Proceedings of the Twentieth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2024)

No Player Left Behind: Evolving Dungeons and Dragons Combat to Optimize Difficulty and Player Contributions

Fiona Shyne, Seth Cooper

Khoury College of Computer Sciences, Northeastern University
shyne.f@northeastern.edu, se.cooper@northeastern.edu

Abstract

Creating well-balanced combat encounters can be a difficult task for Game Managers (GMs) in tabletop games such as Dungeons and Dragons (DnD). This work uses a simulation environment to generate new sets of DnD encounters that can be optimized for both difficulty and balance among player contributions. Encounters are evaluated using simulated games that can either be run probabilistically (using dice rolls) or with deterministic expected outcomes. While the expected approach allows game outcomes to be simulated substantially faster and is a good estimate of difficulty, it is a less reliable measure of balance. A genetic algorithm was used to generate encounters that meet the desired difficulty and where all players are needed for success.

Introduction

In Table Top Role-Playing Games (TTRPGs), game content is often tailored to meet the needs of one set of players. The responsibility for generating this content falls on one special player, the Game Master or Game Manager (GM), who runs the game for the other players. Dungeons and Dragons (DnD) (Crawford et al. 2014) is perhaps the most popular TTRPG system and has a large, complex set of rules. This means GMs must consider a wide set of variables when designing content, a process made even more challenging without the resources for playtesting available to game designers with wider audiences. Procedural content generation (PCG) has the potential to aid GMs in developing personalized game content.

This work looks at combat in DnD (particularly the 5th edition) and proposes a tool to generate a set of combat encounters (the combat day) given a list of player characters (PCs), the desired difficulty, and a set of possible monsters provided by a GM. A multi-objective genetic algorithm is used to optimize a set of encounters that both consider the desired difficulty across the combat day (difficulty fitness) and ensure that all players contribute to the fight (balance fitness). It does this by minimizing the difference in simulated and desired difficulty and maximizing the effect of removing each player from combat (with both functions calculated as a minimization function).

To reduce fitness evaluation time, we introduce a deterministic simulation method that uses the expected outcomes of each action made in the game. This method was demonstrated to be substantially faster than probabilistic simulation while fairly reliably predicting a similar difficulty fitness. However, it was less reliable in producing encounters with a balance of player contributions.

This work contributes both a methodology for reducing evaluation time for simulated games, and a genetic algorithm that simultaneously optimizes the desired difficulty of encounters and the contributions of players. We have found partial support for the usefulness of deterministic simulation based on expected values, particularly for estimating difficulty. Our genetic algorithm was able to find encounters that were fit in both difficulty and balance. However, we found that using balance fitness alone was just as good as multi-objective evolution at generating optimally difficult sets of encounters. This is due to how the balance fitness function, in itself, is related to optimal difficulty.

Related Work

Automated Playtesting

There are many instances where human evaluation of games is not feasible, particularly in procedural content generation (PCG) systems. Albaghajati and Ahmed (2020) found a variety of automated testing techniques for games, including agent-based playtesting methods designed to either achieve a goal or to behave like a human. These automated playtests can have direct implications for game design, such as in (Zook, Fruchter, and Riedl 2014), where automatic playtesting was used to tune game parameters. Shyne (2023) provides a suite of agents to automatically playtest level 1 DnD combat encounters, which we build upon in this work.

It is often important to understand how different players would respond to game environments. Holmgård et al. (2018) proposed the use of procedural personas, where Monte Carlo Search Tree agents were used with varying goal functions. This has been applied to genres such as match three games (Mugrai et al. 2019) and dungeons (Liapis et al. 2015). While these look at single-player games, Lang and Young (2023) models how groups of players with differing desires might plan routes through games. Jaffe et al. (2012) uses automated playtesting to evaluate the power of cards in

an educational game by looking at the impact of removing them. This work expands on previous work to include personas for common DnD archetypes and examine the impact each player character has on the outcome of the encounter.

TTRPG Procedural Content Generation

Guzdial et al. (2020) proposed that TTRPGs themselves are procedural content generators, and we can apply techniques used in TTRPG system design to guide the design of PCG systems. However, the amount of effort required by the DM to create content for games has led to the need to develop automated tools for TTRPGs. Previous works that interviewed GMs about computational tools (Tang et al. 2023; Acharya, Mateas, and Wardrip-Fruin 2021), have demonstrated the desire from GMs for computational tools but the need for customization and controllability.

The amount of textual content in TTRPGs has made natural language approaches, and particularly large language models (LLMs), popular tools for PCG. Several works have investigated the use of LLMs as GM assistants. Zhu et al. (2023) evaluated the ability of LLMs to summarize combat information and brainstorm. Acharya et al. (2023) visualized story information in the TTRPG system Gumshoe, which they later expanded to use LLMs to suggest dialog lines (Kelly, Mateas, and Wardrip-Fruin 2023). LLMs have also been proposed as a method to generate new content; for example, Newman and Liu (2022) generated new DnD spells. However, these tools are not guaranteed to produce content that abides by the mechanics of the TTRPG system, and controllability in LLMs is still an open problem. There are also several ethical concerns to consider with LLM-based tools, such as the high environmental cost (Wu et al. 2022; Everman et al. 2023) along with the sourcing and content of training data (Jiao et al. 2024).

Other techniques for generating TTRPG content have been proposed as well. For example, Crain, Carpenter, and Martens (2022) described a mixed-initiative tool for designing maps using Answer Set Programing. Horswill (2019) designed a simplified logical programming language that allows TTRPG players to define ways to generate game elements, such as characters. Similarly, Faria, Pereira, and Toledo (2019) used a genetic algorithm (GA) to generate characters that might suit the user's tastes. Several works explore generating dungeons using GAs (Alvarez et al. 2018; Liapis 2017; Viana, Pereira, and Toledo 2022) or generative grammars (Dormans 2010). Liapis et al. (2015) used a suite of procedural personas as an evaluator for generating dungeons with a GA. While combat in TTRPGs has been less explored, Guth Jarkovskỳ (2023) looked at generating enemies for a rogue-like game using simulated encounters and Popa (2023) generated NPCs for Pokemon using generated grammars. This work is aimed at combat in TTRPGs, where multiple players work together to defeat a set of monsters in a turn-based combat system.

Surrogate Models for Genetic Algorithms

A major limiting factor of GA approaches is time and resource-costly fitness functions. There are many approaches to reduce the time required to evaluate fitness, including problem simplification, fitness approximation, and balancing high-fidelity and low-fidelity models (Li, Zhan, and Zhang 2022). Often, fitness approximation is done by learning a surrogate model from previous actual fitness calculations. Surrogate models have been used to reduce the number of gameplay simulations including for generating Hearthstone decks (Zhang et al. 2022) and Angry Birds levels (Pereira et al. 2016). This work tackles this problem instead by using the known properties of the game to deterministically simulate games using expected outcomes.

System Overview

We developed a system where a GM can input the PCs in the party, the set of monsters to draw from, and the desired difficulty, which are used to generate a set of encounters (one combat "day") that matches the desired difficulty and requires all PCs.

Fitness type	Encounter	Monsters (Challenge)
Difficulty Only	1	dust mephit $(1/2)$
	2	camel $(1/8)$ camel $(1/8)$ acolyte $(1/4)$ swarm of bats $(1/4)$
	3	reef shark $(1/2)$ dretch$(1/4)$ swarm of quippers(1) grimlock $(1/4)$ reef shark $(1/2)$
Balance Only	1	poisonous snake $(1/8)$ giant sea horse $(1/2)$ drow $(1/4)$
	2	thug $(1/2)$
	3	noble $(1/8)$ giant frog: $(1/4)$
Multi-objective	1	ape $(1/2)$
	2	death dog (1)
	3	boar $(1/4)$ sahuagin $(1/2)$ imp (1)

Table 1: The most fit combat days evolved using difficulty only, balance only, and both difficulty and balance (multi-objective). Days were selected using finesses evaluated by probabilistic simulation with 50 playthroughs.

Combat Simulation

Dungeons and Dragons (DnD) has a turn-based combat system where a party made up of player characters (PCs) controlled by individual players fights against a set of monsters controlled by the Game Manager (GM). Combat starts with all creatures (both PCs and monsters) rolling dice to determine the order of combat (called rolling initiative). Then, in order, each creature makes a move on a grid and takes one

or more actions, as determined by their individual abilities and resources. The end state is often determined by the GM, but this work assumes combat ends when either all monsters or all PCs are dead.

Shyne (2023) built a combat simulation environment for level 1 combat in DnD, which we expand on this work. This environment provides a set of monsters (based on the official DnD "Monster Manual" (Mearls and Crawford 2014)), player characters, and a suite of agents. Encounters can be simulated by providing a list of monsters, player characters, and an agent class. This simulation environment is a simplified version of DnD but attempts to provide an estimate for how real encounters would be played.

Our work expands this environment by adding the following: the ability to simulate multiple encounters in one combat day, new agents based on player character personas, and the ability to estimate games using expected values of dice rolls.

Existing Environment The existing simulation environment attempts to approximate low-level DnD combat. Several simplifications were made to save on computational resources. In this environment, users can define creatures with several stats such as speed (how many grid spaces they can move in a turn), maximum health (hit points or HP, the maximum damage a creature can sustain), and armor class (AC, how vulnerable to damage a creature is). Creatures are also defined by a set of actions they can execute in a turn. Actions include weapon attacks that damage enemies and spells that can harm enemies, heal allies, or have other effects. Creatures can also have special abilities, such as creatures with pack tactics that get damage bonuses when near allies. During simulated combat, creatures take turns moving on a 2d grid and executing actions, as determined by their agent class.

Shyne (2023) evaluated this simulation environment against the difficulty prediction method described in the base rule set in DnD. This prediction is based on the number and Challenge Rating (CR, static predictor of difficulty) of the monsters in the encounter, compared to the number and level of PCs in the party. The previous work conducted a study of experienced GMs, and had them compare two encounters with the same predicted difficulty using CRs but different difficulties using the simulation environment. They found that GMs noted a difference in difficulty between these two encounters, in line with the prediction of the simulation environment.

Multi-Encounter Days The original simulation environment looked at combat encounters in isolation, assuming PCs would be at full health and ability at the beginning of each encounter. However, players often have multiple encounters in one in-game day and do not completely restore their health in between them. To acknowledge this we added the ability to play encounters back to back without restoring health or resources in-between. Future work could further expand this to represent "short rests" where players can partially restore resources in small periods of downtime.

Personas The original simulation environment provides a suite of agents meant to be used by all creatures (monsters and PCs) in combat. We expand this agent suite to include agents representing the major personas of play styles and creature types. We sought to represent four personas (agents that represent archetypal playstyles), which required the development of three new agents. The aggressive persona, which was the highest preforming agent in the original environment, is the default for all creatures, but PCs can be specified to have different personas.

- **Aggressive**: This persona was taken from Shyne (2023). This persona gets as close to enemies as possible and attacks with the largest (most damage) attack. This persona is representative of close-range creatures, such as fighters and barbarians.
- **Ranger**: This persona attempts to attack enemies from as far away as possible. When not close enough to attack with a weapon, it will move towards the closest enemy. Otherwise, it will attack from the maximum distance away from enemies with the largest attack available. This is representative of long-range attackers and spell casters, such as rangers and wizards.
- **Healer**: This persona will get as close as possible and cast the highest (most health) healing spell on the most injured ally. If there are no allies that are injured, it will behave like a ranger. This is representative of healing creatures, such as clerics.
- **Lonely**: This persona will get as close to an ally as possible, and attack with their largest attack. This represents rogues who get a bonus ("sneak attack") whenever they are next to an ally, but it can also represent players who want to stick close to other party members.

Expected Values The outcome of combat in DnD is extremely variable based on random factors. This is particularly true for first-level encounters, where there is a low probability of hitting creatures and attack damage is often a large percentage of a creature's total health (especially for PCs). This means it can take a large number of simulated games to gain an understanding of the encounter.

To attempt to reduce the amount of time to evaluate encounters, we add a mode (called the "expected simulation") to simulate games deterministically. Instead of hit and damage dice being rolled for each attack, the target is damaged by the estimated value of the damage roll multiplied by the probability of a successful attack. For example, an attack could need a 1d20+4 [1] roll to beat 14 for the attack to be successful. The attacker deals 2d4 damage to their target on a successful hit. The probability of a 1d20+4 roll being higher than 14 is 50%, and the expected value of a 2d4 roll is 5. Therefore the expected damage of this attack is 2.5, which would be dealt to the target. In the expected mode, side effects (such as knocking a target prone) are ignored, and combat order is determined by sorting the expected value of each creature's initiative roll.

[1]Dice rolls are described using 'dice notation' which is common for wargames and TTRPGs. 1d20+4 means a 20-sided dice is rolled once with 4 added at the end.

While probabilistic simulation is a more realistic estimation of real combat encounters, expected simulation reduces the number of playthroughs needed to 1. This can dramatically reduce the total time needed to conduct playtests. Expected simulation is limited by the requirement to have deterministic agents, such as the personas described here.

Genetic Algorithm

We further build on previous work by implementing a method to generate new combat encounters based on the desires of a GM. The GM can specify the PCs in the combat, the monsters to select from, and the number of encounters to include in the day. Encounters are then generated based on two fitness functions: the value each PC added to the encounter (balance), and the difference from the desired success of the party at each encounter (difficulty). Both of these fitnesses are represented as a minimization function. A sample of the combat days generated is shown in Table 1.

Representation An individual day is made up of E encounters, where E is given by the GM. Each encounter is represented as a list of 1 to 10 monsters, from a monster set provided by the GM. The PCs and monsters are initially placed in the combat grid such that the PCs are in a line on the upper left side and monsters are placed in a line on the bottom right side, with order being arbitrary. The following genetic operators are applied:

- **Initialization**: Each encounter is initialized by selecting between 1 and 10 monsters randomly from the provided set.
- **Mutation**: First a random encounter in the day is selected. Then a random monster is added or removed. Whether to add or remove is decided randomly at equal rates, unless adding would result in more than 10 monsters or removing would result in no monsters. This process is repeated 5 times per mutation.
- **Cross-Over**: A random encounter time (in terms of the relative order of encounters in the day) is decided, and the encounters at that time are swapped between the two individuals.

Difficulty Fitness The GM can specify the desired difficulty for each encounter in the day by providing E values that represent the desired health of the party at the end of each encounter. Health was chosen for the fitness function as is more variable than other stats, such as PC death, but is correlated with these stats. This health is normalized as the sum of the party members' health divided by the sum of their maximum health. The difficulty function is then determined by the root of the average squared error between every simulated encounter and the desired difficulty for that encounter. This can be calculated with the following formula:

$$\sqrt{(\sum_{e=1}^{E} \frac{\sum_{n=1}^{N}(d_e - s_{e,n})^2}{N})/E}$$

Where E is the number of encounters, N is the number of simulation playthroughs, d_e is the desired difficulty for the eth encounter, and $s_{e,n}$ is the ending health of the party at the eth encounter during the nth playthrough.

Fitness type	Simulation Playthroughs	Average Time in Seconds
Difficulty	expected	0.04
	4	0.15
	10	0.40
	50	1.82
	100	4.16
Balance	expected	0.16
	4	0.72
	10	1.88
	50	8.50
	100	16.10

Table 2: The average time taken per fitness call for both Difficulty and Balance by simulation type and number of playthroughs. The average is taken from all fitness calls throughout evolution

Balance Fitness The balance fitness function attempts to increase the value each PC in the party (of size P) adds throughout the combat day. To accomplish this, the combat days are simulated regularly with all PCs (T) and then again after removing each PC in turn ($T - p_i$).

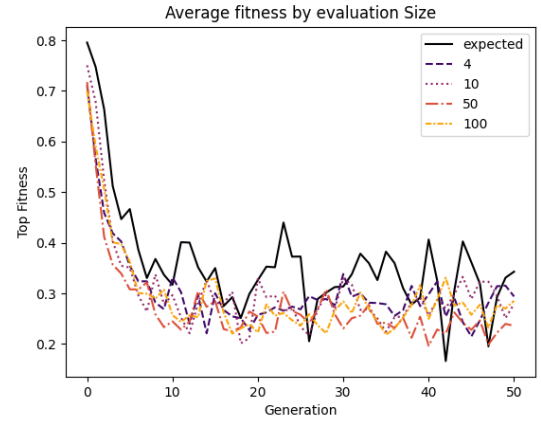
The value of a single PC (p_i) is calculated by how much removing them from the party reduced the average health of the party at the end of the last encounter (at the end of the combat day). This is calculated as the difference between the total ending health and the ending health after removing p_i from the party. In probabilistic simulation, both T and each $T - p_i$ are simulated N times, and the difference becomes the average between each pair of values in T and $T - p_i$ (N^2 total pairs). For the expected simulation, there is only one difference value per PC and no average is taken.

Balance fitness incorporates the average of these individual PC values, and further, to ensure no party member is dominant, their standard deviation. However, standard deviation is weighted less the average. Overall the balance function can be calculated with the following minimization function.

$$0.7(1 - D_{avg}) + 0.3(D_{sd})$$

Where $D = [d_1, d_2, ...d_P]$ where d_i is the average health difference between the pairs in T and $T - p_i$. T is the list of N ending healths with the total party and $T - p_i$ is the list of N ending healths removing p_i from the party.

Multi-Objective Fitness To optimize for both difficulty and balance, we calculate the fitness for both and use non-dominated sorting, similar to Deb et al. (2000). This method uses the concept of domination, which means the individual is at least as good across all fitnesses and better (smaller) in at least one dimension. For example, if we look at a problem



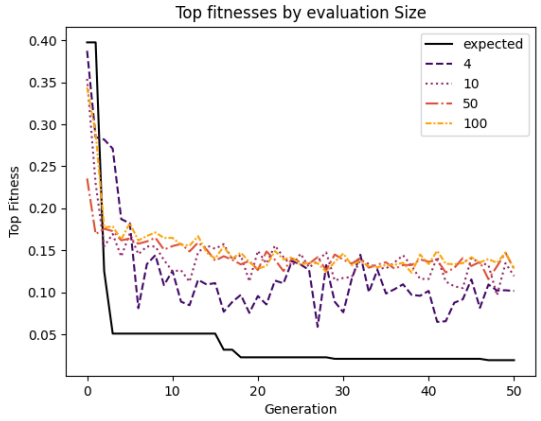


Figure 1: The evaluated fitness over the generation selecting for difficulty, with varying simulation sizes. The top graph shows the average fitness of the population and the bottom graph shows the minimum (best) fitness.

with two fitness values (f_1, f_2), the individual with fitness $(1, 0)$ would dominate an individual with fitness $(1, 1)$, as neither fitness in the second individual is smaller. However, it would not dominate an individual fitness $(0.5, 0.5)$, as the first fitness in the second individual is smaller. The Pareto-Optimal Front, or the set of 'non-inferior' solutions, is the set of individuals in a population that are not dominated by any other individual. In non-dominated sorting the population is first sorted by their fronts, where the first front is the Pareto Optimal Front and the $i + 1$th is the Pareto Optimal Front if the $1 - i$th fronts are removed. Figure 2 illustrates these fronts, where the darkest set of points is the Pareto-Optimal Front, and the lighter colors show higher fronts. The population is then sorted by crowding distance (distance to similarly fit individuals) within each front. The GM can then select from the Pareto Optimal Front to choose their desired trade-off between balance and difficulty.

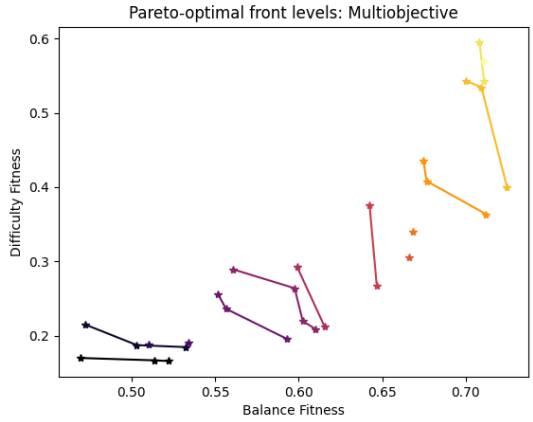


Figure 2: All individuals in the last population of the multi-objective GA run, connected by their fronts. The darker colors represent smaller fronts.

Experiment 1: Single Objective Expected VS Trial Evaluation

The first experiment looks at how closely the fitness of the expected mode simulation compares to probabilistic simulation with different numbers of trials. To do this we ran single-objective GAs with expected and probabilistic simulation for both the difficulty and balance fitness.

Set Up

We ran the GA with a representative party of 4 PCs: a Healer (dwarf cleric), a Ranger (elf wizard), a Lonely (halfling rogue), and an Aggressive (human fighter) [2]. The list of monsters provided is all 109 monsters from the DnD monster manual having challenge ratings (static challenge estimates) between 0 and 1 (some monsters have features that cannot be represented in this simulation environment). All monsters and PCs were provided in the original combat simulation environment. We test this set up with all available monsters, but it is likely that GMs would choose some subset based on the circumstances of the encounter.

The number of encounters (E) was set to 3, each with increasing desired difficulty represented by party health of 1, 0.9, and 0.6 at the end of each encounter respectively. For each run, the GA was run for 50 generations with a population size of 20, 5 elites being saved each generation, a mutation rate of 0.8, and a cross-over rate of 0.6. The GA was run 5 times for each fitness function, once with expected simulation and 4 times with probabilistic simulation at 4, 10, 50, and 100 playthroughs.

After the GA runs had terminated, we evaluated the fitness of the top individuals at generation 50 for each run using probabilistic simulation at 100 trials, which we use as our 'ground truth' method of evaluation. We also evaluate 20 random combat days as a baseline for understanding fitness.

[2]The DnD race and class of PCs are provided as reference

We compare this with the average clock time per fitness call in seconds.

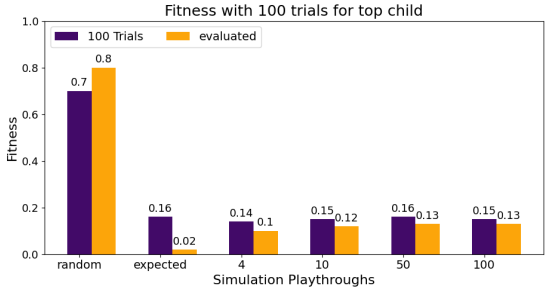


Figure 3: Difficulty Fitness evaluated with probabilistic simulation at 100 trials vs fitness evaluated during evolution (either expected or probabilistic based on GA run). The random column shows the average fitness of 20 randomly generated encounters using both expected simulation simulation and probabilistic simulation over 100 playthroughs.

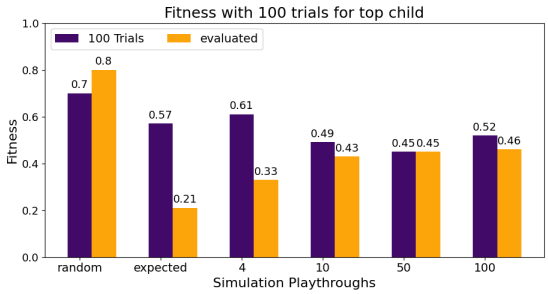


Figure 4: Balance Fitness evaluated with probabilistic simulation at 100 trials vs fitness evaluated during evolution (either expected or probabilistic based on GA run).

Results

Difficulty fitness over evolution is shown in Figure 1. The run using expected simulation was the only run that had strictly non-increasing fitness, as the expected simulation fitness of an individual does not change over multiple evaluations, unlike probabilistic simulation. All runs had a general decreasing pattern over the generations, with playthrough sizes 50 and 100 having the most stable downward trend of the probabilistic simulations runs.

Figure 3 shows the difficulty fitness calculated using probabilistic simulation with 100 playthroughs (the "ground truth" fitness) of the top individual after 50 generations. While the expected simulation fitness was furthest from the ground truth fitness, the resulting fitness was nearly as good as the other trials. Further, the expected simulation fitness function took 0.04 seconds on average compared to the 4.16 seconds needed to calculate fitness with 100 playthroughs, as shown in Table 2.

This process was repeated for the balance fitness as shown in Figure 4. We see a similar pattern: the expected simulation fitness is smaller than ground truth fitness. However,

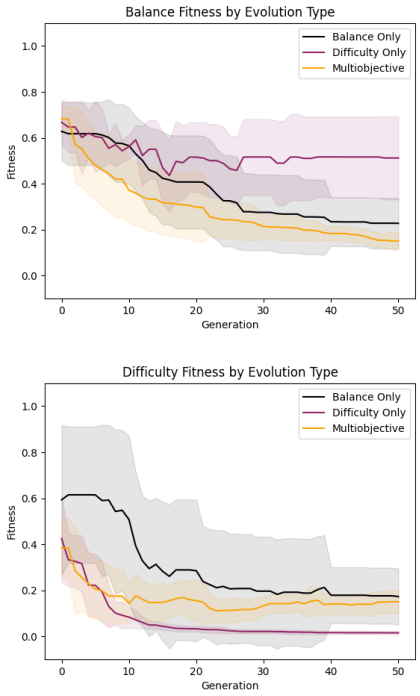


Figure 5: Balance (top) and Difficulty (bottom) fitness over generations by evaluation type. Shaded areas represent standard deviation among trials.

the top ground truth fitness for the encounter evolved using the expected simulation was higher than those that evolved using probabilistic simulation. This shows that the expected simulation is less capable of producing fit individuals for the balance function. That being said, the speed-up is even more pronounced for the balance fitness, as shown in Table 2, due to the extra simulations required by the balance fitness. The top fitness across all runs was also much higher than for difficulty, suggesting balance could be harder to optimize for than difficulty.

Experiment 2: Multi-Objective Evolution

We test the ability of our GA to generate encounters that are both appropriately difficult and rely on all the PCs in the party for success. To do this, we ran the GA using three fitness functions: balance only, difficulty only, and both balance and difficulty (multi-objective). While single-objective GAs were run in the previous experiment, they were run again with larger population sizes in this experiment.

Set Up

These runs used the same party described in the previous section (a healer, a ranger, a lonely, and an aggressive) along with the desired difficulties (1, 0.9, 0.6). For each of the three fitness functions (balance, difficulty, multi-objective) the GA was run for 50 generations with a population size of 30 and elitism of 10. This was run first using expected simulation with 10 trials each, and then once using probabilis-

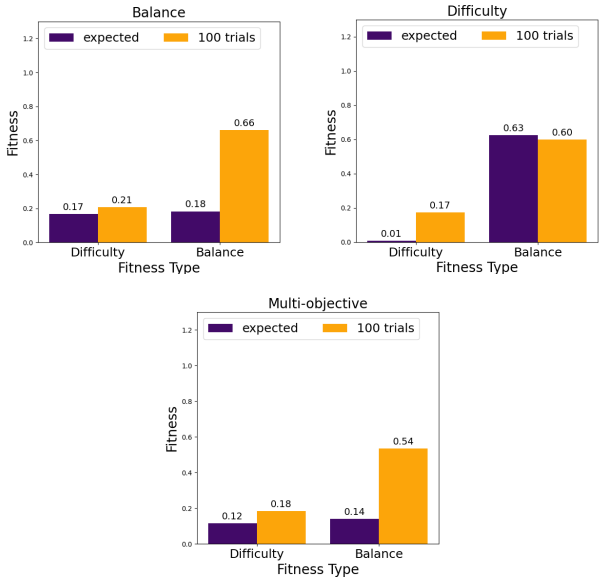


Figure 6: Expected fitness estimations against probabilistic fitness at 100 playthroughs.

tic simulation at 50 playthroughs. Cross-over and mutation were kept the same as in the previous section (0.6 and 0.8 respectively).

Expected Simulation Results

The fitness over generations is shown in Figure 5. The GA appeared to find highly fit individuals across all evaluation types. However, when compared to the 'ground truth' evaluation (probabilistic simulation with 100 playthroughs), the expected balance fitnesses for the top balance only and multiobjective day, differed substantially (see Figure 6). This again demonstrates that expected simulation is less reliable for the balance function than the difficulty function.

Probabilistic Simulation Results

The fitness of the top individual across generations is shown in Figure 7. The balance-only GA struggled to produce fit individuals for several (˜20) generations. This was suspected to be mostly due to chance, which we confirmed in a second trial as shown in the appendix Figure 11. In both trials, the balance-only GA was able to produce individuals nearly as fit as the multi-objective GA for both balance and difficulty, despite ignoring the difficulty fitness. However, in the difficulty-only GA, balance fitness did not decrease over evolution. The difficulty-only run also produced the most fit individuals in terms of difficulty. The fitnesses of the Pareto Optimal Fronts for the final populations is shown in Figure 8. All the front levels for the final multi-objective population are shown in Figure 2. This shows how generally the multi-objective fitness finds a mixture between difficulty and balance, where the balance-only has slightly higher difficulty fitness and difficulty-only has substantially higher balance fitness.

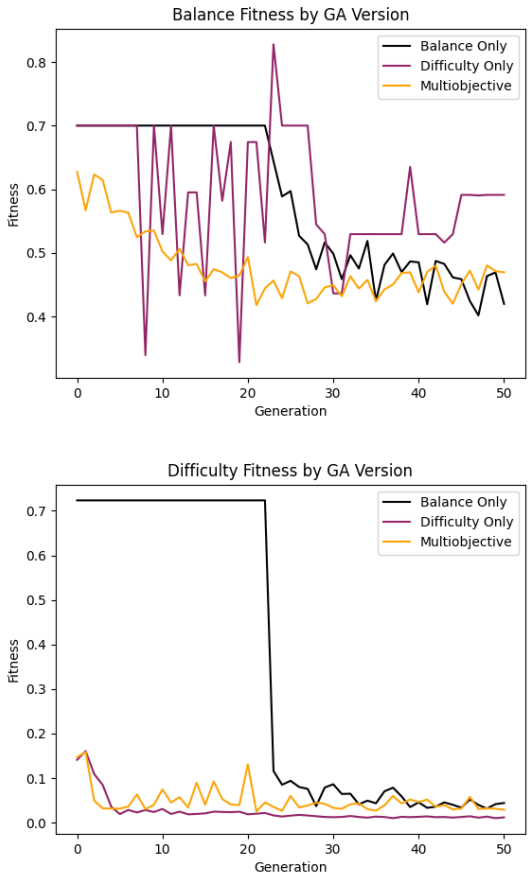


Figure 7: Balance (top) and Difficulty (bottom) fitness of most fit individuals by selection type.

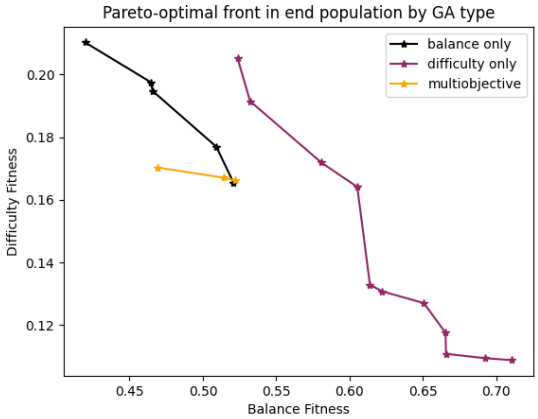


Figure 8: Pareto Optimal Fronts for the final population for the GA evolving using difficulty only, balance only, or both balance and difficulty.

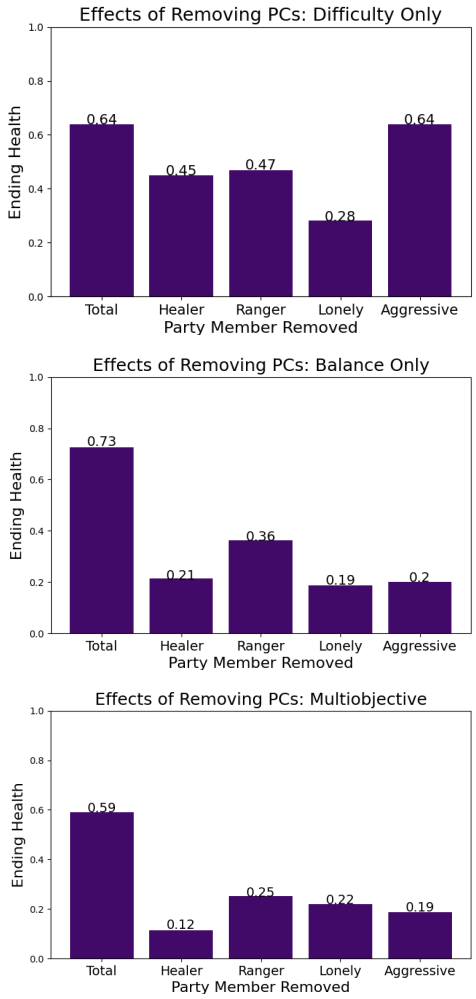


Figure 9: The ending health of the party after the last encounter in the day, with the total party and after removing each party member.

Top Individuals

To get a more in-depth sense of the encounters produced, we examine the top individual at the final population across all three GA runs. For ease of reference, encounters will be referred to as the balanced day (from the run using balance only), the difficulty day (from the run using difficulty only), and the multi-objective day (from the run using both difficulty and balance). The multi-objective day was selected from the first front based on crowding distance. The combat days are provided in Table 1. All monsters are those referenced in the Monster Manual (Mearls and Crawford 2014).

Balance To examine the balance of the encounters in more depth, probabilistic simulations (with 100 playthroughs) were conducted with the entire party, and after removing each party member in turn. The ending health of the party after the last encounter (which is desired to be 0.6 in the difficulty fitness) is shown in Figure 9, along with the average ending health when removing each party member. While the

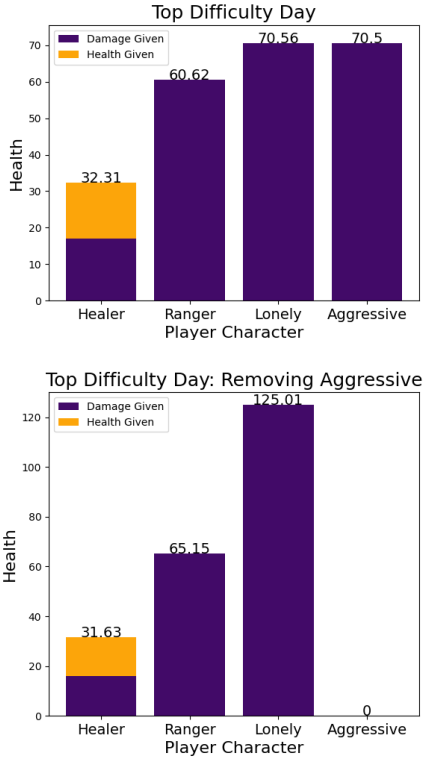


Figure 10: The contributions of each party member through the encounters when fighting with the entire party (top) and after removing the Aggressive player (bottom).

end health over 0.6 with all party members for all three days, only the balance and multi-objective days have substantial reductions in health when removing party members. In particular, removing the Aggressive PC seems to not affect the difficulty day.

We can examine this further by looking how each party member contributes to the fight, both in terms of damage given to enemies and health given to allies. Figure 10 shows how the party members contribute to the difficulty day both when the entire party is present, and when the Aggressive PC is removed. With the entire party present, the Aggressive PC deals about the same amount of damage as the Lonely PC. However, when the Aggressive PC is removed, the Lonely PC roughly doubles the amount of damage it gives.

Difficulty The average ending health for the three encounter sets is shown in Table 3. The difficulty day aligns most closely with the desired difficulty (1, 0.9, 0.6), with the first encounter in the day being trivial, the second having some challenge, and the third being the most challenging. The balance day overall was easier (ending with 0.71 health), and did not increase in difficulty throughout the day. The multi-objective day aligns fairly closely with the desired difficulty, with a slightly harder first encounter and a slightly easier last encounter. This can be seen when looking at the monsters in Table 1, where the difficulty day increases the number of monsters between each encounter and the

Fitness type	Encounter	Average Ending Health
Difficulty	1	1
	2	0.95
	3	0.60
Balance	1	0.98
	2	0.87
	3	0.71
Multiobjective	1	0.96
	2	0.89
	3	0.65

Table 3: The average ending health at the end of each encounter for the three encounter sets.

multi-objective day increases the challenge rating between encounters 1 and 2. However, the monsters in the balance day do not increase much in numbers or challenge.

Discussion

We proposed a method for evaluating DnD encounters that allows game information to be retrieved using a single simulated game. This measure appears to be a reasonably reliable method for estimating the difficulty of an encounter. However, the expected simulation struggled more to estimate balance fitness.

An interesting outcome from our experiments was that optimizing for balance also somewhat optimized for the desired difficulty in these tests. This could explained by the properties of the balance function. The balance function increases the difference in health when each player is removed. If the encounter is very difficult, the ending health with the entire party would be low and the difference of removing a party member would be minimal. In the other direction, if the encounter is too easy, a reduced party can still be successful. In this way, balance is a measure itself of optimal difficulty. However, the difficulty function still allows for greater control, particularly in terms of controlling difficulty at each encounter.

Future Work

Future work can look at how to refine simulations using the expected outcomes for events. In particular, simulations could incorporate discrete events, like being knocked prone, where the probability of the event cannot be multiplied by the expected output. This type of expected simulation can be applied to other genres with large amounts of probabilistic outcomes, such as board games or fighting video games. Additionally, defining games in terms of expected outcomes can be a way to build tools that are game-agnostic. A general TTRPG simulation system could define all actions by their expected changes to the game state.

Future work could also validate generated encounters with human playtests. While previous work (Shyne 2023) suggested the simulation environment can detect difficulties beyond what is available in the base ruleset, this has yet to be verified with human playtests. Similarly, user studies with GMs could study how useful tools like this are in designing encounters. Other measures for encounters, such as player enjoyment or implied narrative could be considered as well.

The capacities of the simulation environment and GA could be expanded. For example, this environment could be expanded to represent encounters beyond the first level. The GA could be expanded to generate new monsters along with swapping between static monsters.

Conclusion

Combat in DnD is a complicated process, making designing new encounters a challenging problem. To address this we created a generic algorithm that could generate new sets of encounters based on a party and the desired difficulty given by a GM. In an attempt to reduce evaluation time, we created a method for deterministically simulating games using expected outcomes. This was successful with the fitness function for difficulty but struggled to produce fit individuals using the balance function. Probabilistic simulation was then used to generate encounters that both matched the desired difficulty and ensured that every player was necessary for success. We found that while the difficulty fitness allowed for more fine-grained control, the balance fitness alone also captured a measure of optimal difficulty. In the examined combat days, the multi-objective provided the best balance between these two fitness functions.

Appendix

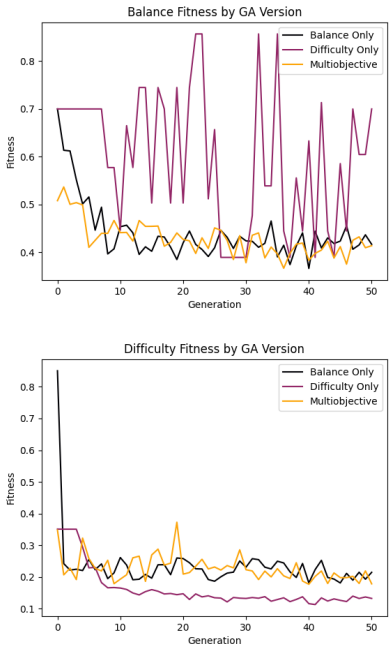


Figure 11: Experiment two was run again, to demonstrate balance can converge quickly.

References

Acharya, D.; Kelly, J.; Tate, W.; Joslyn, M.; Mateas, M.; and Wardrip-Fruin, N. 2023. Shoelace: A storytelling assistant for GUMSHOE One-2-One. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–9.

Acharya, D.; Mateas, M.; and Wardrip-Fruin, N. 2021. Interviews towards designing support tools for TTRPG game masters. In *International Conference on Interactive Digital Storytelling*, 283–287. Springer.

Albaghajati, A.; and Ahmed, M. 2020. Video game automated testing approaches: An assessment framework. *IEEE transactions on games*, 15(1): 81–94.

Alvarez, A.; Dahlskog, S.; Font, J.; Holmberg, J.; Nolasco, C.; and Österman, A. 2018. Fostering creativity in the mixed-initiative evolutionary dungeon designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 1–8.

Crain, H.; Carpenter, D.; and Martens, C. 2022. Evaluating a Casual Procedural Generation Tool for Tabletop Role-Playing Game Maps. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–6. IEEE.

Crawford, J.; Wyatt, J.; Schwalb, R. J.; and Cordell, B. R. 2014. *Player's handbook*. Wizards of the Coast LLC.

Deb, K.; Agrawal, S.; Pratap, A.; and Meyarivan, T. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, 849–858. Springer.

Dormans, J. 2010. Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the 2010 workshop on procedural content generation in games*, 1–8.

Everman, B.; Villwock, T.; Chen, D.; Soto, N.; Zhang, O.; and Zong, Z. 2023. Evaluating the carbon impact of large language models at the inference stage. In *2023 IEEE international performance, computing, and communications conference (IPCCC)*, 150–157. IEEE.

Faria, F. G.; Pereira, L. T.; and Toledo, C. F. M. 2019. Adaptive generation of characters for tabletop role playing games. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 39–46. IEEE.

Guth Jarkovskỳ, T. 2023. *Adaptive generated encounters in a rogue-like role-playing video game*. Master's thesis, Univerzita Karlova.

Guzdial, M.; Acharya, D.; Kreminski, M.; Cook, M.; Eladhari, M.; Liapis, A.; and Sullivan, A. 2020. Tabletop role-playing games as procedural content generators. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–9.

Holmgård, C.; Green, M. C.; Liapis, A.; and Togelius, J. 2018. Automated playtesting with procedural personas through MCTS with evolved heuristics. *IEEE Transactions on Games*, 11(4): 352–362.

Horswill, I. 2019. Imaginarium: A tool for casual constraint-based pcg. In *Proceedings of the AIIDE Workshop on Experimental AI and Games (EXAG)*.

Jaffe, A.; Miller, A.; Andersen, E.; Liu, Y.-E.; Karlin, A.; and Popovic, Z. 2012. Evaluating competitive game balance with restricted play. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 8, 26–31.

Jiao, J.; Afroogh, S.; Xu, Y.; and Phillips, C. 2024. Navigating LLM Ethics: Advancements, Challenges, and Future Directions. *arXiv preprint arXiv:2406.18841*.

Kelly, J.; Mateas, M.; and Wardrip-Fruin, N. 2023. Towards computational support with language models for TTRPG game masters. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–4.

Lang, E. W.; and Young, R. M. 2023. MULTISTYLE: characterizing multiplayer cooperative gameplay by incorporating distinct player playstyles in a multi-agent planner. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 97–106.

Li, J.-Y.; Zhan, Z.-H.; and Zhang, J. 2022. Evolutionary computation for expensive optimization: A survey. *Machine Intelligence Research*, 19(1): 3–23.

Liapis, A. 2017. Multi-segment evolution of dungeon game levels. In *Proceedings of the genetic and evolutionary computation conference*, 203–210.

Liapis, A.; Holmgård, C.; Yannakakis, G. N.; and Togelius, J. 2015. Procedural personas as critics for dungeon generation. In *Applications of Evolutionary Computation: 18th European Conference, EvoApplications 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 18*, 331–343. Springer.

Mearls, M.; and Crawford, J. 2014. *Monster Manual: Dungeons and Dragons*. Wizards of the Coast.

Mugrai, L.; Silva, F.; Holmgård, C.; and Togelius, J. 2019. Automated playtesting of matching tile games. In *2019 IEEE Conference on Games (CoG)*, 1–7. IEEE.

Newman, P.; and Liu, Y. 2022. Generating descriptive and rules-adhering spells for dungeons & dragons fifth edition. In *Proceedings of the 9th Workshop on Games and Natural Language Processing within the 13th Language Resources and Evaluation Conference*, 54–60.

Pereira, L. T.; Toledo, C.; Ferreira, L. N.; and Lelis, L. H. 2016. Learning to speed up evolutionary content generation in physics-based puzzle games. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, 901–907. IEEE.

Popa, A. 2023. *Gotta adjust them all!: dynamic difficulty adjustment of role-playing games through procedural generation of non-player characters*. Master's thesis, University of Twente.

Shyne, F. 2023. *Automatic Play-testing of Dungeons and Dragons Combat Encounters*. Undergraduate thesis, Union College.

Tang, K.; Gasque, T. M.; Donley, R.; and Sullivan, A. 2023. "It Has to Ignite Their Creativity": Opportunities for Generative Tools for Game Masters. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–6.

Viana, B. M.; Pereira, L. T.; and Toledo, C. F. 2022. Illuminating the space of dungeon maps, locked-door missions and enemy placement through MAP-Elites. *arXiv preprint arXiv:2202.09301*.

Wu, C.-J.; Raghavendra, R.; Gupta, U.; Acun, B.; Ardalani, N.; Maeng, K.; Chang, G.; Aga, F.; Huang, J.; Bai, C.; et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4: 795–813.

Zhang, Y.; Fontaine, M. C.; Hoover, A. K.; and Nikolaidis, S. 2022. Deep surrogate assisted map-elites for automated hearthstone deckbuilding. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 158–167.

Zhu, A.; Martin, L.; Head, A.; and Callison-Burch, C. 2023. CALYPSO: LLMs as Dungeon Master's Assistants. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 380–390.

Zook, A.; Fruchter, E.; and Riedl, M. O. 2014. Automatic playtesting for game parameter tuning via active learning. *Proceedings of the 9th International Conference on the Foundations of Digital Games*.