# An Online System for Player-vs-Level Matchmaking in Human Computation Games

Anurag Sarkar
*Northeastern University*
Boston, MA, USA
sarkar.an@northeastern.edu

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

***Abstract*—We present a fully online system for skill and ratings-based dynamic difficulty adjustment (DDA) in human computation games (HCGs). Prior work used an initial offline phase for collecting level ratings to avoid cold-start when using the system online. Further, such systems set a fixed target score win/lose threshold for each level. In this work, we address these issues by 1) using an $\epsilon$-greedy variant of the ratings and skill-based DDA algorithm and 2) applying rating arrays, proposed in past work but not yet used in an online setting. We demonstrate our system in two online matchmaking experiments using two HCGs. Results suggest that the $\epsilon$-greedy modification helps address the cold-start and that using rating arrays leads to players completing more levels.**

***Index Terms*—dynamic difficulty adjustment, human computation games, rating systems, skill chains, matchmaking, $\epsilon$-greedy**

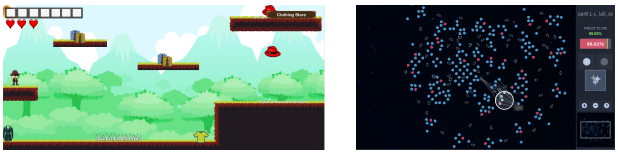
## I. Introduction

Human computation games (HCGs) are games whose levels model real-world problems in order to help solve them by leveraging the collective abilities of players. Previously, rating systems such as Glicko-2 [1], typically used for player-vs-player (PvP) matchmaking, have been used for dynamic difficulty adjustment (DDA) in HCGs. By framing DDA as a player-vs-level (PvL) matchmaking problem, and assigning ratings to players and levels based on abilities and difficulty respectively, such systems can match players with levels of appropriate difficulty and improve player engagement [2], [3]. These systems have also incorporated skill chains to match players with levels that require specific skills [4], [5].

However, though effective, such systems have been evaluated in situations where the ratings for levels are gathered and updated in a separate offline phase via playthroughs and then held fixed during the actual matchmaking phase, as failing to do so causes the system to suffer from a cold-start problem where level ratings do not accurately capture the difficulty of a level. More specifically, if ratings are updated online, harder levels may primarily be played by more advanced players and thus, these levels can end up with a low rating. Additionally, in previous work, these systems were tested with a fixed target score win/lose threshold for each level, kept constant across players; this did not allow more fine-grained DDA via setting target scores dynamically per match.

This material is based upon work supported by the National Science Foundation under Grant No. 1652537.





Iowa James Paradox

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

In this paper, we present a fully online matchmaking system for DDA in HCGs that extends prior work. First, we use an $\epsilon$-greedy strategy to address cold-start by allowing players to be assigned, with low probability, random levels rather than matched levels; this helps even out playthroughs so that levels are more likely to be played by both advanced and novice players. We tested this approach using the puzzle HCG *Paradox* and the platformer HCG *Iowa James*, and found that it helps address the cold-start problem. Second, we incorporate rating arrays, introduced in prior work [6] but not yet tested in a live setting. Rather than assign a single rating, this approach assigns levels an array of ratings corresponding to different score thresholds, which can enable setting a level's target score dynamically by comparing a player's rating with the ratings of each threshold, potentially identifying players capable of setting new high scores. We tested rating arrays in *Paradox* and found they enable players to complete more levels. This work thus contributes a fully online version of the ratings and skill-based matchmaking system for DDA in HCGs, incorporating an $\epsilon$-greedy matchmaking strategy and rating arrays.

## II. Background

Dynamic difficulty adjustment (DDA) [7], [8] is a well-studied problem, generally intended to balance difficulty and improve engagement in games [9], and there exist several DDA techniques such as player modeling [10], machine learning [11], level design modification [12], and emotion self-reporting [13] among others. However, these primarily require modifying the content of the level to alter difficulty, making them unsuitable for use with human computation games (HCGs). Thus, skill chains and rating systems have been used for DDA in HCGs. Both do this by modifying the order in which levels are served, skill chains by defining the progression of skill acquisition within a game and rating systems by being repurposed to perform PvL (rather than PvP) matchmaking. Skill chains were defined by Cook [14]

as a means of modeling the order of skill acquisition in a game via a directed graph formulation with nodes representing skills and edges indicating relations between them and how complex skills are composed of simpler ones. In addition to DDA, skill chains have found use in refining the progression of educational games [15] and in analyzing game difficulty via the use of AI agents [16]. Similar to skill chains, rating systems perform DDA by modifying the order in which players encounter levels. Typically used to produce fair matches in PvP settings such as chess or MOBAs, rating systems have been repurposed to perform PvL matchmaking for DDA in HCGs in several prior works [2], [3]. This involves treating levels as players whose ratings indicate the difficulty of the level. By comparing ratings, the system can then assign players with levels that pose an appropriate amount of challenge based on the player's ability. More recently, skill chains and rating systems were combined into a unified DDA model [4], [5] where skill chains were used to define hierarchies of levels requiring similar sets of skills and rating systems used to then pick the best match for a player from the most appropriate hierarchy. Though effective, like prior DDA systems utilizing ratings, this too needed the level ratings to have been collected prior to online use. In this work, we attempt to enable such systems to be operational fully online without having to gather level ratings in a separate process. Our approach relies on an $\epsilon$-greedy strategy of serving random levels with low probability while using the matchmaking system.

## III. Games

For our experiments, we used two HCGs—*Iowa James* and *Paradox*—both games (shown in Figure 1) have been used in past works involving ratings and skill-based DDA systems.

### A. Iowa James

Iowa James is a 2D side-scrolling platformer similar to *Gwario* [17] where players have to collect items that belong to a certain category while avoiding items that do not. Levels feature static and/or moving hazards, relevant as well as irrelevant items and a locked treasure chest at the end. The player has to complete each level by unlocking the chest via collecting all relevant items. In each level, players have three lives. They lose a life by either collecting an incorrect item or making contact with a hazard. In this work, we used 3-item and 7-item versions of each of 14 different maps for a total of 28 levels. A player's score on a level was proportional to the number of lives remaining when completing it. The skill chain for the game comprised of skills related to typical platformer mechanics. *Navigating* is typical platformer jumping and moving actions. *Hazard-Static* involves jumping over stationary hazards, and requires *Navigating*. The remaining skills require *Hazard-Static*. These are: *Hazard-Moving*, jumping over moving hazards; *Hazard-Timed*, jumping or moving over hazards that rise and fall; and *Platforming*, jumping/moving across platforms.

### B. Paradox

*Paradox* is a 2D puzzle game where levels model boolean maximum satisfiability problems, represented as graph-like structures. Players satisfy constraints by assigning values to variables using one of three brush-like tools, which also correspond to skills. Two of these—*White* and *Black*—are used to assign true and false values to a single variable at a time, while a third advanced tool, called *Star*, requires both *White* and *Black* and assigns values to a group of variables by running an automated solver. A fourth *Challenge* skill depends on *Star* and was added to the game's skill chain in order to differentiate non-tutorial challenge levels from tutorial levels. Players get a score between 0 and 1 based on the percentage of constraints they can satisfy. For this work, we used 5 tutorial and 19 challenge levels for a total of 24 levels.

## IV. Ratings and Skill-based Matchmaking

The DDA model using rating systems and skill chains for PvL matchmaking was first defined in [4] and then extended in [5] where a detailed description of its working can be found. The model involves three stages: 1) defining the skill chain of the game being used, 2) annotating the levels of the game with the skills required to complete them and assigning them ratings based on their difficulty and 3) performing PvL matchmaking using the skill annotations and assigned ratings and updating player ratings and skills after each match.

In prior work, level ratings were based on an initial phase that gathers a large number of playthroughs of levels served randomly, with the level ratings then kept fixed during the following DDA phase. Levels were also assigned a single target score, limiting the matchmaking possibilities. We address these limitations in a full online system in two evaluations.

## V. Evaluation: $\epsilon$-greedy

Briefly, the set of levels eligible to be served is based on the player's current skills and those required by the levels. Only levels requiring one or fewer additional skills than what the player currently has are eligible. If such a level is not found, the system looks for those requiring two or fewer skills and so on. From among the eligible levels, the best match is determined based on the player and level ratings. More details can be found in [5]. In a fully online system, where all levels start with the default rating and are updated after matches, we could run into a cold-start problem—initial default ratings are inaccurate, and harder levels (i.e. those requiring more skills) may only be involved in matches with advanced players, thus skewing their ratings. Prior work [18] explored this issue for the use case of extracting difficulty curves from gameplay data by using *phantom matches*—simulated matches between players and levels to even out the skewed distribution—but we found these to not be useful in an online system, as player ratings are not stable and simulating matches was not reliable.

In this work, we used a simple $\epsilon$-greedy based strategy: with low probability, we served levels at random rather than as directed by the matchmaking system. Through this, we hoped to even out the distribution of players encountered by each level and correct for the skew as described earlier. Since the matchmaking system works in a two-step process, namely, determining eligible levels using the skill chain and then using the rating system to determine which of the eligible levels to

**Algorithm 1** $\epsilon$-greedy matchmaking

**Input:** $all\_levels$
**Output:** $level$
$level$ = $\varnothing$
$candidates$ = { levels from $all\_levels$ player hasn't completed or just played in the previous match }
**if** $random < \epsilon_1$ **then**
  $level \leftarrow$ random choice from $candidates$
**else**
  $candidates \leftarrow$ remove ineligible levels based on player's current skills from $candidates$
  **if** $random < \epsilon_2$ **then**
    $level \leftarrow$ random choice from $candidates$, weighted inversely by number of playthroughs
  **else**
    $level \leftarrow$ best match from $candidates$, as determined by rating system comparing player and level ratings
  **end if**
**end if**
**return** $level$

**Algorithm 2** Rating Array Level Selection

*// Replaces selection of best match from Algorithm 1*
**Input:** $candidates$
**Output:** $level$, $target\_score$
$exp\_scores$ = { player's expected score on each candidate level based on player's rating and the level's rating array }
$target\_scores$ = { on each candidate level, the score that poses an adequate amount of challenge to the player as determined by the DDA system (details in [5]) }
$eligible \leftarrow candidates$ levels where player's $exp\_score$ and $target\_score$ are both greater than the current high score on that level
**if** $eligible = \varnothing$ **then**
  $eligible \leftarrow candidates$ levels where at least player's $exp\_score$ is greater than the current high score on that level
  **if** $eligible = \varnothing$ **then**
    $eligible \leftarrow candidates$
  **end if**
**end if**
$level \leftarrow$ best match from $eligible$, as determined by rating system comparing player and level ratings
**return** $level$, $target\_scores_{level}$

serve, we used two $\epsilon$ values—$\epsilon_1$ and $\epsilon_2$. With probability $\epsilon_1$, we served a level completely at random without using the skill chain to determine eligibility. Else we determined the set of eligible levels using the skill chain and with probability $(1-\epsilon_1)\epsilon_2$, from these, served a level at random, weighted by the number of times they had been played with levels played fewer times given more weight. Else with probability $(1-\epsilon_1)(1-\epsilon_2)$, we used the matchmaking system to find the best matching level from among the eligible ones as per usual. The method is described in Algorithm 1. For the $\epsilon$-greedy evaluation, each level had a single fixed target score.

To test the $\epsilon$-greedy algorithm, we ran a human intelligence task (HIT) for each game, recruiting players via running HITs on Amazon Mechanical Turk. Each HIT paid $1.40 but payment was made in advance and playing the game was entirely optional. This was based on a payment strategy explored in [19]. Players in each game were randomly assigned to one of 4 conditions based on the epsilon values (for simplicity, we use the same value for both epsilons, i.e. $\epsilon = \epsilon_1 = \epsilon_2$)—0, 0.1, 0.2 or 1. Epsilon values of 0 and 1 were equivalent to using the regular matchmaking algorithm and serving levels completely at random respectively. 196 and 213 players were recruited for *Iowa James* and *Paradox*, respectively. We looked at:

- *Play Time* - time in seconds that the player spent playing
- *Levels Completed* - no. of levels completed by players

Iowa James

| *Variable* | $\epsilon = 0$ (orig.) | $\epsilon = 0.1$ | $\epsilon = 0.2$ | $\epsilon = 1$ (random) |
|---|---|---|---|---|
| Play Time ($p = .56$) | 224.75 | 246.1 | 347.58 | 510.97 |
| **Levels Completed** ($p = .03$) | $2^a$ | $1^{ab}$ | $1^{ab}$ | $0^b$ |
| Levels Lost ($p = .15$) | 3 | 3 | 4.5 | 4 |
| **Level Rating Error** ($p < .01$) | $207.3^a$ | $162.45^b$ | $176.61^{ab}$ | $93.64^c$ |

Paradox

| *Variable* | $\epsilon = 0$ (orig.) | $\epsilon = 0.1$ | $\epsilon = 0.2$ | $\epsilon = 1$ (random) |
|---|---|---|---|---|
| Play Time ($p = .51$) | 653.08 | 620.24 | 290.62 | 749.07 |
| **Levels Completed** ($p < .01$) | $4^a$ | $4^a$ | $3.5^a$ | $1^b$ |
| Levels Lost ($p = .14$) | 2 | 1 | 2 | 2 |
| **Level Rating Error** ($p < .01$) | $181.56^a$ | $126.87^b$ | $88.81^c$ | $73.22^d$ |

TABLE I
IOWA JAMES (TOP) AND PARADOX (BOTTOM) EPSILON COMPARISON SUMMARY. SIGNIFICANT OMNIBUS TESTS SHOWN IN BOLD; CONDITIONS SHARING A LETTER SUPERSCRIPT WERE NOT FOUND SIGNIFICANTLY DIFFERENT IN POST-HOC TESTS.

| *Variable* | Array | No-Array |
|---|---|---|
| Play Time ($p = .3$) | 529.16 | 411.39 |
| **Levels Completed** ($p < .01$) | $4^a$ | $3^b$ |
| Levels Lost ($p = .77$) | 2 | 2 |

TABLE II
PARADOX ARRAYS COMPARISON SUMMARY. SIGNIFICANT OMNIBUS TESTS SHOWN IN BOLD; CONDITIONS SHARING A LETTER SUPERSCRIPT WERE NOT FOUND SIGNIFICANTLY DIFFERENT IN POST-HOC TESTS.

- *Levels Lost* - no. of levels players failed to complete
- *Level Rating Error* - mean squared error between level ratings encountered by the player in their matches, and final level ratings after the experiment; final ratings were determined by plays when $\epsilon = 1$, as that corresponded to random level assignment, used in prior work, and assumed to be most accurate

For each variable, we ran an omnibus Kruskal-Wallis test across the four conditions. If significant, we then ran pairwise post-hoc Wilcoxon Rank-Sum tests using the Holm correction, for all pairs of conditions. Results are given in Table I. For both games, the omnibus test found significant differences across conditions for *Levels Completed* and *Level Rating Error* with significant post-hoc differences also being observed for both variables in both games. Looking at significant comparisons, we found that in both games, $\epsilon = 1$ (i.e. random assignment) had the lowest rating error, unsurprising as this condition was used for calculating the rating error. Similarly, in both games, $\epsilon = 0.1$, and additionally in *Paradox*, $\epsilon = 0.2$, had lower rating error than $\epsilon = 0$. This indicates that using a non-zero epsilon could help improve level rating estimates in an online matchmaking system. In both games, there were no differences in level completions among $\epsilon = 0, 0.1, 0.2$, and $\epsilon = 1$ was different from at least one of the other conditions, indicating that even in the presence of the $\epsilon$-greedy approach, matchmaking is still able to perform useful level assignment.

## VI. EVALUATION: RATING ARRAYS

We also tested the use of level rating arrays [6] in an online setting. Here, we assign each level an array of ratings for specific scores rather than a single rating. This lets us set score targets for each level dynamically for different players rather than use the same target across players, enabling more fine-grained DDA. A further benefit is we might find players likely to set new high scores for levels, and thus, find better solutions to the problem that level represents. More details about rating

arrays can be found in [6]. For our experiment, for each level, we use an array of 10 elements corresponding to evenly spaced scoring thresholds 0.1 apart (i.e. thresholds of 0.1, 0.2,, etc).

For testing rating arrays, we ran a HIT using the same payment parameters as above but for only *Paradox*, since specific levels in *Iowa James* do not necessarily correspond to specific problem instances for which new high scores might be valuable. Players were randomly assigned to an array or no-array condition with the former using rating arrays and the latter using one rating per level as in the prior experiment. Under the array condition, we used a modified version of the previous algorithm to prioritize serving levels where players could set new high scores, described in Algorithm 2. Note that this algorithm describes how the matchmaking would take place only if both $\epsilon$ checks were passed. In cases where either $\epsilon$ check was not passed, or in the no-array condition, a target score of 1 (the maximum possible) was used.

A total of 114 players were recruited. Based on the prior experiment, we used $\epsilon = 0.1$, looked at *Play Time*, *Levels Completed* and *Levels Lost* and ran a pairwise Wilcoxon rank-sum test using the Holm correction. Results (shown in Table II) suggest that players completed significantly more levels using rating arrays than when using a single rating per level.

Of the 19 non-tutorial levels used, we found that the highest score was found in the array condition 5 times, in the no-array condition 8 times, and there was a tie between the two 6 times. In each of the tie situations, the high score was reached in a fewer number of matches in the array condition than in the no-array condition. A paired Wilcoxon signed-rank test found the difference in high scores not significant with $p = .5$.

Looking at significant comparisons, we found that although using rating arrays did not result in finding more number of higher scores in this case, it did result in more level completions, likely due to the tailoring of target score thresholds making it easier for players to complete levels. Interestingly however, this did not appear to prevent the array approach, in aggregate, from finding similar high scores as the fixed threshold no-array approach, even though the no-array condition always asked players to reach the highest possible score.

## VII. Conclusion and Future Work

We presented an online version of a prior DDA system by using an $\epsilon$-greedy strategy to address a cold-start that previously made this infeasible. The system also applied rating arrays, which were shown to help players complete more levels, thus finding more potential solutions to underlying problems modeled by the levels.

There are several avenues for future work. The skill chains and skills required for each level were defined manually. In the future, we could explore automatically inferring the skill chain of a game and level skills by, e.g., analyzing playthroughs of levels to extract patterns of skill usage. A simplifying assumption made by the system is that once a player completes a level, they are considered to acquire all the skills necessary to complete that level, and retain them for the rest of the game. However, a player might complete a level without executing each listed skill. Thus, we'd like to explicitly track the specific skills executed by players when playing through levels, and probabilistic or partial skill acquisition. Finally, we applied rating arrays in an online setting but only for *Paradox*. In the future, we want to test rating arrays using other games where levels have well-defined notions of partial completion, such as the protein-folding HCG *Foldit*. Rating arrays could also find use in educational games where levels try to teach concepts and thus the array ratings could correspond to varying levels of students' mastery of the concepts being taught.

## References


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

[2] 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 *International Conference on the Foundations of Digital Games*, 2017.

[3] A. Sarkar and S. Cooper, "Meet your match rating: Providing skill information and choice in player-versus-level matchmaking," in *International Conference on the Foundations of Digital Games*, 2018.

[4] ——, "Using a disjoint skill model for game and task difficulty in human computation games," in *Annual Symposium on Computer-Human Interaction in Play Companion*, 2019.

[5] ——, "Evaluating and comparing skill chains and rating systems for dynamic difficulty adjustment," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2020.

[6] ——, "Using rating arrays to estimate score distributions for player-versus-level matchmaking," in *International Conference on the Foundations of Digital Games*, 2019.

[7] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, 2005.

[8] M. Zohaib and H. Nakanishi, "Dynamic difficulty adjustment (DDA) in computer games: a review," *Advances in Human-Computer Interaction*, 2018.

[9] J. Alexander, J. Sear, and A. Oikonomou, "An investigation of the effects of game difficulty on player enjoyment," *Entertainment Computing*, vol. 4, no. 1, 2013.

[10] A. Zook and M. Riedl, "A temporal data-driven player model for dynamic difficulty adjustment," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.

[11] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: dynamic difficulty adjustment through level generation," in *FDG Workshop on Procedural Content Generation*, 2010.

[12] Valve Corporation, "*Left 4 Dead*," Game, 2008.

[13] J. Frommel, F. Fischbach, K. Rogers, and M. Weber, "Emotion-based dynamic difficulty adjustment using parameterized difficulty and self-reports of emotion," in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, 2018.

[14] D. Cook, "The chemistry of game design," 2007, *Gamasutra*. http://www.gamasutra.com/view/feature/1524/the_chemistry_of_game_design.php.

[15] A. Echeverria, E. Barrios, M. Nussbaum, M. Amestica, and S. Leclerc, "The atomic intrinsic integration approach: A structured methodology for the design of games for the conceptual understanding of physics," *Computers & Education*, vol. 59, no. 2, pp. 806–816, 2012.

[16] B. Horn, J. A. Miller, G. Smith, and S. Cooper, "A Monte Carlo approach to skill-based automated playtesting," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2018.

[17] K. Siu, M. Guzdial, and M. Riedl, "Evaluating single player and multiplayer in human computation games," in *International Conference on the Foundations of Digital Games*, 2017.

[18] A. Sarkar and S. Cooper, "Inferring and comparing game difficulty curves using player-vs-level match data," in *IEEE Conference on Games*, 2019.

[19] ——, "Comparing paid and volunteer recruitment in human computation games," in *International Conference on the Foundations of Digital Games*, 2018.