Solution Path Heuristics for Predicting Difficulty and Enjoyment Ratings of Roguelike Level Segments

Colan Biemer
biemer.c@northeastern.edu
Northeastern University
Boston, USA

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

ABSTRACT

When generating levels, algorithmically evaluating the results is essential. In this paper, we looked at predicting a level's difficulty and enjoyment. Past work has approached this problem for puzzle games like Sudoku by analyzing the characteristics of the initial level, the solved level, and the process that led to that solution. In this work, we examined a set of heuristics for Roguelike levels and their solutions, and their relationship to subjective player ratings of the levels. We gathered ratings of difficulty and enjoyment of levels in a study with 143 players. We ran an ablation study on the set of heuristics to find the best combination of heuristics for predicting difficulty and enjoyment with a linear regression model, and found solution path-based heursitics performed well. However, these models did not outperform a simple baseline for predicting enjoyment. Jaccard similarity on paths—a method we have not seen used in the field of game AI—was a useful predictor of difficulty. Testing proximity to enemies across a solution path is the only heuristic needed to predict how enjoyable a level will be.

CCS CONCEPTS

• **Human-centered computing**;

KEYWORDS

procedural content generation, difficulty, player study

ACM Reference Format:
Colan Biemer and Seth Cooper. 2024. Solution Path Heuristics for Predicting Difficulty and Enjoyment Ratings of Roguelike Level Segments. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG 2024), May 21–24, 2024, Worcester, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3649921.3659846

1 INTRODUCTION

Procedural content generation (PCG) is an area of game artificial intelligence that has been extensively researched [20, 27, 29, 33]. Creating enjoyable and challenging levels poses a significant challenge in level generation. As a result, there has been a plethora of work looking at dynamic difficulty adjustment [16, 42], experience-driven games [15, 30], and more. In the context of PCG, we need methods that automatically and accurately assess content based on concepts like difficulty and enjoyment.

One way to approach this problem is player modeling [39]. By developing a model that finds what a player likes and dislikes, it can be used to guide a PCG system to build levels that are custom-tailored to the user [18]. The problem is that the player model must quickly and accurately model the user to be usable. One previous work used a method that found levels within a reasonable degree of difficulty for a target user within four levels played [11].

In our work, we conducted a study with 143 players on a roguelike game, where we asked players to rate levels on a 7-point Likert scale based on difficulty and enjoyment. Separately, levels were analyzed based on eleven heuristics. These heuristics used the initial level, the solved level, and the path found by an agent to solve the level. We used these heuristics as input for a linear regression model and conducted an ablation study to determine which heuristics were most useful.

We found that heuristics based on solution paths were the most useful for predicting the difficulty and enjoyment of a level. A heuristic that uses Jaccard similarity, see Section 3.2, was highly effective in predicting the difficulty of a level. This finding stands out because we are not aware of any other work that has used Jaccard similarity for games and difficulty research. Jaccard similarity as a heuristic played a less prominent role in predicting how enjoyable a level was, though. Instead, the most important heuristic was one that used the path to solve the level and looked for enemies nearby. This heuristic was also used prominently in predicting difficulty. However, we did not find a combination of heuristics as input into a linear regression model which outperformed a simple baseline for predicting enjoyment.

Surprisingly, we found little if any correlation between the degree of difficulty and enjoyment experienced by the player [22].

2 RELATED WORK

Evaluating a roguelike level for its difficulty is typically based on solution paths. Gellel and Sweetser [8] calculated the "interestingness" of a level as the difference between the length of the solution path and the length of the solution path if there were no obstacles—e.g. locks, keys, enemies, etc. More recently, Weeks and Davis [37] used the same approach except on a layout of rooms and hallways to estimate the difficulty of a layout. Sampaio et al. [25] used a different approach where given a desired difficulty, required items were placed into a dungeon based on the distance to the player's starting point and the nearby entities to the placement point. Unfortunately, these studies did not validate their heuristics with a player validation study.

FDG 2024, May 21–24, 2024, Worcester, MA, USA

ACM ISBN 979-8-4007-0955-5/24/05
https://doi.org/10.1145/3649921.3659846

Solution paths are not the only way to approach difficulty. Gonzalez-Duque et al. built an adaptive system to predict the time it will take a specific player to beat a level [10], specifically for a roguelike game and Sudoku. Time is an approximation for difficulty in puzzle games in that we can consider that the longer the player takes to solve a level, the harder the level. In their work, they evaluated a Bayesian optimization algorithm by running a user study and found positive results when compared to other baseline approaches.

A different approach to creating heuristics by hand is to use crowd-sourced data. Jennings-Teats et al. ran a study where users were shown short-level segments of a platformer and asked to classify difficulty between one (easy) and six (hard) [18]. With this data, they trained a classifier to rank the difficulty of any input level. Reis et al. took a different approach in their work by crowd-sourcing difficulty evaluation of all platformer level segments [22]. One interesting point of note is that Reis et al. found a high correlation between difficulty and enjoyment, which we could not replicate in our work—we discuss this more in Section 4.4.

Mariño et al. ran a study on metrics for estimating difficulty, enjoyment, and visual aesthetics of *Mario* levels on a 7-point Likert scale [21]. They found that the computation metrics they used (linearity [31], leniency [31], density, and compression distance [28]) could not accurately predict enjoyment and can be misleading when estimating difficulty. They conclude that current computational metrics cannot replace a well-designed user study. While our work uses a roguelike instead of a platformer, we also found that the heuristics we tested poorly estimated enjoyment. When it comes to difficulty, we found that a combination of heuristics can provide a strong estimation of difficulty, but the only way a combination was found was through a user study.

Wong et al. created a method designed to be game agnostic in the sense that it could estimate difficulty for any puzzle game that can be solved by a solver [38], such as Clingo [7]. They used the results of the solver to rate difficulty based on the number of required solver calculations, guesses, backtraces, certain branches, uncertain branches, and the ratio of certain to uncertain branches. These variables can be broken down into three categories: *initial features* (features related to the start state of the puzzle), *solution features* (features related to the end state of the puzzle), and *dynamic features* (features related to the solution of the puzzle) [36]. These six variables were fed into a genetic algorithm to find a formula that best matched a training set of Sudoku puzzles to difficulty. They found that the number of uncertain branches was highly correlated to difficulty. A different approach is to use weighted linear regression instead of a genetic algorithm [36].

Another area of work is the study of chess puzzle difficulty [12, 14]. Of note in this area are websites like Chess.com and Lichess.org, which have many players playing chess puzzles. With so many players, there is no need to automate difficulty evaluation. Instead, these websites can actively update puzzle ratings—referring to the expected player rating needed to solve a puzzle—based on player performance and update real-time [9, 26]. Lichess keeps an open database of puzzles with ratings that could be used in future research to validate generic puzzle difficulty approaches.[1]

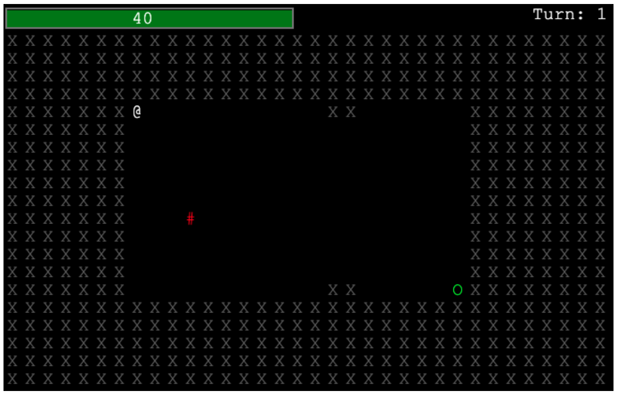

Figure 1: Tutorial level for *DungeonGrams*. The player is represented by the '@' symbol.

3 METHOD

3.1 DungeonGrams

DungeonGrams[2] is a simple top-down roguelike, see Figure 1. *DungeonGrams* was chosen because it has a dataset of 191 playable levels publicly available [3], and it also comes with an already implemented A* tree search [13] that can find a solution to a level.

The game starts with the player at the top left. A portal is at the bottom-right of the level and the goal is to unlock the portal by hitting every switch while avoiding enemies. There is a stamina mechanic where the player starts with forty stamina and every movement costs 1 stamina. The player loses if their stamina goes down to zero or if an enemy or spike comes in contact with them. However, the player can gain stamina by coming into contact with food: $stamina \leftarrow min(stamina + 25, 40)$.

We created an online version of the game for this work.[3]

3.2 Heuristics

Heuristics used initial features, solution features, and dynamic features [36]. We used the already implemented A* tree search in DungeonGrams to obtain the latter two feature types. A* returns an array of tile coordinates (x, y) which we refer to as a *path*.

The downside of using A* is that we don't have access to information like backtracking and uncertain branches [36, 38]. The benefit is that our work may be more accessible because it's easier to build a tree search for a game than re-implement a game as a logic program.

A* was run three times for each level. The first time was on the original level. The second was on a version of the level where enemies were removed. The third ran on a version of the level without enemies and switches.

We built 7 heuristics based on work discussed in Section 2; we used 2 heuristics—linearity and leniency—from the work of Smith and Whitehead [31], and we developed two of our own using Jaccard similarity. In total, we used 11 heuristics.

[1] https://database.lichess.org/#puzzles

[2] https://github.com/crowdgames/dungeongrams

[3] Link removed for anonymity.

$$\text{JaccardSimilarity(A,B)} = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

Jaccard similarity [17], see equation 1, is the size of the intersection of two sets divided by the size of the union. In terms of paths, it is the number of common points between two paths divided by the number of unique points for both paths. Our motivation for introducing and testing Jaccard similarity as a heuristic for difficulty and enjoyment comes from the observation that path length difference does not capture enough of the differences between two paths. For example, if there are two paths P_a—path to beat a level—and P_b—path to beat the level, but the level was modified so that all enemies were removed—and $|P_a| = |P_b|$. The path length difference ($|P_a| - |P_b| = 0$) implies that the two paths are the same. As a heuristic for this example, path length difference says that the two paths are equivalent. But, what if $P_a \cap P_b = \emptyset$? Since there are no common points between the two paths it is wrong to say that the paths are 100% similar. From this line of reasoning, we hypothesized that Jaccard Similarity would be better suited as a heuristic for estimating difficulty and enjoyment because it better captures changes in solution paths.

Below are the heuristics we used to evaluate levels:

path-no-enemies - Difference between the path length of the original level and the path length of the level with no enemies.
path-nothing - Difference between the path length of the original level and the path length of the level with no enemies and no switches.
jaccard-no-enemies - Jaccard similarity of the path of the original level and the path of the level with no enemies.
jacard-nothing - Jaccard similarity of the path of the original level and the path of the level with no enemies and no switches.
proximity-to-enemies - Using the completion path found for the original level, each position in the path was used to search for enemies up to three tiles away in every direction. For each enemy, one over the Manhattan distance to that enemy from the tile on the path was summed. This means that the more enemies near the optimal completion path, the larger the value of this heuristic. The sum was then divided by the length of the path. This is similar to a heuristic from Tremblay et al. which uses the exact distance for every agent via an A* search to calculate a distance to enemy metric [35].
proximity-to-food - The same as *proximity-to-enemies*, except it searches for food rather than enemies.
stamina-percent-enemies - Percent difference between the ending stamina for the tree search on the original level and the stamina left for the level with no enemies.
stamina-percent-nothing - Percent difference between the ending stamina for the tree search on the original level and the stamina left for the level with no enemies and no switches.
density [31] - Number of solid tiles, including spikes, divided by the total number of tiles in the level.
leniency [31] - Number of enemies, spikes, and switches divided by the total number of tiles in the level.
food-density - Number of food tiles divided by the total number of tiles in the level.

3.3 Player Study

We ran an online study that recruited 150 workers on Mechanical Turk. Methods had approval from the authors's Institutional Review Board.

Each participant was paid 1 dollar. We estimated hourly wage by tracking a user's time to beat a level, allowing for a maximum of 60 seconds per attempt. Based on actual playtimes after running the study, we added a bonus of 50 cents for an estimated median hourly wage of twelve dollars an hour.

Each participant was asked to agree to a consent form and then taken to the game. At the game, they were immediately presented with a payment code so they could receive payment without playing a single level. For those who opted to play, they first played a tutorial level that helped show the basics of how to play the game. If the player lost a level twice, they were given the option to give up and skip the level to avoid players becoming frustrated and quitting.

After they beat the tutorial level or gave up, a questionnaire appeared with two statements: "This level was difficult" and "This level was enjoyable to play." The player could respond by selecting between "strongly disagree" and "strongly agree" on a 7-point Likert scale, with the middle being "neutral".

After the survey, a random level was selected from the level dataset for the player to play and the survey followed after they completed the level. This continued until the player hit 11 levels played,[4] including the tutorial level, or they quit. No identifying information was requested or stored.

3.4 Predicting Difficulty and Enjoyment

The heuristics from Section 3.2 were calculated for every level and made into a dataset with median player-rated difficulty and median player-rated enjoyment. Note that the expected output is a continuous value instead of categorical. Our reasoning for this decision is as follows. We have multiple player responses for each level. One solution to take advantage of these would be to make the correct output for each level be the mode. However, there is an ordering factor implicit in our scale and we wanted to capture that not only for this work but for future work as well. As a result, we decided that converting the survey results to continuous values was best for this work. In that regard, we used the median instead of the mean because it better represents the ordinal data from a Likert survey [40], especially when the dataset size is small [32].

To evaluate which heuristics were important, we ran an ablation study using the training portion of the data. We used a linear regression model [24]. Due to the limited size of the dataset, we used k-fold cross-validation, where $k = 5$, and minimized the mean square error (MSE) with a train-test split of 0.8. We removed the tutorial level from the dataset due to being an outlier in the dataset, likely a result of being the first level played. The top ten combinations out of $2,047$ potentials were used to evaluate feature importance.[5] The best combination of heuristic features—as defined by the combination of heuristics which resulted in the lowest MSE—was used to train a model on the full training set and tested on the test set.

[4] Based on a pilot study, we calculated the average time spent per level. We used the average time to estimate costs and came up with eleven as the maximum number of levels a player can play and still be compensated fairly.

[5] There are 11 heuristics, meaning $2^{11} = 2048$ possible models. We did not test a model with no inputs, resulting in 2047 total models.

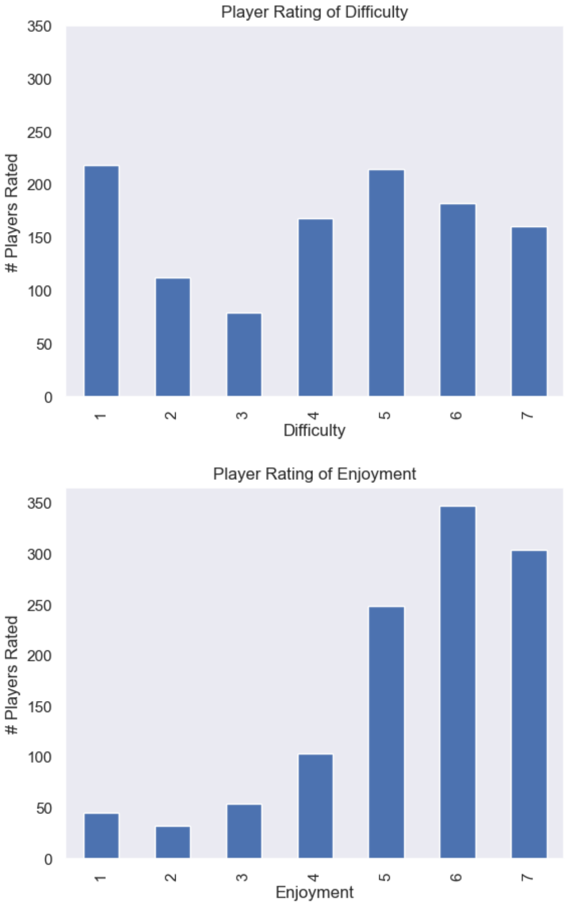

Figure 2: Survey results from players ranking difficulty and enjoyment for every level they played.

We also compared the best linear regression model to a simple baseline. For difficulty, it returned the mean difficulty across all surveys. For enjoyment, it returned the mean enjoyment score across all surveys.

4 EVALUATION

4.1 Player Study

Out of the 150 recruited, 143 participated and completed at least one level. Seventy participants completed all eleven levels. On average, each player completed 7.9 levels; this average does not take into account whether the player beat the level or gave up. In the case of giving up, players gave up on 41% of the levels they played. Not considering the tutorial level, levels were played by a minimum of 1 player, a mean of 5.2 players, a median of 5 players, and a max of 10 players with a standard deviation of 2.0; a non-random level selection method would have improved this distribution.

In total, players made 3,086 attempts to complete the levels. There were three reasons players lost: coming into contact with an enemy (71%), running out of stamina (20%), and running into a spike (9%). There was an overall win rate of 18%. On average, one playthrough of a level (not necessarily beating it) took 11.3 seconds. Ratings of difficulty and enjoyment can be seen in Figure 2. The median difficulty rating was 4, which links to "neutral". On the other hand, players gave enjoyment a median rating of 6, which links to "agree".

Heuristic	Difficulty	Enjoyment
density	**1.0**	0.0
food-density	0.0	0.0
jaccard-no-enemies	0.0	**0.5**
jaccard-nothing	**1.0**	0.0
leniency	0.2	0.0
path-no-enemies	0.0	0.0
path-nothing	0.4	**0.8**
proximity-to-enemies	**1.0**	**1.0**
proximity-to-food	0.2	**0.4**
stamina-percent-nothing	0.2	**0.4**
stamina-percent-enemies	**0.5**	0.0

Table 1: Percent heuristic usage for the top-ten best performing heuristic combinations for predicting difficulty and enjoyment. The top four heuristics for both are bolded. Note the tie for *proximity-to-food* and *stamina-percent-nothing* for predicting enjoyment.

We also tested for agreement in level ratings. To do this we created two columns, *agree* and *disagree*, in which each row was the sum of ratings per a level with a threshold of 4 or "neutral", where user ratings less than 4 went to *disagree* and user ratings greater than 4 went to *agree*; neutral ratings were not included in either category. For each level, we calculated the max of the two variables divided by the sum as *level agreement*. The mean of *level agreement* across all levels was calculated as the agreement of user ratings. Agreement on the difficulty of a level was 73.4%; players generally agreed on whether a level was difficult or not. Agreement for enjoyment was much higher at 87.2%.

4.2 Difficulty

The number of times a heuristic was used in the top ten heuristic combinations can be seen in Table 1. *density*, *jaccard-nothing*, and *proximity-to-enemies* were used by all ten. We assumed that path length was a good representative of difficulty [8], however, *path-no-enemies* wasn't used once and *path-nothing* only 4 out of 10 times. Instead, using Jaccard similarity is a stronger approach because it is primarily concerned with how similar the paths are. Path length doesn't capture differences if both paths are the same length. However, Jaccard similarity was only successful in the case of *jaccard-nothing*, whereas *jaccard-no-enemies* was not used.

Figure 3 shows scatter plots of the top four features—*jaccard-nothing*, *proximity-to-enemies*, *density*, and *stamina-percent-enemies*—against difficulty. *stamina-percent-enemies* is the bottom performing of these top four, and it is easy to see why: When the value is equal to zero, there is a vertical line of points from 1 to 7 for difficulty. The other three do not suffer from this as much, but it is somewhat prominent for *jaccard-nothing* and *proximity-to-enemies*. There was also some surprise that *density* played such a prominent role in

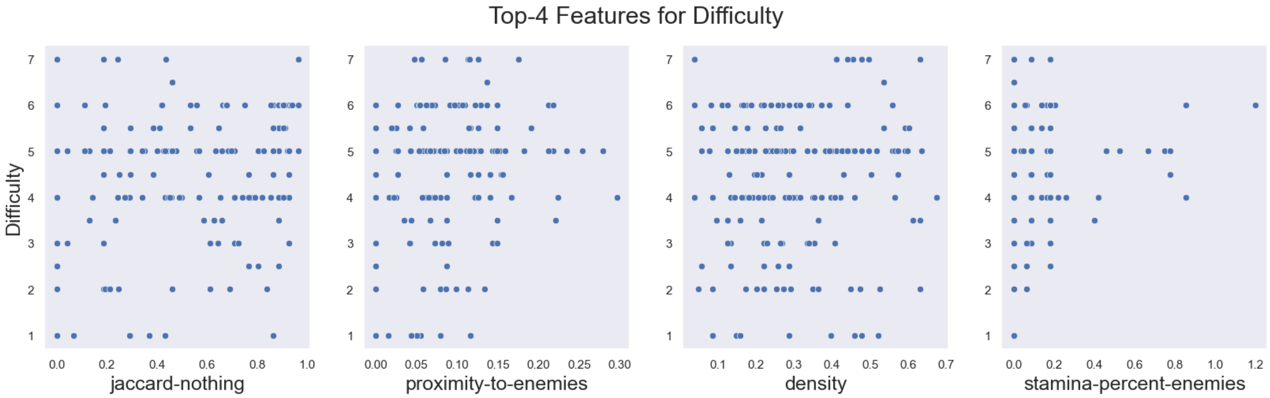

Figure 3: Top four features used to predict difficulty.

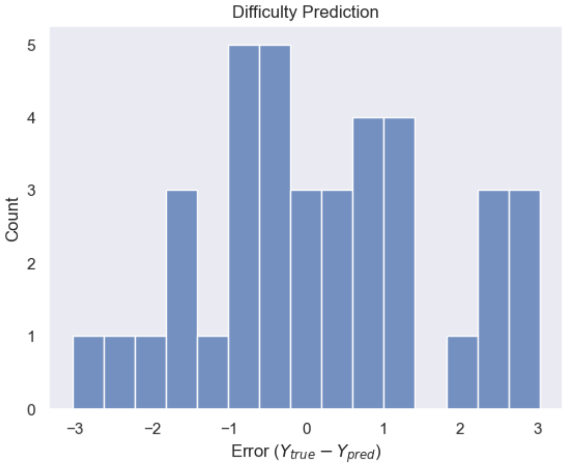

Figure 4: Difficulty prediction histogram for $Y_{true} - Y_{pred}$.

predicting difficulty, but it does reduce the number of available actions to the user and force them near enemies. Further, the graph shows that while there is noise, there are no vertical lines of points which likely improves predicting difficulty. However, each plot does not show a clear pattern and is clearly subject to noise. As a result, it is unsurprising to find that the top ten models—for difficulty prediction and enjoyment prediction—all used a combination of input heuristics rather than a single heuristic as input.

The best combination of heuristics was *jaccard-nothing*, *proximity-to-enemies*, *stamina-percent-enemies*, and *density*. Using these as input for a linear regression model, we tested the model on the test set and found that the median square error was 0.96, and the max square error was 8.68. A value of less than one for the median is a good sign because it means that the prediction was within one category of the 7-Likert scale; the model was generally close to the user's rating of difficulty.

The baseline returns the mean difficulty from the player study. When run on the test set, it had a median square error of 1.27 and a max square error of 5.80. The linear regression model had a lower median square error but a higher max square error.

The square error doesn't give a perfect picture of whether the model is generally over or underestimating difficulty. For that, we use the difference ($Y_{true} - Y_{pred}$), as seen in Figure 4, where Y_{pred} is calculated with a linear regression model trained with the best combination of heuristics and Y_{true} is the test dataset. Predictions were to the left of the column associated with zero 17 times and predictions were on the right 17 times. The error size was larger in the positive direction, meaning that difficulty was generally underestimated.

4.3 Enjoyment

Table 1 also shows the number of times a heuristic was used in the top ten heuristic combinations for predicting enjoyment. The best-performing heuristic was *proximity-to-enemies*. We expected *proximity-to-enemies* to play less of a role in predicting enjoyment and more difficulty, but it played a significant role for both. The second most used heuristic was *path-nothing*. This yields the interesting conclusion that path length was more useful in predicting enjoyment than difficulty. Also, note that *path-no-enemies* wasn't used once. There is a similar pattern where *jaccard-nothing* was a top-performing heuristic for difficulty and it is not used once for enjoyment, but *jaccard-no-enemies* is used for five out of the ten best-performing heuristic combinations for predicting enjoyment. Only two other heuristics were used to predict enjoyment: *proximity-to-food* and *stamina-percent-nothing*. We expected *proximity-to-food* to play less of a role in the prediction of enjoyment, but it's reasonable to assume that acquiring extra stamina may have given players a small sense of accomplishment and/or relief while playing.

Figure 5 shows scatter plots for the top four features against enjoyment. In the case of all four, the vertical lines noted in Section 4.2 are occurring. The least pronounced of these is with *proximity-to-enemies*. *path-nothing* is more pronounced in the vertical lines than the other three, but it is the second-best performer. *jaccard-no-enemies* displays minimal vertical lines except for when the path is unchanged—i.e. equal to 0. In this regard, *proximity-to-food* also behaves poorly when the path found to complete the level does not have nearby food.

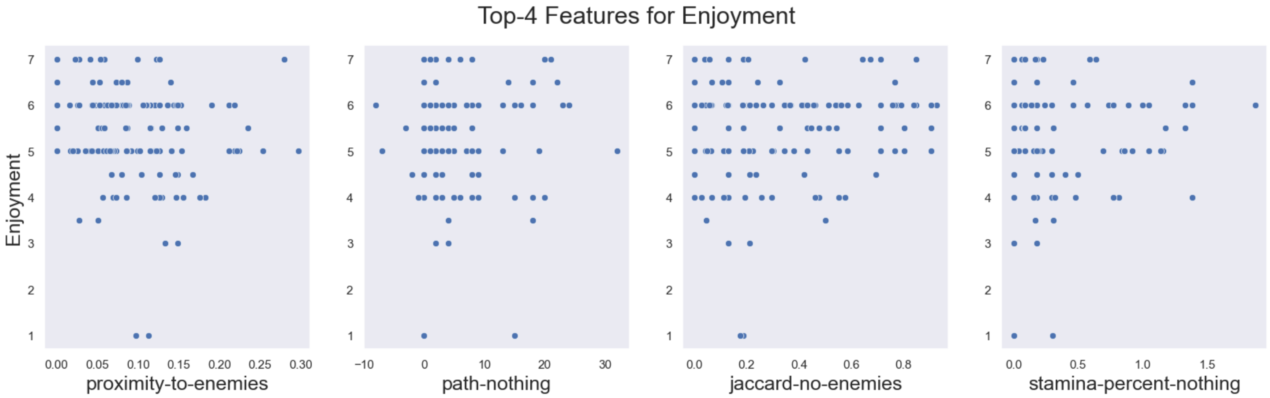

Figure 5: Top four features used to predict enjoyment.

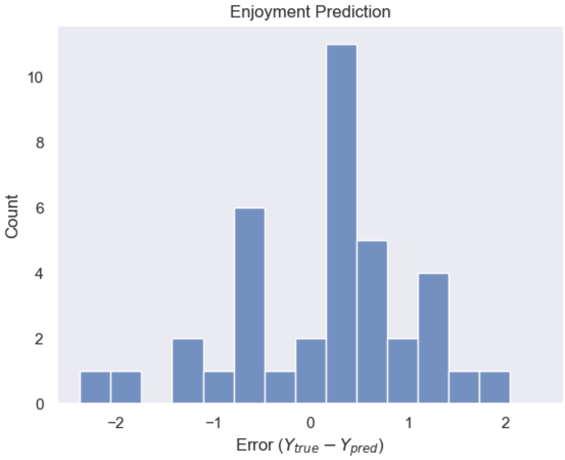

Figure 6: Enjoyment prediction histogram for $Y_{true} - Y_{pred}$.

The top-performing model used two heuristics: *path-nothing* and *proximity-to-enemies*. The second best-performing combination added *stamina-percent-nothing*. We trained a linear regression model with *path-nothing* and *proximity-to-enemies* as input. For the test set, the median square error was 0.79 and the max square error was 5.27. We were better able to predict enjoyment than difficulty, but also note that enjoyment followed a clearer distribution than difficulty, see Figure 2.

The baseline, which returns the mean enjoyment score from the player study, had a median square error of 0.70 and a max square error of 2.84. The baseline outperformed our best-performing model in both cases. Future work should identify better heuristics for predicting enjoyment.

Figure 6 shows the distribution of errors for $Y_{true} - Y_{pred}$, where Y_{pred} is calculated with a linear regression model trained with only *proximity-to-enemies* and Y_{true} is the test dataset. Again, this is useful because it gives us information on whether we are overestimating or underestimating the expected enjoyment of a level. There are 12 levels predicted to be more enjoyable than what users said and 24 levels predicted to be less enjoyable than what users said. Thus, the model generally predicts that levels are less enjoyable than they are. It is also interesting to note that there are only two levels within the bin for zero error.

4.4 Correlation between Difficulty and Enjoyment

As noted in the related work section, past work has observed a positive correlation between difficulty and enjoyment [22], but we were unable to replicate this. We ran our tests by finding the median difficulty and enjoyment for each level. We tested the correlation between difficulty and enjoyment and found that the Pearson correlation coefficient [5] was -0.083 with a p-value of 0.005, the Kendall rank correlation coefficient [1] was -0.122 with a p-value of 0.033, and the Spearman rank correlation coefficient [41] -0.194 with a p-value of 0.007. The coefficients are very close to zero, meaning there is little if any relationship between difficulty and enjoyment, and any relationship appears to be negative. Further, each coefficient showed a low p-value which indicates statistical significance in our finding with 190 total samples. This suggests that it is incorrect to assume a positive correlation between difficulty and enjoyment of roguelikes where players have to avoid enemies.

We can also examine this subjectively. Figure 7 shows four levels evaluated by participants that have been selected based on being the most or least difficult and the most or least enjoyable. Starting at the bottom left, it can be seen that the path to beat the level doesn't require any interactions with the enemies—enemies are represented by a red hashtag—to beat the level. The result is a level that requires very little thought to complete.

Going to the right to min difficulty and max enjoyment, we can see that the player is required to get food—the green '&'— to have enough stamina to beat the level, and there is also a switch—the blue '*'—which the player must come in contact with to complete the level. These additional factors contribute to the experience, but we think the most interesting part of the level can be seen by looking at the path. In the middle, the path passes two enemies, but notice that the agent does not move to avoid the enemies. The agent moves as if the enemies aren't there. We believe that this realization led to a more satisfying solution.

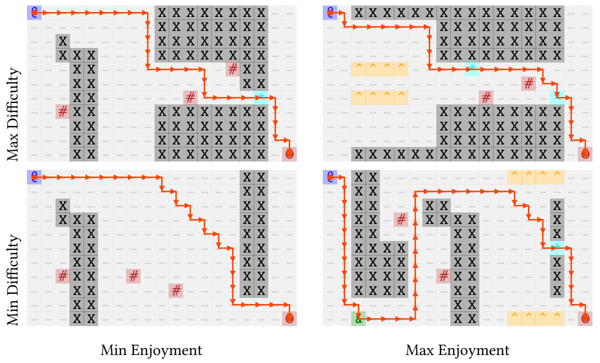

Figure 7: Example levels for min and max of difficulty and enjoyment based on mean ratings from study participants. Red arrows show path found by the A* agent to complete the level.

Changing to max difficulty and minimum enjoyment, it is clear that the section with two enemies and one pathway to get to the switch is a difficult challenge. It may be that the lack of options on how to solve the movement puzzle resulted in low enjoyment and high difficulty.

Finally, the maximum of both difficulty and enjoyment features a very similar level to the one we just examined, but it has more open space, which lends some credence to the idea that difficulty comes at the cost of fun when player options are reduced.

5 CONCLUSION

In this work, we ran a player study with 143 participants to evaluate *DungeonGrams* levels in terms of difficulty and enjoyment on a 7-Likert scale. We found that most players found it enjoyable to play levels in *DungeonGrams*, but the results were more diverse in terms of difficulty. We found that there was not a correlation between difficulty and enjoyment for *DungeonGrams*. We used 9 heuristics from past work and 2 new ones with Jaccard similarity. These were used as input into a model to predict how difficult and enjoyable a level would be for the average player. We found that heuristics that incorporate solution path information — beyond just lengths — were useful in predictions.

Besides using manually built heuristics, there is another approach that appears promising for future work. Rather than define heuristics ourselves, heuristics can be found automatically [4, 19]. Our biases for what we think is difficult and for what we think is enjoyable for a level won't affect the generated heuristics. This in turn may yield surprising results, which could cause us to reconsider what makes a roguelike level difficult and enjoyable.

To test which manually built heuristics were most useful, we ran a complete ablation study where all possible combinations of feature inputs were tested with a linear regression model. The best combination of heuristics to predict difficulty was *jaccard-nothing*, *proximity-to-enemies*, *stamina-percent-enemies*, and *density*. The best combination to predict enjoyment was *path-nothing* and *proximity-to-enemies*. Of note, we found that *proximity-to-enemies* was used by both sets of top ten models to predict difficulty and enjoyment. Further, each top-performing model predicted difficulty and enjoyment within one point on the Likert scale based on the median square error. However, our model for predicting enjoyment did not beat our baseline approach. Future work should find new heuristics to better estimate enjoyment.

A limitation is our use of linear regression, which has an unbounded output on a problem bounded between 1 and 7. One way to address this is to convert player scores to a percentage. With the expected output bounded between 0 and 1, Beta regression [6], which can model rates and proportions, could improve prediction. Another approach is to move away from regression and instead view the problem as a classification problem with 7 classes of difficulty.

The player study did not take into account the player's learning curve [2] outside of a single tutorial level. It is easily understood that the longer a player plays a game, the more they improve [23, 34], at least up to a certain point. This could be addressed in future work by adding tutorial levels or weighting the player's ranking based on the number of levels they've played.

REFERENCES

[1] Hervé Abdi. 2007. The Kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA* (2007), 508–510.

[2] Sarit Barzilai and Ina Blau. 2014. Scaffolding game-based learning: Impact on learning achievements, perceived learning, and game experiences. *Computers & Education* 70 (2014), 65–79.

[3] Colan Biemer, Alejandro Hervella, and Seth Cooper. 2021. Gram-Elites: N-Gram Based Quality-Diversity Search. In *Proceedings of the FDG workshop on Procedural Content Generation*. 1–6.

[4] Edmund K Burke, Matthew R Hyde, Graham Kendall, and John Woodward. 2007. Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 1559–1565.

[5] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. *Noise reduction in speech processing* (2009), 1–4.

[6] Silvia Ferrari and Francisco Cribari-Neto. 2004. Beta regression for modelling rates and proportions. *Journal of applied statistics* 31, 7 (2004), 799–815.

[7] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2014. Clingo= ASP+ control: Preliminary report. *arXiv preprint arXiv:1405.3694* (2014).

[8] Alexander Gellel and Penny Sweetser. 2020. A hybrid approach to procedural generation of roguelike video game levels. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*. 1–10.

[9] Mark E Glickman. 2012. Example of the Glicko-2 system. *Boston University* 28 (2012).

[10] Miguel González-Duque, Rasmus Berg Palm, David Ha, and Sebastian Risi. 2020. Finding game levels with the right difficulty in a few trials through intelligent trial-and-error. In *2020 IEEE Conference on Games (CoG)*. IEEE, 503–510.

[11] Miguel Gonzalez-Duque, Rasmus Berg Palm, and Sebastian Risi. 2021. Fast game content adaptation through Bayesian-based player modelling. In *2021 IEEE Conference on Games (CoG)*. IEEE, 01–08.

[12] Matej Guid and Ivan Bratko. 2013. Search-based estimation of problem difficulty for humans. In *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings 16*. Springer, 860–863.

[13] Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[14] Dayana Hristova, Matej Guid, and Ivan Bratko. 2014. Assessing the difficulty of chess tactical problems. *International Journal on Advances in Intelligent Systems* 7, 3 (2014), 728–738.

[15] Tobias Huber, Silvan Mertes, Stanislava Rangelova, Simon Flutura, and Elisabeth André. 2021. Dynamic Difficulty Adjustment in Virtual Reality Exergames through Experience-driven Procedural Content Generation. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1–8.

[16] Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 429–433.

[17] Paul Jaccard. 1908. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.* 44 (1908), 223–270.

[18] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. 2010. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the*

2010 Workshop on Procedural Content Generation in Games. 1–4.

[19] Kalev Kask and Rina Dechter. 2001. A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence* 129, 1-2 (2001), 91–131.

[20] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 16. 95–101.

[21] Julian Mariño, Willian Reis, and Levi Lelis. 2015. An empirical evaluation of evaluation metrics of procedurally generated Mario levels. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 11. 44–50.

[22] Willian M. P. Reis, Levi H. S. Lelis, and Ya'akov Kobi Gal. 2015. Human computation for procedural content generation in platform games. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. 99–106. https://doi.org/10.1109/CIG.2015.7317906

[23] Charles Reynaldo, Ryan Christian, Hansel Hosea, and Alexander AS Gunawan. 2021. Using video games to improve capabilities in decision making and cognitive skill: A literature review. *Procedia Computer Science* 179 (2021), 211–221.

[24] Stuart Russell and Peter Norvig. 2009. *Artificial intelligence: a modern approach* (3rd edition ed.). Pearson, Upper Saddle River.

[25] Pedro Sampaio, Augusto Baffa, Bruno Feijó, and Mauricio Lana. 2017. A fast approach for automatic generation of populated maps with seed and difficulty control. In *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 10–18.

[26] Anurag Sarkar and Seth Cooper. 2019. Transforming game difficulty curves using function composition. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–7.

[27] Anurag Sarkar and Seth Cooper. 2021. Procedural Content Generation using Behavior Trees (PCGBT). *arXiv preprint arXiv:2107.06638* (2021).

[28] Noor Shaker, Miguel Nicolau, Georgios N Yannakakis, Julian Togelius, and Michael O'neill. 2012. Evolving levels for super mario bros using grammatical evolution. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 304–311.

[29] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games*. Springer.

[30] Tianye Shu, Jialin Liu, and Georgios N Yannakakis. 2021. Experience-driven PCG via reinforcement learning: A Super Mario Bros study. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–9.

[31] Gillian Smith and Jim Whitehead. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. 1–7.

[32] Gail M Sullivan and Anthony R Artino Jr. 2013. Analyzing and interpreting data from Likert-type scales. *Journal of graduate medical education* 5, 4 (2013), 541–542.

[33] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270.

[34] Ryo Takaoka, Masayuki Shimokawa, and Toshio Okamoto. 2011. A Framework of Educational Control in Game-based Learning Environment. In *2011 IEEE 11th International Conference on Advanced Learning Technologies*. 32–36. https://doi.org/10.1109/ICALT.2011.18

[35] Jonathan Tremblay, Pedro Andrade Torres, and Clark Verbrugge. 2014. Measuring risk in stealth games.. In *FDG*. Citeseer.

[36] Marc van Kreveld, Maarten Löffler, and Paul Mutser. 2015. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. 415–422. https://doi.org/10.1109/CIG.2015.7317913

[37] Michael Weeks and Jonathon Davis. 2022. Procedural dungeon generation for a 2D top-down game. In *Proceedings of the 2022 ACM Southeast Conference*. 60–66.

[38] H Wong et al. 2012. Rating logic puzzle difficulty automatically in a human perspective. In *Proceedings of DiGRA Nordic 2012 Conference: Local and Global–Games in Culture and Society*.

[39] Georgios N. Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. 2013. Player Modeling. In *Artificial and Computational Intelligence in Games*.

[40] Georgios N Yannakakis and Julian Togelius. 2018. *Artificial intelligence and games*. Vol. 2. Springer.

[41] Jerrold H Zar. 2005. Spearman rank correlation. *Encyclopedia of biostatistics* 7 (2005).

[42] Mohammad Zohaib. 2018. Dynamic difficulty adjustment (DDA) in computer games: A review. *Advances in Human-Computer Interaction* 2018 (2018).