A Metaheuristic Approach to Combat Encounter Generation in Dungeons & Dragons

Paola Cuadra[0009−0009−3751−931X]*, Benoît Bossavit[0000−0003−1236−3020], and Christian Cintrano[0000−0003−2346−2198]

ITIS Software, University of Málaga, Spain
{pcr21, benoit.bossavit, cintrano}@uma.es

Abstract. Procedural generation of game encounters in tabletop RPGs, such as Dungeons and Dragons, poses a dual challenge: ensuring both balanced difficulty and sufficient diversity to maintain player engagement. This work explores the application of metaheuristic optimization techniques to automatically generate diverse combat encounters under predefined constraints. We model encounter generation as an optimization problem, where candidate solutions are evaluated through a fitness function that captures difficulty, coherence, and structural validity.
Our approach compares several metaheuristics, including evolutionary algorithms, hill climbing, and random search, focusing on their ability to optimize fitness and to promote diversity across generated encounters.
Experimental results show that while simpler methods, such as random search, can produce valid encounters, metaheuristic approaches achieve a better trade-off between fitness and diversity. These findings highlight the importance of explicitly addressing diversity in procedural content generation and demonstrate that metaheuristics provide an effective framework for generating varied and engaging combat scenarios.

Keywords: Evolutionary Algorithms · Heuristic Search · Stochastic Simulation · Hill Climb · Procedural Content Generation.

1 Introduction

Tabletop Role-Playing Games (RPGs) involve a group of players who interpret characters within a fictional world. One player assumes the role of Game Master (GM), who defines a storyline composed of non-player characters, missions, events, including combat encounters, and narrates the adventure while ensuring the rules are followed. This universe is supported by a structured ruleset, among which Dungeons & Dragons 5th ed. (D&D) has become both a reference and an influential model for most role-playing games. Although no exact figures regarding the number of users have been formally published, on the occasion of its 50th anniversary, the publisher Hasbro reported more than 50 million participants, with an estimated economic value exceeding 1.5 billion dollars [2].

* Corresponding author: pcr21@uma.es

Most research in this field focuses on supporting the generation of narrative, which is a major feature of RPGs [11]. However, these games also engage players through exploration, tactical decision-making, character progression, and combat. Beyond narrative, the design of combat encounters remains particularly challenging, as party failure may terminate the campaign. Therefore, the GM must provide engaging encounters while maintaining an appropriate level of challenge. D&D proposes an estimation of encounter difficulty known as Challenge Rating (CR), a numerical value intended to represent the appropriate character level for a party of four. However, this generalized metric does not account for several relevant factors, such as party heterogeneity, available inventory, environmental conditions, narrative coherence, and the desired difficulty intended by the GM. These elements are crucial for maintaining engagement according to the theory of game flow [1], where insufficient challenge leads to boredom and excessive difficulty results in frustration.

Several studies have explored computational approaches to support the generation of combat encounters. One study proposed a formal logic model that considers character statistics to estimate encounter difficulty, although the resulting gameplay experience was not convincing [3]. Another approach applied deep learning based on players' behavior from an online RPG video game; however, the system focused on one-versus-one combat, excluding the rest of the party [6]. Reinforcement learning models have also been investigated [4], [7], where the best results were obtained when the reward function incorporated victory rate, number of combat turns, and the remaining health of the party [7]. Nevertheless, these studies rely on simplified simulators and a limited set of enemy types, restricting encounter diversity. Another line of work employs genetic algorithms to generate D&D combat content [10]. In that work, population-based optimization is used to evolve complete combat days composed of multiple encounters, while simultaneously balancing encounter challenge and player contribution. Encounter difficulty is estimated from the remaining party health after each encounter, and experiments are conducted on a restricted subset of low-level monsters. In contrast, our approach focuses on the generation of individual encounters and evaluates difficulty through a target victory rate obtained from repeated combat simulations.

In this paper, we focus on optimizing combat encounters while considering the diversity of generated solutions using a large repository of 334 D&D monsters and maintaining game flow throughout play. We model encounters as dynamic components where the balance between player capability and encounter difficulty is set by the GM, and validate the generation with simulations based on the D&D ruleset. In this context, we investigate the following research questions (RQ):

RQ1. Can combat encounters be optimized while maintaining an appropriate level of challenge according to game flow theory?

RQ2. Does considering encounter diversity increase the variety of possible combat configurations?

RQ3. How do population-based and trajectory-based genetic metaheuristics compare in generating diverse and balanced encounters?

2 Combat Encounter Optimization Problem

In this section, we will formalize our problem of optimizing enemy encounters under specific game-flow conditions. First, we consider the following variables:

- A set $E = \{e_1, e_2, \ldots, e_N\}$ that represents an enemy encounter. Each e_i is a real enemy from the 334 monster in the Monster Manual.
- A set $P = \{p_1, p_2, \ldots, p_M\}$ as the party (set) of players p_i that fights against the enemies in E.
- A set of pairs $S =< E, P >$ of fight simulations between E and P. And $S' \subseteq S$ is the subset of simulations in which the party wins the fight.
- Let $d(s)$ denote the number of defeated party members in a successful simulation $s \in S'$.
- A variable $target \in [0, 1]$ that represents the difficulty level the game master wants for the encounter. A value close to 1 indicates an easy encounter for the players, while a value of 0 indicates a battle that is impossible to win.

Equation 1 mathematically defines the formulation of our problem

$$\min \quad |target - wr_P(E)| + \lambda\, dr_P(E) \tag{1a}$$

$$\text{with} \quad wr_P(E) = \frac{1}{|S|} \sum_{s \in S} s(E, P) \tag{1b}$$

$$dr_P(E) = \frac{1}{|P|\,|S'|} \sum_{s \in S'} d(s) \tag{1c}$$

With this formulation, we aim to identify matchups that minimize the gap between the *target* specified by the GM and the average win rate $wr_P(E)$ observed in the simulator. However, encounters in which some players are eliminated early while the party still achieves victory tend to produce less satisfying gameplay experiences. To account for this, we introduce a penalty function $dr_P(E)$ that measures partial party mortality in successful encounters. For each simulation in which the party wins, we compute the proportion of defeated party members and use its average as a penalty term. This encourages encounters that result in more cohesive outcomes, i.e., either a full party victory or a total party defeat.

Therefore, a solution is represented by a list $E' = [e_1, e_2, \ldots, e_i]$, $e_i \in E$ of enemies. This list is limited to 10 enemies due to the DM's approach to usability.

3 Algorithms

Given the stochastic nature of the simulation-based evaluation, we selected metaheuristic algorithms, such as Random Restart Hill Climb (RRHC) and Steady-State Genetic Algorithm (SSGA). We also run a Random Search (RS) as a baseline for comparison in our problem instance. Let P, M, S, w, E, and k denote the players, enemies (monster set), simulator, target win-rate, stopping criterion, and number of simulations per evaluation, respectively. We consider the following approaches.

3.1 Random Restart Hill Climb (RRHC)

RRHC [8] is based on a local improvement mechanism. Starting from an initial randomly generated encounter, iteratively explores neighboring solutions and moves to one with better fitness value, as shown in Algorithm 1. Neighborhoods are visited by applying a random modification to the current encounter: adding, removing, or replacing a single monster. These operations define a local search space over encounter compositions. The search is periodically restarted at a new random encounter when no improvement is observed for r consecutive iterations, to escape local optima. This strategy aims to balance the exploitation of the local search space with the diversification achieved through random restarts.

Algorithm 1: Random-Restart Hill Climbing

```
Input: P, M, S, w, T, k
Init: e ← RandomEncounter(P, M);
f ← Fitness(Sim(e, k), w); E* ← {e}; f* ← f;
i ← 0;
while termination criterion T not met do
    e' ← Neighbor(e);
    f' ← Fitness(Sim(e', k), w);
    if f' < f then
        e ← e'; f ← f'; i = 0
    else
        i ← i + 1;
        if i = r then
            E* ← E* ∪ {e'};
            e ← RandomEncounter(P, M);
return E*
```

3.2 Steady-state Genetic Algorithm (SSGA)

SSGA [12] evolves a population of encounters through selection, crossover, and mutation. At each iteration, a single offspring is generated and inserted into the population by replacing the worst non-elite individual. This replacement strategy preserves the best solution found while continuously introducing new candidate encounters. A pseudocode of the algorithm is presented in Algorithm 2.

This incremental strategy balances exploration and exploitation with a lower computational cost than generational approaches. Moreover, the steady-state process allows gradual improvement under a fixed evaluation budget, which is necessary to evaluate different algorithms under the same criteria. The parameters p_c and p_m denote the crossover and mutation probabilities, respectively. The crossover operator combines the monsters present in both parents into a common pool. The offspring is then generated by randomly sampling monsters from

this pool until the desired encounter size is reached. To promote encounter diversity and reduce premature convergence, offspring identical to existing population members are discarded and not evaluated. Mutation applies local perturbations by randomly adding, removing, or replacing a single monster.

Algorithm 2: Steady-State Evolutionary Search

Input: $P, M, S, w, N, T, k, p_c, p_m$
Init: $Pop \leftarrow$ random encounters of size N;
foreach $e \in Pop$ **do**
  $f(e) \leftarrow Fitness(Sim(e, k), w)$;
while *termination criterion* T *not met* **do**
  $p_1, p_2 \leftarrow SelectionTournament(Pop)$;
  if $rand() < p_c$ **then**
    $c \leftarrow Crossover(p_1, p_2)$;
  else
    $c \leftarrow p_1$;
  if $rand() < p_m$ **then**
    $c \leftarrow Mutation(c, M)$;
  $f(c) \leftarrow Fitness(Sim(c, k), w)$;
  $e^* \leftarrow \arg\max_{e \in Pop} f(e)$;
  if $f(c) < f(e^*)$ **then**
    replace e^* with c in Pop;
return Pop

3.3 Random Search (RS)

RS is used as a baseline method, providing a simple yet effective way to sample the search space without introducing any bias. Despite its simplicity, it provides a lower bound on performance and serves as a reference point for the expected behavior of RRHC and SSGA. We can see the pseudocode in Algorithm 3.

Algorithm 3: Random Encounter Search

Input: P, M, S, w, T, k
Init: $e^* \leftarrow \emptyset$, $f^* \leftarrow +\infty$, $t \leftarrow 0$;
while *termination criterion* T *not met* **do**
  $e \leftarrow RandomEncounter(P, M)$;
  $f \leftarrow Fitness(Sim(e, k), w)$; $t{+}{+}$;
  if $f < f^*$ **then**
    $e^* \leftarrow e$; $f^* \leftarrow f$
return e^*

4 Experimental Setup

The encounters are encoded as variable-length lists of monsters, with sizes ranging from 1 to 10. Monsters are selected from a repository containing 334 different creatures extracted from the D&D 5th Edition ruleset. The party of players is composed of four level 5 characters.

The development of an encounter determines its win rate, which ranges from 0 to 1. In the experiments, the target win rates for the encounters are $target = \{0.6, 0.8, 0.95\}$, which represent a difficult, moderate, and easy encounter. And the penalization function is weighed by $\lambda = 1$. To assess its quality, encounters are evaluated through a stochastic simulation process. Each encounter is simulated 100 times, and its fitness is computed as mentioned before.

During each simulation, both players and enemies follow consistent and class-aware behaviors. Players act according to their roles and abilities, approximating rational decision-making in combat. For instance, melee-oriented characters prioritize weapon-based attacks, while spellcasters rely on their available spells.

Regarding running parameters for experiments, SSGA uses $p_c = 0.9$ and $p_m = 0.4$. The number of iterations to restart for RRHC was set to $r = 10$.

All the experiments have been executed on a Windows machine with an AMD Ryzen 7 5700U with Radeon Graphics (1.80 GHz), 16 GB of RAM. Also, the source code is available in [5].

5 Experimental Results

To address our research questions, we compare the performance of three optimization algorithms: RS, RRHC, and SSGA, each evaluated under controlled experimental conditions.

5.1 Performance Evaluation

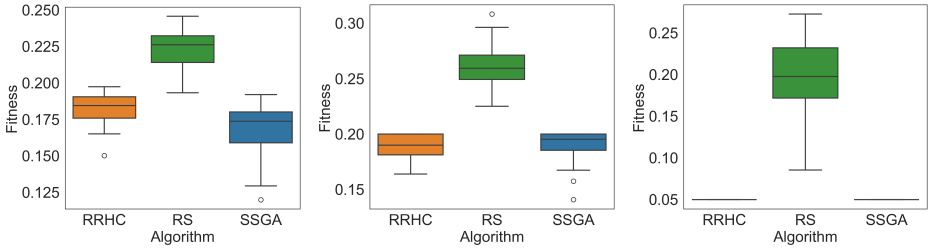
To compare the performance of the three algorithms, we adopt a time-based evaluation protocol combined with multiple independent runs. Each algorithm is executed within a fixed time budget of 385 seconds, corresponding to the time required by the slower algorithm (SSGA) to perform 10,000 fitness evaluations. This ensures a fair comparison under equal computational resources.

Due to the stochastic nature of both the optimization algorithms and the combat simulator, we perform 30 independent runs for each algorithm and each target win rate value. With this experimental design, we aim to evaluate the average performance, variability, and robustness of each method. Consequently, the comparison is based on the distribution of the best fitness values obtained in each run, rather than on a single execution. Table 1 summarizes the distribution of fitness values across all configurations.

The results are analyzed using both descriptive statistics and non-parametric statistical tests. Since the results are paired across seeds and may not follow a normal distribution, we employ the Friedman test to detect global differences

Table 1: Descriptive statistics of fitness values across 30 runs for each algorithm and target win rate. Lower values indicate better performance. Best values for each column and target are highlighted.

target	Algorithm	Mean	Median	Std	Min	Max
0.6	RRHC	0.182	0.185	**0.011**	0.150	0.198
	SSGA	**0.168**	**0.174**	0.019	**0.120**	**0.192**
	RS	0.223	0.226	0.014	0.193	0.246
0.8	RRHC	**0.190**	**0.190**	**0.011**	0.164	**0.200**
	SSGA	**0.190**	0.195	0.015	**0.141**	**0.200**
	RS	0.260	0.259	0.020	0.225	0.308
0.95	RRHC	**0.050**	**0.050**	**0.000**	**0.050**	**0.050**
	SSGA	**0.050**	**0.050**	**0.000**	**0.050**	**0.050**
	RS	0.197	0.198	0.040	0.086	0.273



(a) $target = 0.6$ (b) $target = 0.8$ (c) $target = 0.95$

Fig. 1: Distribution of fitness values for RS, RRHC, and SSGA across 30 runs for different *target* win rates. Lower values indicate better performance.

among the algorithms. When significant differences are found, we perform post-hoc pairwise comparisons using the Wilcoxon signed-rank test with correction for multiple comparisons. We find significant differences between each algorithm (p-values < 0.001) except for $target = 0.8$, where no significant differences are found between SSGA and RRHC (p-value $= 0.69$). In this way, we assess the practical performance of each algorithm in terms of the best solutions it can produce under realistic constraints.

Fig.1 shows the distribution of fitness in each scenario. For the most challenging scenario ($target = 0.6$), SSGA consistently achieves lower fitness values than both RRHC and RS. This is reflected in the lower median and better minimum values, indicating its ability to discover higher-quality solutions. However, SSGA exhibits a wider spread in its distribution, indicating a more exploratory search behavior compared to RRHC, which yields more concentrated results around slightly higher fitness values. RS clearly performs worse, with consistently higher fitness values across all runs.

For the intermediate scenario ($target = 0.8$), the distributions of SSGA and RRHC largely overlap, indicating similar performance between both algorithms. Although SSGA still shows slightly greater variability, no clear dominance is observed. In contrast, RS remains clearly separated from both methods, with consistently worse fitness values.

In the scenario, $target = 0.95$; both SSGA and RRHC collapse to equivalent distributions and consistently achieve the best fitness values across all runs. Meanwhile, RS continues to produce significantly worse results, with a much wider distribution and higher fitness values.

5.2 Encounter Diversity

In addition to solution quality, we analyze the structural diversity of the generated encounters, which is a key factor in encounter design that directly impacts gameplay variety and player experience. We evaluate the best encounter produced by each algorithm using metrics such as the number of unique monster types and Shannon entropy [9]. Shannon entropy is computed on the distribution of monster types within an encounter. Each monster type is treated as a category, and its probability is defined as its relative frequency in the encounter. The entropy value reflects both the number of distinct monster types and their even distribution. Higher values indicate more heterogeneous and evenly distributed compositions; lower values correspond to repetitive or homogeneous encounters.

Equation 2 shows the entropy metric, where E represents the encounter, n is the number of distinct monsters, and p_i denotes the relative frequency of monster i within the encounter, computed as the ratio between its occurrences and the total number of monsters. The resulting entropy value reflects how evenly the different monsters are distributed.

$$H(E) = -\sum_{i=1}^{n} p_i \log(p_i), \quad \text{where } p_i = \frac{f_i}{|E|} \tag{2}$$

This analysis allows us to assess whether different metaheuristic strategies achieve balanced encounters and produce varied, non-repetitive configurations. The results below were obtained from $n = 10,000$ fitness evaluations per algorithm execution. Table 2 presents the diversity metrics of the best encounters obtained across 30 independent runs. The results reveal a clear relationship between diversity, solution quality, and problem difficulty.

RS produces the most diverse encounters, reaching maximum size and entropy values across all target win rates. This behavior is associated with significantly worse fitness values, indicating that diversity alone is insufficient to generate balanced encounters and highlighting the need for guided optimization.

RRHC shows a tendency to reduce diversity as the target win rate increases. For more challenging scenarios ($target = 0.6$), it generates moderately diverse encounters, while for easier targets (0.8 and 0.95), it converges towards simpler configurations with fewer unique monsters. This pattern suggests that trajectory-

Table 2: Average diversity statistics of the best encounters across 30 runs for each algorithm and *target* win rate. The total CR represents the aggregated challenge rating of all enemies in the encounter.

target	Algorithm	Fitness	Size	Unique	$H(E)$	CR (total)
0.6	RRHC	0.182	5.70	5.63	1.662	11.70
	RS	0.223	10.00	9.87	2.284	7.53
	SSGA	0.168	7.97	7.67	2.002	9.31
0.8	RRHC	0.190	3.77	3.73	1.194	7.78
	RS	0.260	10.00	9.73	2.264	6.31
	SSGA	0.190	4.03	3.80	1.266	4.79
0.95	RRHC	0.050	2.37	2.37	0.720	1.03
	RS	0.197	10.00	9.70	2.261	3.57
	SSGA	0.050	3.97	3.63	1.228	1.09

based methods naturally favor low-complexity solutions when the search space contains many acceptable configurations.

SSGA exhibits a more adaptive behavior across all scenarios. It maintains higher diversity in challenging cases, where exploration is necessary to discover high-quality solutions, and reduces diversity in simpler cases where optimal solutions are easier to obtain. This indicates that population-based approaches can dynamically balance exploration and exploitation, preserving structural variety only when it improves solution quality.

According to the Dungeon Master's Guide, a balanced encounter is defined by the Challenge Rating (CR). For instance, an encounter with a total CR of 5 is thought for a party of four level 5 players. Looking at the generation of the best fitness values in Table 2, we can observe that within the same win rate target, the CR value varies from one technique to another. For instance, a difficult encounter with $target = 0.6$, the CR ranges from 9.31 (SSGA) to 11.7 (RRHC), nearly twice the recommended CR of 5. For a moderate encounter with $target = 0.8$, the CR obtained with SSGA approximates what is considered a balanced encounter with a CR=4.79, while the CR obtained from RRHC falls at 7.78, both obtaining the same fitness value. Finally, both metaheuristics approached found a CR close to 1 for an easy encounter with $target = 0.95$.

6 Discussion

Given the results of the previous experiments, we discuss their relevance to each research question proposed at the beginning.

RQ1: Optimization of Combat Encounters under Flow Theory

The results demonstrate that combat encounters can be optimized to match a desired win rate, supporting the applicability of flow theory in this context.

More importantly, they reveal that the difficulty of the optimization problem itself plays a key role in determining the algorithms' relative performance.

For more challenging scenarios (e.g., $target = 0.6$), the better results of SSGA suggest that maintaining a population of solutions is crucial for adequately exploring the search space. In these cases, the encounter generation problem presents a complex landscape, in which simple trajectory-based approaches such as RRHC are more likely to converge prematurely to suboptimal configurations. The ability of SSGA to maintain diversity within the population allows it to escape these local optima and discover higher-quality encounters that better match the desired difficulty.

In easier combats, *target* values 0.8 and 0.95, both SSGA and RRHC consistently achieve the best solutions in the experiments. This indicates that, in these cases, the search space becomes less complex and contains a higher density of acceptable solutions. As a result, even simpler optimization strategies are sufficient to approximate the desired difficulty, reducing the advantage of more sophisticated methods. Moreover, the experiments reveal the consistently poor performance of RS. This highlights the importance of guided search.

With these results, we can say that the metaheuristic techniques applied to this problem make it possible to generate encounters without relying on the CR value, which are tailored to a specific desired difficulty level. This allows the game flow to be preserved while also enhancing it.

RQ2: Impact of Diversity on Encounter Configuration Variety

The results indicate that diversity plays a nuanced role in encounter generation, acting as a necessary but insufficient condition for producing high-quality solutions. Increasing structural variety expands the space of possible configurations, though diverse encounters do not guarantee alignment with the desired difficulty.

The behavior observed for RS suggests that unrestricted exploration leads to structurally rich encounters that lack balance. This points to the existence of a large portion of the search space composed of diverse yet low-quality configurations, reinforcing the need for guidance during optimization.

However, RRHC reduces diversity, reflecting a search dynamic focused on exploiting readily available solutions. This behavior indicates that, when acceptable configurations are abundant, a trajectory-based method favors simpler structures, limiting their ability to generate varied encounters. SSGA, in turn, exhibits a more adaptive mechanism, where diversity is preserved in scenarios that require exploration and reduced when the search becomes less demanding.

As mentioned in Sect. 5.2, the results obtained call into question the usefulness of CR for proposing specific difficulty levels in battles. The official guide is supposed to provide the CR value to define balanced battles, offering a reference for the GM to define more challenging battles. However, this guide may fall short since an encounter with 80% win rate may have a CR value of 4.79 with SSGA and 7.78 with RRHC. Therefore, the metaheuristic approaches allow for defining a more specific and challenging target win rate without relying on CR. This is why the approach to encounter generation using metaheuristic techniques is

interesting, especially from a game design perspective, where we do not always want to provide an experience that poses a trivial challenge to players.

RQ3: Comparison of Population-Based and Trajectory-Based Metaheuristics

The comparison between population-based and trajectory-based metaheuristics highlights complementary strengths rather than a strictly dominant approach. SSGA consistently provides better or equivalent performance, particularly in more challenging scenarios, where maintaining a diverse set of candidate solutions enables a more effective exploration of the search space.

In contrast, RRHC performs competitively in simpler scenarios where the search space contains many acceptable solutions and extensive exploration is less critical. However, its reliance on a single search trajectory limits its ability to escape local optima, reducing its effectiveness in more complex settings.

Overall, these results indicate that population-based approaches are better suited to solving complex and highly stochastic encounter generation problems, while trajectory-based methods remain a viable option when faster convergence and lower computational complexity are required.

7 Conclusion and Future Works

This work has presented a metaheuristic approach to the generation of combat encounters, addressing both solution quality and structural diversity under different target difficulty levels. Among the evaluated methods, the SSGA consistently achieves the best trade-off between fitness and diversity, followed by the RRHC, which reaches competitive results, as confirmed by statistical analysis. The findings highlight that maximizing diversity alone is insufficient and that effective encounter design requires balancing diversity with optimization.

The proposed metaheuristic approach is particularly relevant for tabletop role-playing games such as D&D, where designing balanced and varied combat encounters is a complex and time-consuming task. By automating encounter generation while controlling difficulty and diversity, the method supports game masters in creating richer and more engaging gameplay experiences. Furthermore, it enables the systematic exploration of the encounter design space, allowing the generation of encounters across a wide spectrum of difficulty levels, from accessible challenges to highly demanding combats. This approach shows the potential of metaheuristic optimization as a practical tool for enhancing procedural content generation and improving player experience in role-playing games.

Future work will focus on extending the diversity analysis by incorporating additional domain-specific features, such as monster roles or player characteristics. Also, incorporate more realistic D&D combats as tactical maneuvers. Since SSGA is the algorithm that yields the best results overall, we would like to improve it in the future by adding a set of new operators that ensure the consistency of the enemies selected by the generation process.

Acknowledgments. This research has been supported by MICIU/AEI/10.13039/501100011033 under projects PLEC2023-010266 (SOFIA) and PID2024-158752OB-I00 (AIM-ZERO).

References

1. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. Harper Perennial, New York, NY (1991)
2. Hasbro: Dungeons & dragons 50th anniversary statistics. https://investor.hasbro.com/news-releases/news-release-details/dungeons-dragons-celebrates-50th-anniversary-2024-more-50 (2024), accessed: 2026-04-14
3. von Kacsoh, L., Horst, R., Dörner, R.: From stat block to tabletop: A first-order logic framework for authoring immersive rpg combat encounters. In: Fang, X. (ed.) HCI in Games. pp. 298–314. Springer Nature Switzerland, Cham (2025)
4. Kozma, M., Cerny, V., Gemrot, J.: Tackling generation of combat encounters in role-playing digital games. In: Brejová, B., Ciencialová, L., Holeňa, M., Mráz, F., Pardubská, D., Plátek, M., Vinař, T. (eds.) Proceedings of the 21st Conference Information Technologies – Applications and Theory. p. paper34. CEUR-WS (2021)
5. Paola Cuadra, Benoît Bossavit: Source code. https://github.com/paulapcr/CombatComposer.git (2026), accessed: 2026-05-17
6. Pfau, J., Liapis, A., Volkmar, G., Yannakakis, G.N., Malaka, R.: Dungeons & replicants: Automated game balancing via deep player behavior modeling. In: 2020 IEEE Conference on Games (CoG). pp. 431–438 (2020). https://doi.org/10.1109/CoG47356.2020.9231958
7. Romeo, C., Bagdanov, A.D.: Ntrl: Encounter generation via reinforcement learning for dynamic difficulty adjustment in dungeons and dragons. In: 2025 IEEE Conference on Games (CoG). pp. 1–8 (2025). https://doi.org/10.1109/CoG64752.2025.11114131
8. Russell, S., Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson series in artificial intelligence, Pearson (2020), https://books.google.es/books?id=koFptAEACAAJ
9. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**(3), 379–423 (1948). https://doi.org/https://doi.org/10.1002/j.1538-7305.1948.tb01338.x, https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x
10. Shyne, F., Cooper, S.: No player left behind: Evolving dungeons and dragons combat to optimize difficulty and player contributions. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **20**(1), 224–234 (Nov 2024). https://doi.org/10.1609/aiide.v20i1.31883, https://ojs.aaai.org/index.php/AIIDE/article/view/31883
11. Shyne, F., Cooper, S.: Computational tools for table-top role-playing games: A scoping review. In: Proceedings of the 20th International Conference on the Foundations of Digital Games. FDG '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3723498.3723816, https://doi.org/10.1145/3723498.3723816
12. Syswerda, G.: A study of reproduction in generational and steady state genetic algorithms. vol. 1 (01 1990). https://doi.org/10.1016/B978-0-08-050684-5.50009-4