Ordering Levels in Human Computation Games using Playtraces and Level Structure

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

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

***Abstract*—Prior work using skill chains for matchmaking-based dynamic difficulty adjustment in human computation games required skill chains to be manually defined for a game, and each level to be manually annotated with the individual skills needed to complete that level. In this work, we present two approaches for defining level orderings for DDA in the platformer HCG *Iowa James* without using such manually-defined skill chains and annotations. The first involves sequences of action-context pairs found in gameplay traces. The second consists of applying K-means clustering on segments of levels. Our results show that both new approaches outperform baseline random level ordering and perform similarly to the skill chain approach.**

***Index Terms*—dynamic difficulty adjustment, human computation games, rating systems, skill chains, playtrace, clustering**

I. Introduction

Human computation games (HCGs) attempt to solve real-world problems by modeling them as game levels and harnessing the collective ability of crowdsourced players. Dynamic difficulty adjustment (DDA) in such games is a challenge since levels cannot easily be modified to tailor difficulty without altering the problem being modeled. To this end, prior work has performed DDA in HCGs through the combined use of ratings-based matchmaking and skill chains. The former involves repurposing player-vs-player (PvP) rating systems (e.g. Glicko-2 [1]) as player-vs-level (PvL) rating systems by assigning ratings to players and levels based on ability and difficulty respectively. Then matchmaking is used to match players with appropriately hard levels. Skill chains, on the other hand, define the order in which in-game skills build on each other and should ideally be acquired. For performing DDA, previous work has used the game's skill chain to identify the set of levels most suitable for a given player given their currently acquired set of skills. Then, the most appropriate level from within this set is chosen via matchmaking using the rating system. In this way, the rating system and skill chain work together to accomplish DDA by means of dynamically ordering the levels appropriate to the ability of the player.

While shown to be effective, the use of skill chains in this approach necessitates significant manual authoring. First, a skill chain must be defined for a given game. Then, each level in the game must be annotated with the set of individual skills required to complete that level. Ideally, we want a DDA approach that requires a lesser amount of manual authoring.

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

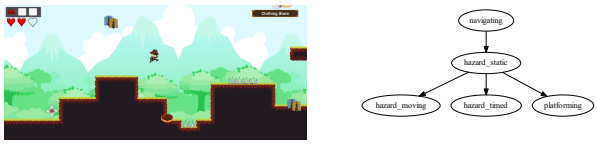


Fig. 1. (left) Iowa James screenshot and (right) skill chain, developed in previous work, used in the skill-chain based ordering.

In this paper, we thus explore two approaches for obtaining level orderings for DDA in HCGs without having to rely on manually authored skill chains and annotations. For the first of these, we consider sequences of action-context pairs in playtraces of successful player attempts at levels and order levels based on relative proportions of similar action-context pairs. For the second, we apply K-means clustering on level segments to cluster segments with similar level structure. We then order levels based on cluster memberships of their constituent segments, the idea being that levels requiring similar skills will contain similar level structures within them. Like skill chains, both these new methods thus help identify sets of levels eligible for the player based on their current ability, from which we select the most appropriate level again using ratings-based matchmaking as in the prior approach. While both approaches do require some designer input to set up for a specific game, as they are based on gameplay traces and level geometry, they do not require the manual specification of a skill chain or per-level manual annotation of skills. We tested our new approaches using the platformer HCG *Iowa James* and compared with the original skill chain-based approach and a random baseline. Our results suggest that the proposed approaches perform similarly to the prior manual approach, which all outperform the random baseline.

II. Background

Dynamic difficulty adjustment (DDA) [2] refers to dynamically balancing the difficulty of a game relative to player ability. Traditional methods for DDA that modify game content are ill-suited for HCGs which model real-world problems and thus are not amenable to methods that modify levels. Instead, prior works [3], [4], [5] have used skill chains and rating systems for DDA in HCGs. Rating systems (e.g. Glicko-2 [1]) are typically used for PvP matchmaking for chess and esports but have found use for DDA by being repurposed

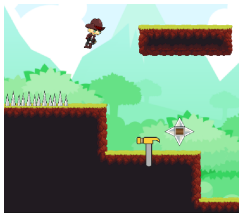


Fig. 2. Example action-context pair of ('jump', 101101). Context bits indicate (*Ground*, *Moving-Platform*, *Item*, *Spikes*, *Timed-Spikes*, *Ninja-Star*).

to matchmake between players and levels. Skill chains [6] define the order of skill acquisition in a game by means of a directed graph and are used for defining levels suitable for players based on their skills. Together, both of these are used for DDA in HCGs by controlling level ordering rather than modifying the content within them. However, skill chains require significant manual authoring in terms of both defining them for a game and then annotating each level with the required skills. We wish to reduce this load through using playtrace data and clustering. Our use of playtraces to order levels is similar to [7] who use a trace-based framework for generating problem progressions for an algebra learning game and [8] who generate progressions for a foreign language learning game via reinforcement learning. Additionally, prior works applying clustering on level structures include [9] which clusters similar Mario and Lode Runner structures for level generation and [10] which clusters categories of Mario level chunks for training probabilistic graphical models to generate new levels. However, such applications in the context of HCGs are less prevalent.

III. Iowa James

We use *Iowa James*, a 2D side-scrolling platformer HCG modeling the task of object categorization. Similar to *Gwario* [11] which inspired it, each level in Iowa James represents a scenario where players must collect and avoid items relevant and irrelevant to that scenario respectively. Collecting all relevant items unlocks a treasure chest at the end to progress to the next level. The game features typical platformer mechanics of running and jumping as well as hazards. Players lose a life either by collecting an incorrect item or being struck by a hazard and have three lives to complete each level. The game featured 3-item and 7-item versions of 25 different maps for a total of 50 levels. A skill chain for the game developed in prior work [5] is shown in Figure 1.

IV. Method

A. Skill Chain-based DDA

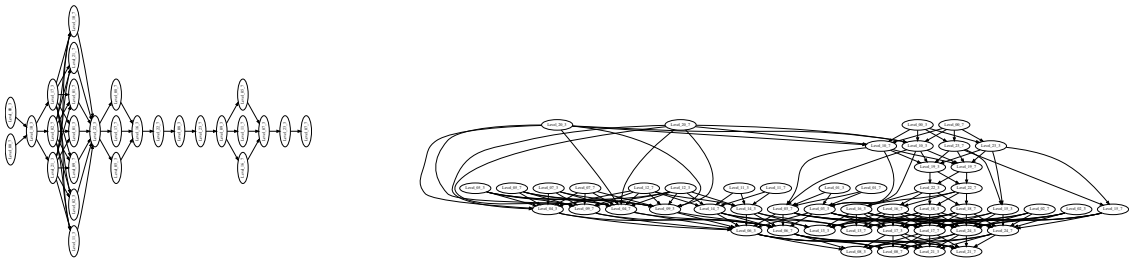
The original skill chain and ratings-based approach for DDA has been introduced and described in prior work [3], [4]. It consists of three steps: 1) manually defining the skill chain for the game 2) manually annotating each level in the game with the individual skills needed to complete it, and determining a rating for each level based on its difficulty and 3) using the Glicko-2 [1] rating system to matchmake between players and levels and then update the player ratings and skills after each match. For this work, we took the level ratings and skill annotations used in prior work [4] involving Iowa James, with level ratings ranging from 1496 (easy) to 1927 (hard). These level skills and ratings are then used in step 3 where PvL matchmaking is done to serve levels to players. Briefly, this starts by assigning a default rating of 1500 and an empty set of skills to each new player. Then to get the next level, the set of eligible levels is determined based on the player's currently acquired set of skills. The rating system then chooses which eligible level to serve based on the player's current rating. After the player plays through the level, the player's rating and skills are updated based on if the player completed the level or not. A more detailed description of the working of this system can be found in [4]. Note that in this method, there are two points where manual input is required. First, one must manually craft the skill chain of the game. Second, each level must be manually annotated with the set of skills required to complete them. It is this authorial burden we wish to reduce with the two new approaches described next.

B. Action-Context Pairs in Play Traces

For our first approach, we used sequences of action-context pairs found in play traces of successful player attempts at levels i.e. instances of players completing the level. Each such pair was thus a 2-tuple consisting of a player action and a level context indicating the context within the 10-tile neighborhood of the player when the action was performed. For this work, action was one of *left*, *right*, *jump* and *wrong_item* with the latter indicating players collecting an incorrect item. The context was in the form of a length-6 bitstring with each bit indicating the presence or absence of a specific game element within the neighborhood of the player while performing the action. In order, the bits correspond to: *ground*, *moving platform*, *collectable item*, *spikes*, *timed rising spikes*, *ninja star*. An example action-context pair is shown in Figure 2.

We gathered playtrace data using players recruited via a human intelligence task (HIT) on Amazon Mechanical Turk. Players were paid $2 but payment was made in advance and playing the game was completely optional based on a payment strategy explored in [12]. To ensure similar amounts of data across levels of all difficulties, players were served levels at random. Through this HIT, we gathered data for 60 players. For each instance of a player playing a level, we logged the trajectory of time-ordered action-context pairs during the playthrough of the level. Since we are interested in ordering levels based on the skills required to complete them, we filter out trajectories of losing playthroughs since we do not want to consider actions that did not lead to level completion.

Using the winning trajectories, for each level, we determine the set of unique action-context pairs that appear in a certain threshold percentage of trajectories involving that level. Then, to order the levels, we consider each pair of levels A and B. If the percentage of A's action-context pairs in B are greater than the percentage of B's action-context pairs in A, then A comes before B in the ordering. E.g., if 50% of the skills required for A are also required for B but only 30% of the skills required for B are required for A, then A comes before



PT-10, left-to-right KM-20, top-to-bottom

Fig. 3. Iowa James level orderings using 10% thresholding on playtrace data (PT-10) and clustering segments using K-means with K=20 (KM-20).

B. Intuitively, an easier level should require fewer of the skills required by harder levels than vice-versa. Representing these (partial) orderings as graphs with levels as nodes, if the above condition is true, then there is a directed edge from A to B, else the edge is from B to A unless they have the same percentage in which case there is no edge. Finally, after processing all level pairs, we apply a transitive reduction to get the final level ordering graph. Transitive reduction removes direct edges between two nodes if they are connected by a separate path of length greater than 1 and is thus useful for reducing the density of a graph. To determine the percentage of winning trajectories to consider, we generated the above graphs for thresholds=10, 20, ..., 100% and analyzed the resulting orderings based on our knowledge of the levels in the game and general idea of level difficulties based on skill annotations and ratings from prior works. For e.g., we know levels 00_3 and 00_7 should appear in the beginning since they were designed to be starting levels. We found that higher thresholds led to sub-optimal level orderings since this caused the harder levels to end up with fewer action-context pairs than the easier levels as more actions are possible in big difficult levels than small simple levels. Since we only consider relative proportions of action-context pairs, this causes harder levels to be ordered ahead of easier ones. By using lower thresholds, we obtained orderings closer to expectation. We thus opted to use the graphs obtained using 10% (PT-10) and 20% (PT-20) thresholding. While we use a fair amount of heuristics here, we note that it is common practice (and arguably beneficial) to incorporate some designer knowledge into automated and semi-automated methods.

C. Clustering

For the second approach, we clustered segments from all levels in the game. For this, we slid a 16x16 window across all 50 levels to obtain 52,716 segments on which we applied K-means using scikit-learn [13]. Clusters thus represent groups of segments having similar level structures. After obtaining the clusters, for each level, we identified the clusters that contained at least 1 segment from that level. Then the order for each pair of levels was determined based on if cluster memberships of one level were a subset of those of the other. For e.g., consider cluster memberships represented as length-k bit vectors where k is the total number of clusters and the i-th bit is set to 1 or 0 if a segment from the i-th cluster is present or absent in that level. Then if k=3 and vectors for levels A, B and C are 100, 101 and 110 respectively, then A comes before B and C since 100 is a subset of both 101 and 110 but there is no relationship between B and C. We represent these (partial) orderings as graphs with nodes representing levels and a directed edge from A to B indicating A is a subset of B. We applied transitive reduction to obtain the final graph, generating such graphs for k=1 to 20. Again, we want orderings that have levels 00_3 and 00_7 appearing first and prefer deeper over shallower graphs. We found either or both of these constraints to not be satisfied in graphs generated using values of k less than 6. Lower values of k lead to flatter, broader graphs since there are fewer clusters, making it more likely for a greater number of levels to have their segments belong to similar clusters. To cover the range of remaining k values, we experimentally compared the graphs produced by k=6 (KM-6) and k=20 (KM-20).

Note that while there is still some amount of manual authoring involved in the new approaches, it is only for the game as a whole rather than for each level in the game unlike the skill chain-based approach. In the playtrace method, the context must be manually defined but once defined, the same context is used across all levels. The clustering based method is even more automated. For space, we only show orderings PT-10 and KM-20 in Figure 3. We can think of these as skill chains where the levels themselves correspond to skills and completing a level corresponds to acquiring that skill.

V. Evaluation

In addition to the preliminary HIT to gather playtraces, we ran 3 HITs—the first to determine which playtrace-based ordering to use, the second to determine which clustering-based ordering to use and the third to compare the orderings selected in the first 2 HITs to the original skill chain-based ordering and a random baseline. All 3 HITs used the same payment scheme as the preliminary HIT. The HITs to find which playtrace-based and clustering-based orderings to use each had 2 conditions with each player being randomly assigned to one of the two. For these HITs, we looked at:

- *Levels Completed* - no. of levels completed by the player
- *Total Matches* - total no. of matches played by the player

We gathered data for 111 and 113 players for the playtrace and clustering HITs respectively. Results are shown in Table I. For the playtrace HIT, we observed that 10% thresholding (PT-10) led to players completing a significantly higher number of levels while playing a similar number of matches compared

Variable	PT-10 (n=59)	PT-20 (n=52)
Levels Completed ($p = .039$)	2	1
Total Matches ($p = .24$)	6	5.5

Variable	KM-6 (n=55)	KM-20 (n=58)
Levels Completed ($p = .52$)	1	2
Total Matches ($p = .9$)	6	6

TABLE I

MEDIAN VALUES FOR VARIABLES COMPARING THE PLAYTRACE-BASED (PT) AND CLUSTERING-BASED (KM) ORDERINGS. BOLD VALUES WERE SIGNIFICANT BASED ON A WILCOXON RANK-SUM TEST.

Variable	RAND (n=78)	SKILL (n=96)	KM-20 (n=85)	PT-10 (n=76)
Levels Completed ($p < .01$)	1^a	2^b	2^b	2^b
Total Matches ($p = .77$)	8	6	6	6
Correct Items ($p = .052$)	7.5^a	9.5^{ab}	8^{ab}	14^b
Incorrect Items ($p = .33$)	7	6	6	7.5
Highest Level Rating ($p < .01$)	1496^a	1669^b	1669^b	1854^c

TABLE II

MEDIAN VALUES FOR VARIABLES ACROSS ALL CONDITIONS. BOLD VALUES WERE SIGNIFICANT BASED ON AN OMNIBUS KRUSKAL-WALLIS TEST (BORDERLINE SIGNIFICANT VALUES INCLUDED). VALUES WITH SHARED LETTER SUPERSCRIPTS^{abc} WERE NOT SIGNIFICANTLY DIFFERENT IN PAIRWISE POST-HOC COMPARISONS.

to 20% thresholding (PT-20), determined by a Wilcoxon rank-sum test. Using similar testing on the clustering data, we observed no statistical differences between the two conditions but the median number of levels completed was higher for KM-20. We thus used PT-10 and KM-20 for the final HIT which hence had 4 conditions:

- RAND - randomly pick a level yet to be completed
- SKILL - use skill chain to determine eligible levels and rating system to pick level to serve
- KM-20 - use KM-20 level ordering to determine eligible levels and rating system to pick level to serve
- PT-10 - use PT-10 level ordering to determine eligible levels and rating system to pick level to serve

For this HIT, we considered 3 additional variables:

- *Correct Items* - the total number of correct items collected by a player during a playthrough
- *Incorrect Items* - the total number of incorrect items collected by a player during a playthrough
- *Highest Level Rating* - the highest rated (i.e. most difficult) level completed by a player

We recruited 335 players, each randomly assigned to one of the 4 conditions. For each variable, we ran an omnibus Kruskal-Wallis test across conditions and if significant, pairwise Wilcoxon rank-sum tests with the Holm correction. Results are shown in Table II. We found significant omnibus differences for *Levels Completed* and *Highest Level Rating* and a borderline significant difference for *Correct Items*. For *Levels Completed*, unsurprisingly, RAND did significantly worse than the others with no difference between those three. For *Correct Items*, significant post-hoc differences were observed only between PT-10 and RAND but the median correct items collected for PT-10 was far higher than any other condition. For *Highest Level Rating*, PT-10 did significantly better than other conditions while RAND did significantly worse than others, with KM-20 and SKILL performing similarly.

Our results suggest that the new orderings allow players to complete a similar amount of levels as the skill chain method while reducing authorial load. PT-10 also allowed players to complete significantly harder levels. This is particularly useful for HCGs where we want to solve hard problems which would typically be represented by difficult levels. PT-10 was also the only one that led players to collect significantly more correct items than RAND though it is worth noting that players also collected the most incorrect items under PT-10 suggesting that there may be some trade-off involved where this ordering favors the higher item versions of levels leading to players collecting more correct items but also being more likely to collect more incorrect items. While KM-20 does not outperform SKILL or PT-10 along any variable, it requires the least amount of manual input among the three methods and does not do any worse than either of the other two. Overall, we find that our new ordering approaches either improve upon the skill chain-based approach or do no worse, while reducing authorial burden. It is also worth noting that the median number of matches played varied from 6 to 8, suggesting that early parts of orderings are more important than later parts.

VI. CONCLUSION AND FUTURE WORK

We presented two approaches for learning level orderings for DDA in HCGs using less authorial load than the existing skill chain-based approach. The new methods involved analyzing playtrace data of player actions and clustering level segments based on structure. We found that the new methods either outperformed or performed similarly to the skill chain-based DDA system. In the future, we are interested in applying these methods on other types of HCGs as well as to learn progressions for educational games. We also want to explore context relationships where a context without an element is considered a subset of the same context with that element.

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

[3] A. Sarkar and S. Cooper, "Using a disjoint skill model for game and task difficulty in human computation games," in *CHI Play*, 2019.

[4] ——, "Evaluating and comparing skill chains and rating systems for dynamic difficulty adjustment," in *AIIDE*, 2020.

[5] ——, "An online system for player-vs-level matchmaking in human computation games," in *IEEE Conference on Games (CoG)*, 2021.

[6] D. Cook, "The chemistry of game design," 2007, *Gamasutra*.

[7] E. Andersen, S. Gulwani, and Z. Popovic, "A trace-based framework for analyzing and synthesizing educational progressions," in *CHI*, 2013.

[8] T. Mu, S. Wang, E. Andersen, and E. Brunskill, "Combining adaptivity with progression ordering for intelligent tutoring systems," in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, 2018.

[9] S. Snodgrass and S. Ontanon, "A hierarchical mdmc approach to 2d video game map generation," in *AIIDE*, 2015.

[10] M. Guzdial and M. Riedl, "Game level generation from gameplay videos," in *AIIDE*, 2016.

[11] K. Siu, M. Guzdial, and M. Riedl, "Evaluating single player and multiplayer in human computation games," in *FDG*, 2017.

[12] A. Sarkar and S. Cooper, "Comparing paid and volunteer recruitment in human computation games," in *FDG*, 2018.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.