CDF of their score s on a level:

$$F_s(x) = P(s \leq x); P(s \leq 100) = 1$$

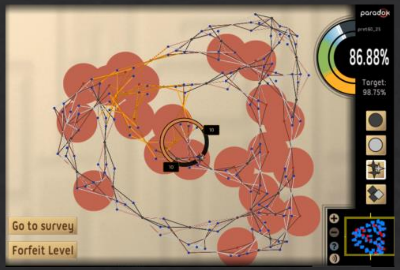
- Given player rating and level rating array, construct CDF with probabilities that player will not pass a threshold τ^t: $F_s{}^t = P(s \leq \tau^t)$
- Construct $F_s(x)$ by linear interpolation between the two thresholds surrounding x



Method: Rating Updates

- After each PvL match, update ratings using Glicko-2 as if player simultaneously played vs. all thresholds

vs.



- If player scores s
  - Loses against all thresholds $\tau^t > s$
  - Wins against all thresholds $\tau^t \leq s$

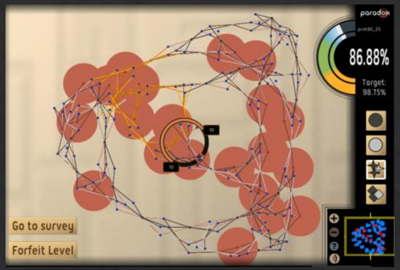
vs.

0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

Method: Rating Updates

- After each PvL match, update ratings using Glicko-2 as if player simultaneously played vs. all thresholds

vs.



- If player scores s
  - Loses against all thresholds $\tau^t > s$
  - Wins against all thresholds $\tau^t \leq s$

vs.

0%	→	305
10%	→	929
20%	→	1140
30%	→	1280
40%	→	1395
50%	→	1500
60%	→	1605
70%	→	1720
80%	→	1860
90%	→	2071

- Updates could lead to non-strictly increasing threshold ratings

- Post-processing:
  - If rating for τ^t >= rating for τ^{t+1} → set rating for τ^t = (rating for τ^{t+1}) – 1
  - If rating for τ^{t+1} < rating for τ^t → set rating for τ^{t+1} = (rating for τ^t) + 1

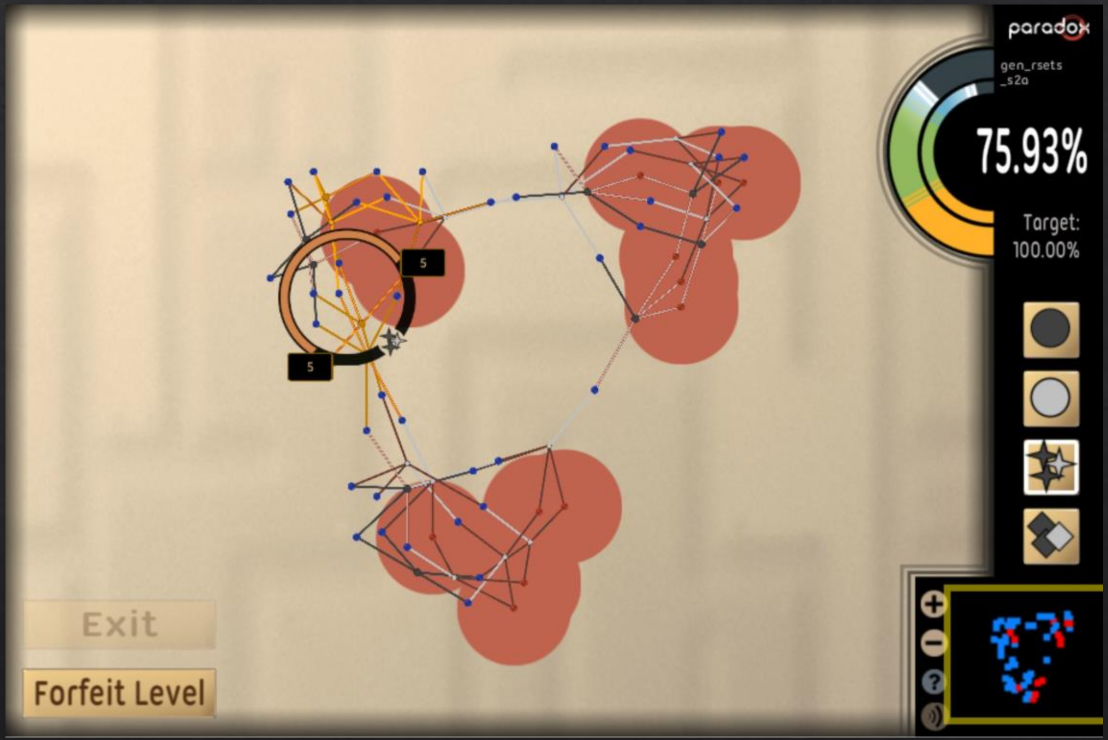
Datasets

- *Paradox*
- Synthetic data using Elo ratings
- Match data with instances of players playing levels treated as PvL matches
- Each entry consists of
  - Timestamp
  - Player and Level IDs
  - Player and Level Scores
  - Result

PlayerID	LevelName	Time	LevelStart	LevelMax	PlayerCur	PlayerMax	Result
p1	gen_tree_ma	1544722425148	84	107	107	107	win
p2	pret60_25	1544722434565	139	160	157	157	loss
p2	medium	1544722465193	735	953	903	903	loss
p3	par8-3-c	1544722465649	264	298	291	291	loss
p4	flat50-1	1544722472911	417	545	509	518	loss
p5	dubois21	1544722490918	149	168	165	165	loss
p2	hole6	1544722500092	70	133	132	132	loss
p2	gen_tree_la	1544722516825	198	242	216	216	loss
p5	gen_rsets_s1a	1544722539585	40	54	54	54	win
p4	ii8a1	1544722545307	151	186	183	184	loss
p2	gen_rsets_s2a	1544722545492	36	54	51	51	loss

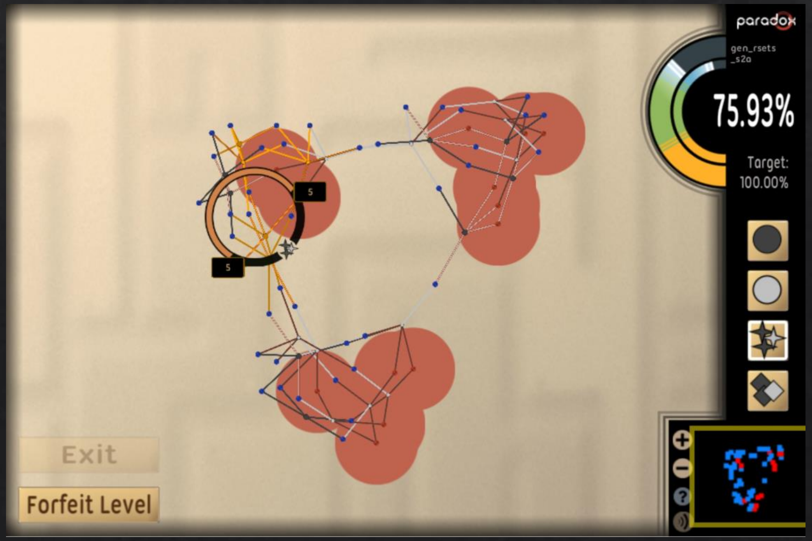
Paradox

- 2D human computation puzzle game
- Each level is a boolean constraint satisfaction problem
- Players assign values to variables to solve constraints
- Score: percentage of satisfied constraints
- Target score reached → *Level Completed*



Paradox

- 100 players recruited using Amazon Mechanical Turk, final data set had 98 players and 691 matches
  - 9 tutorial levels (static order)
  - 50 challenge levels (random order)
  - Players had to play at least 5 challenge levels



Synthetic Elo Data

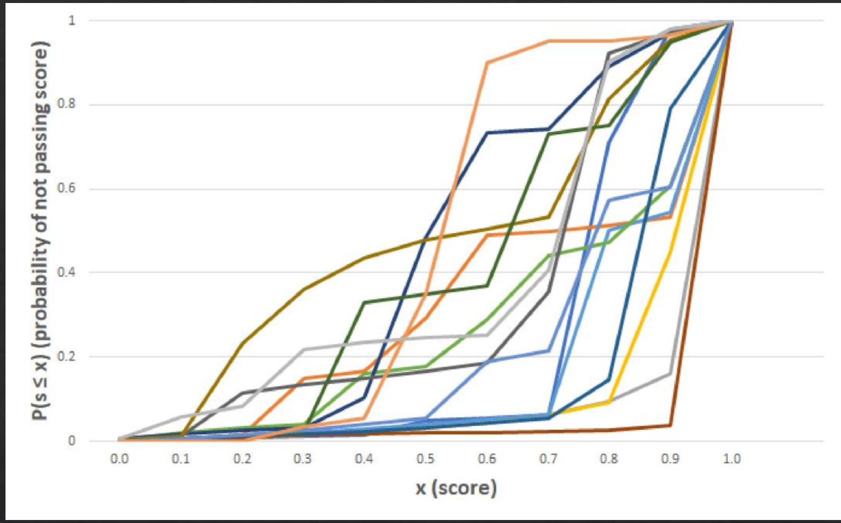
- 100 generated players and 50 generated levels with uniformly random ratings (900 – 2100)
- Simulated 1000 matches by randomly selecting a player and a level
- Player score vs. a level was the Elo expected score based on both ratings

Evaluations

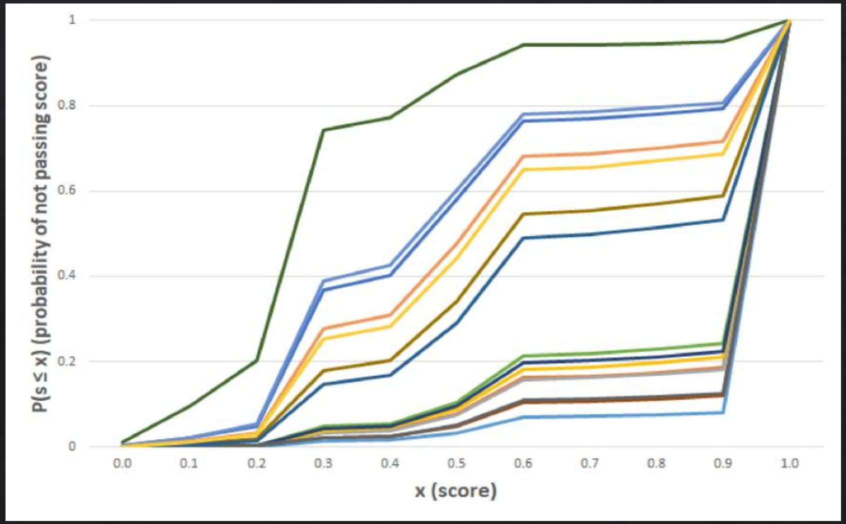
- Accuracy of the CDF in predicting probabilities of events
- Accuracy of the CDF in predicting player scores
- Using the CDF to serve players with levels for setting new high scores

Evaluations

- To evaluate both data sets, performed ratings playback to update ratings for players and level arrays
- Rating updates and CDF computations using matches up to current point of playback (training data)
- Predictions made on all future matches (test data)
- Example player and level score CDFs



Example Player CDF



Example Level CDF

CDF Accuracy

- Count how often scores predicted to happen between 0-10%, 10-20% … 90-100% of the time, actually happened within that range

CDF Accuracy

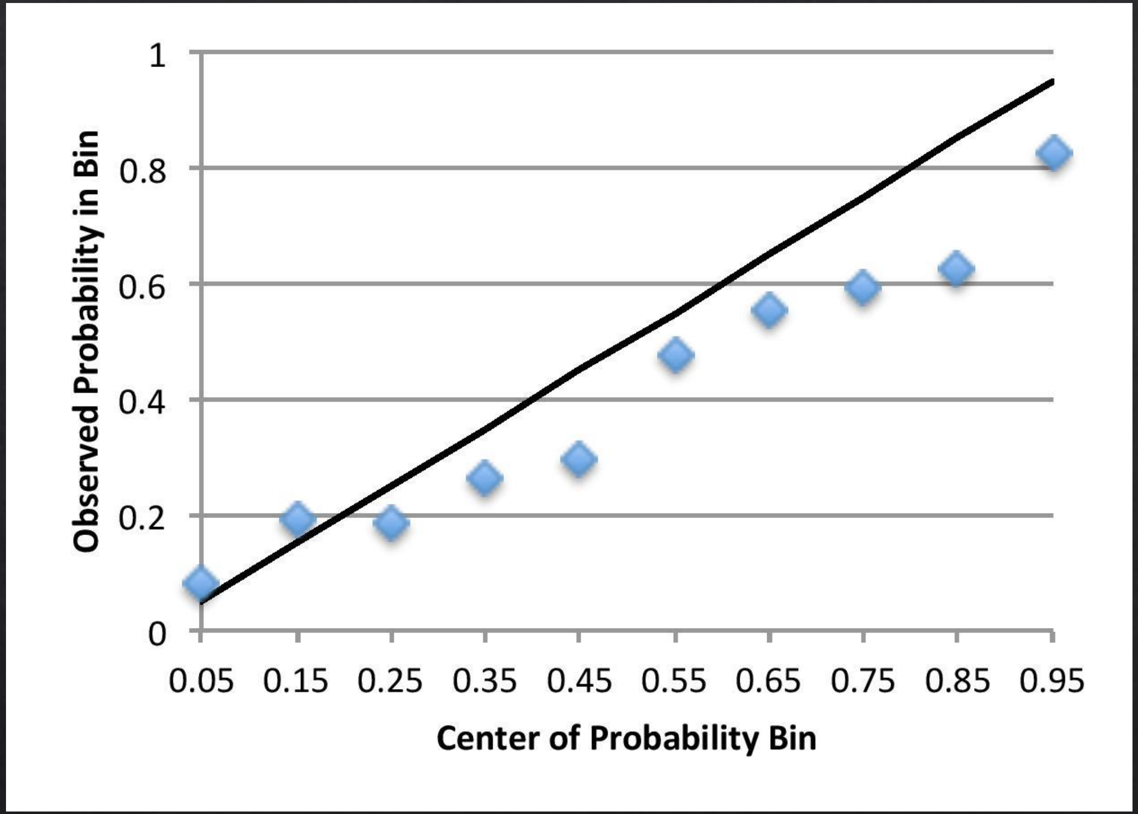
- Count how often scores predicted to happen between 0-10%, 10-20% … 90-100% of the time, actually happened within that range

- For each match, used CDF to compute probability of score falling in various ranges

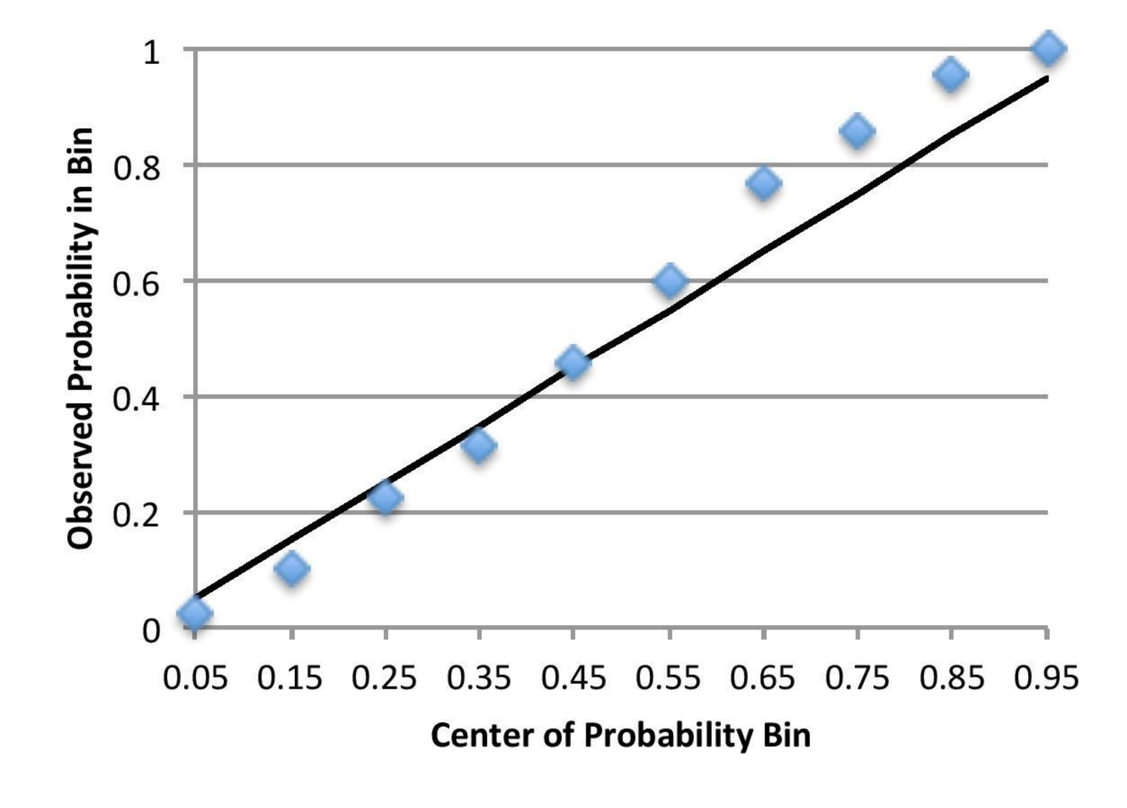
CDF Accuracy

- Count how often scores predicted to happen between 0-10%, 10-20% … 90-100% of the time, actually happened within that range
- For each match, used CDF to compute probability of score falling in various ranges
- Compared center of predicted probabilities in each bin with observed probabilities in that bin

CDF Accuracy



Paradox ($\rho = 0.980$, $p < 0.001$)



Synthetic ($\rho = 0.995$, $p < 0.001$)

Score Estimation

- Accuracy of player scores predicted using CDF compared to using a single Glicko-2 rating

Score Estimation

- Accuracy of player scores predicted using CDF compared to using a single Glicko-2 rating
- For both data sets
  - RMSD of actual player score vs expected score predicted by CDF (Err_{cdf})
  - RMSD of actual player score vs expected score predicted by Glicko-2 (Err_{gl2})
  - RMSD of CDF and Glicko-2 predictions ($Diff_{cdf\text{-}gl2}$)

Score Estimation

- Accuracy of player scores predicted using CDF compared to using a single Glicko-2 rating
- For both data sets
  - RMSD of actual player score vs expected score predicted by CDF (Err_{cdf})
  - RMSD of actual player score vs expected score predicted by Glicko-2 (Err_{gl2})
  - RMSD of CDF and Glicko-2 predictions ($Diff_{cdf\text{-}gl2}$)
- $E(s) = \int_0^1 \left(1 - Fs(x)\right) dx$

Score Estimation

- Accuracy of player scores predicted using CDF compared to using a single Glicko-2 rating
- For both data sets
  - RMSD of actual player score vs expected score predicted by CDF (Err_{cdf})
  - RMSD of actual player score vs expected score predicted by Glicko-2 (Err_{gl2})
  - RMSD of CDF and Glicko-2 predictions ($Diff_{cdf\text{-}gl2}$)
- $E(s) = \int_0^1 \left(1 - Fs(x)\right) dx$

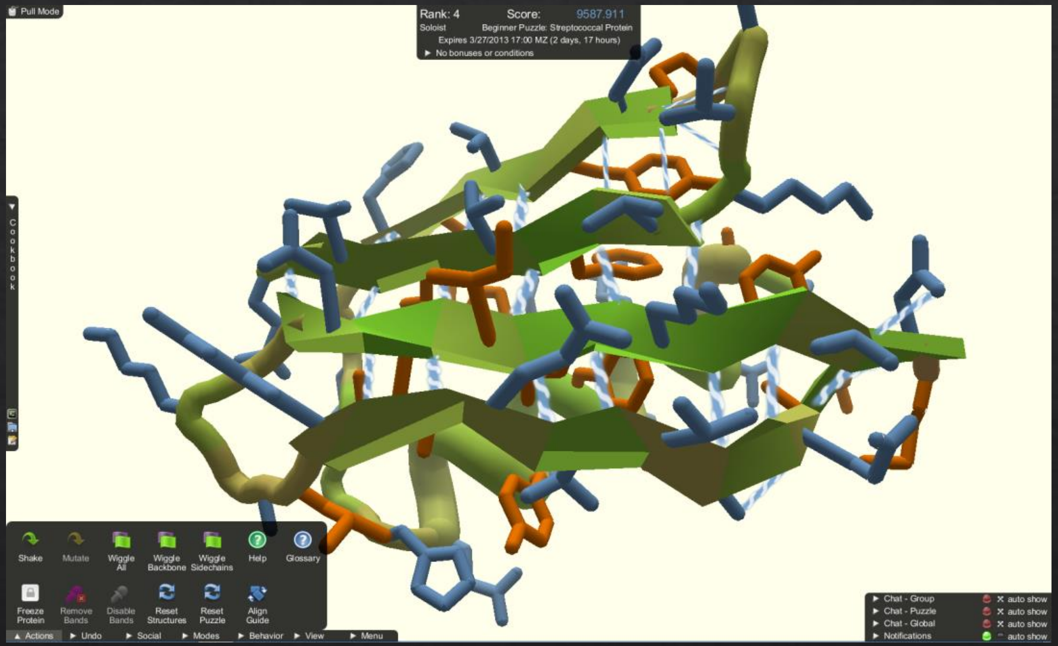
	$\mathbf{Err_{cdf}}$	$\mathbf{Err_{gl2}}$	$\mathbf{Diff_{cdf\text{-}gl2}}$
Paradox	**0.407**	**0.401**	**0.058**
Elo	**0.115**	**0.126**	**0.066**

High Scores

- Serve levels with aim of setting high scores while performing dynamic difficulty adjustment (DDA)

High Scores

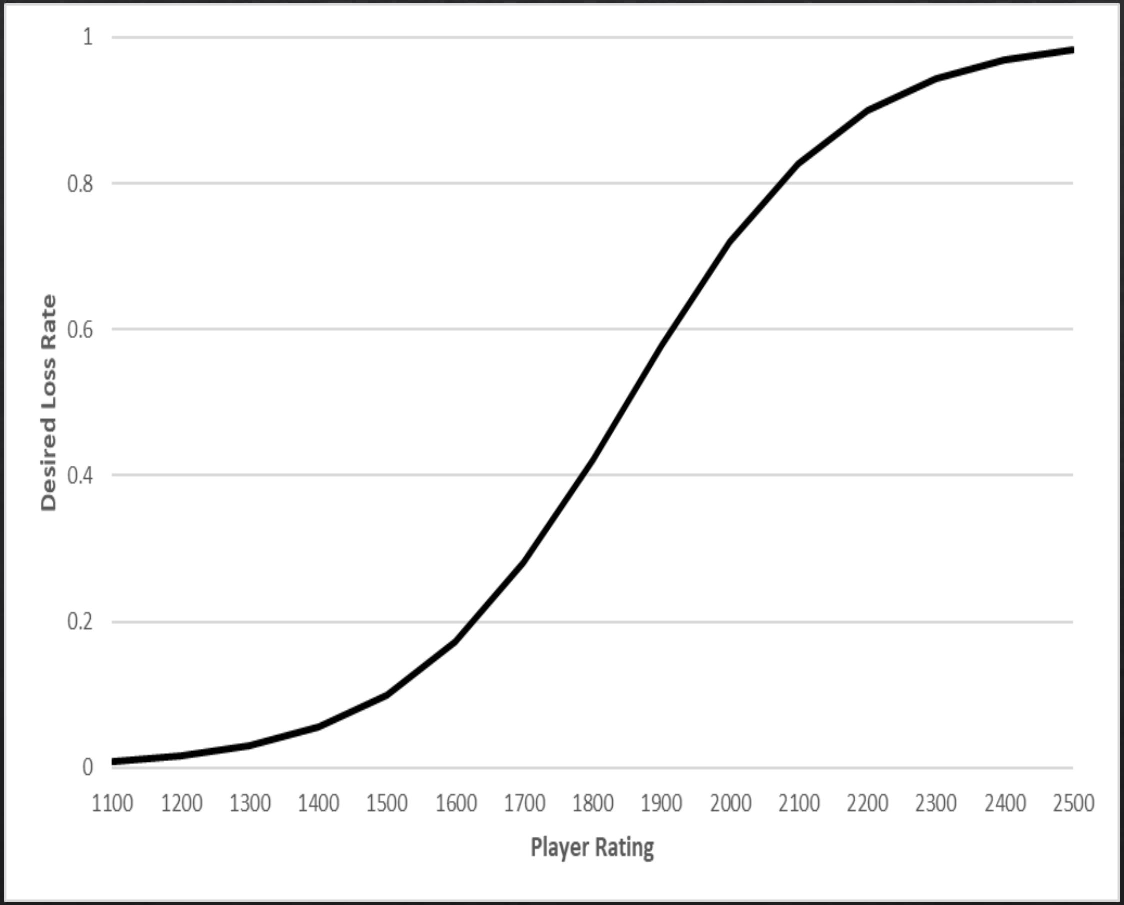
- Serve levels with aim of setting high scores while performing dynamic difficulty adjustment (DDA)
- Useful in HCGs → high scores may imply new/better solutions



Foldit

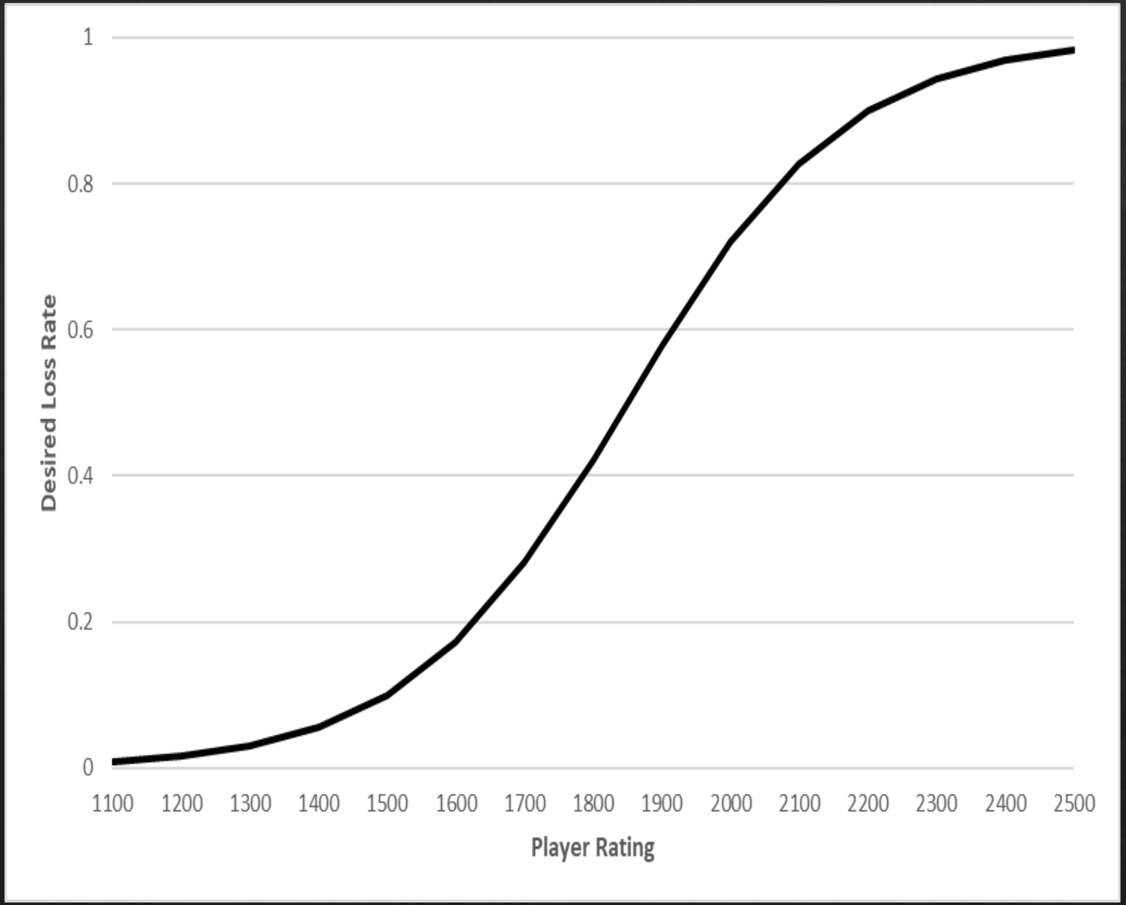
High Scores

- Serve levels with aim of setting high scores while performing dynamic difficulty adjustment (DDA)
- Useful in HCGs → high scores may imply new/better solutions
- Previously DDA in *Paradox* done using player's desired loss rate $DLR = \frac{1}{1+e^{\alpha(\beta - x)}}$



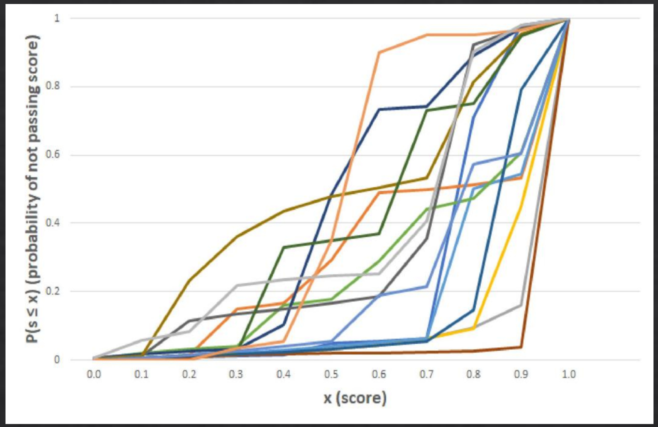
High Scores

- Serve levels with aim of setting high scores while performing dynamic difficulty adjustment (DDA)
- Useful in HCGs → high scores may imply new/better solutions
- Previously DDA in *Paradox* done using player's desired loss rate $DLR = \frac{1}{1+\mathrm{e}^{\alpha(\beta-x)}}$
- Computed using player's Glicko-2 rating
  - DLR goes up as rating goes up
  - Player is matched with harder levels

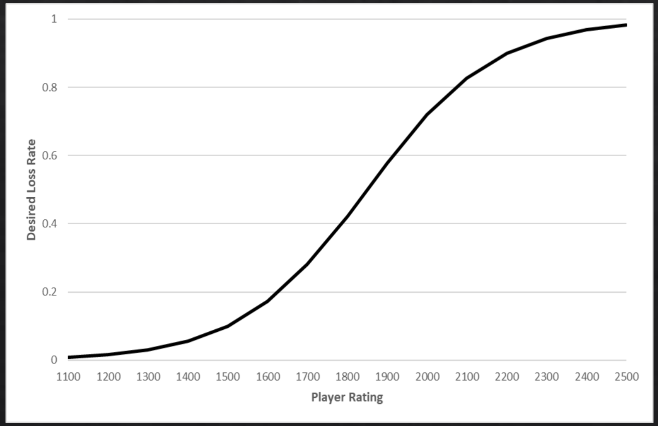


High Scores

- s_{exp} → expected score predicted by the CDF
- s_{dlr} → DLR score
- s_{max} → max score seen on a level



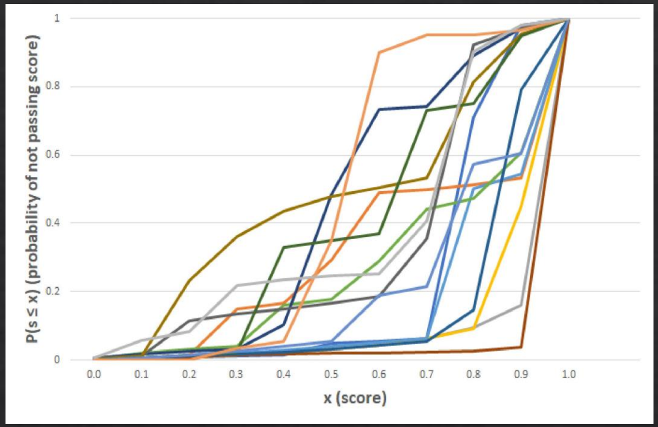
→ s_{exp}



→ s_{dlr}

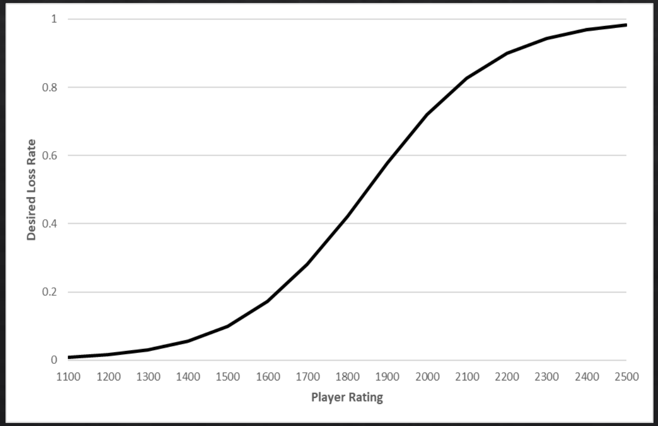
High Scores

- s_{exp} → expected score predicted by the CDF
- s_{dlr} → DLR score
- s_{max} → max score seen on a level



→ s_{exp}

- Two approaches to selecting level to serve player
  - If $s_{exp} > s_{max}$ → looking only for increased high scores
  - If both s_{exp} *and* $s_{dlr} > s_{max}$ → looking for increased high scores while doing DDA



→ s_{dlr}

High Scores

- Trade-off between increased accuracy using only $\mathbf{s}_{\mathbf{exp}}$ and ability to perform DDA using $\mathbf{s}_{\mathbf{dlr}}$

High Scores

- Trade-off between increased accuracy using only $\mathbf{S_{exp}}$ and ability to perform DDA using $\mathbf{S_{dlr}}$
- Only $\mathbf{S_{exp}}$ → ignores desired difficulty curve when serving levels
- Only $\mathbf{S_{dlr}}$ → ignores player’s ability to set new high scores

High Scores

- Trade-off between increased accuracy using only $\mathbf{s_{exp}}$ and ability to perform DDA using $\mathbf{s_{dlr}}$

- Only $\mathbf{s_{exp}}$ → ignores desired difficulty curve when serving levels
- Only $\mathbf{s_{dlr}}$ → ignores player's ability to set new high scores

- Combining both → serving levels where players can improve high scores while also doing DDA

Conclusion and Future Work

- Introduced level rating arrays for improved PvL score prediction and matchmaking
- Enables deriving score CDFs for both players and levels
- Helps decide if a level should be served to a player to try to set a new high score + DDA

Conclusion and Future Work

- Introduced level rating arrays for improved PvL score prediction and matchmaking
- Enables deriving score CDFs for both players and levels
- Helps decide if a level should be served to a player to try to set a new high score + DDA

- For future work, use rating arrays to do live matchmaking
- Improve prediction metrics by considering other difficulty curves
- Investigate application of rating arrays in educational games

Conclusion and Future Work

- Introduced level rating arrays for improved PvL score prediction and matchmaking
- Enables deriving score CDFs for both players and levels
- Helps decide if a level should be served to a player to try to set a new high score + DDA

- For future work, use rating arrays to do live matchmaking
- Improve prediction metrics by considering other difficulty curves
- Investigate application of rating arrays in educational games

Contact

Anurag Sarkar

Northeastern University

sarkar.an@husky.neu.edu

Acknowledgments

This work was supported by the **National Science Foundation** under grant no. 1652537