

Controlling Energy-Propagation-Based Dungeon Generation using Genetic Algorithms

Rafael D. M. dos Santos¹, Rodrigo C. Contreras^{1,2},
Gabriel B. Guimarães², Monique S. Viana³, Rodrigo C. Guido³

¹ Federal University of São Paulo (UNIFESP)
São José dos Campos, SP, 04023-062, Brazil

²University of São Paulo (USP)
São Carlos - SP, 13566-590, Brazil

³São Paulo State University (UNESP)
São José do Rio Preto - SP, 15054-000, Brazil

Abstract. *Introduction:* Procedural Content Generation plays an important role in game development, especially in scenarios that require control over random processes. However, balancing stochastic behavior with rigid Level Design constraints remains a challenging problem. **Objective:** This work investigates the use of classic metaheuristics, specifically a Genetic Algorithm (GA), to control dungeon generation in a high-entropy environment. **Steps:** The approach consists of comparing the performance of the Genetic Algorithm with a Random Search strategy within the same generative environment. **Expected Results:** Experimental results indicate that, despite the high variance of the environment, the GA is able to find high-quality solutions more frequently and with lower computational cost, highlighting advantages and limitations of evolutionary approaches in noisy optimization scenarios.

Keywords Procedural Content Generation, Genetic Algorithms, Level Design, Metaheuristics, Dungeon Generation.

1. Introduction

Game development has become a complex and costly process, particularly regarding content creation, such as maps and environments, which is often carried out strictly manually. In this context, Procedural Content Generation (PCG) emerges as a powerful solution [Zhang et al. 2022]. Considering the process applied to the generation of maps and dungeons in roguelike games, popular titles such as *The Binding of Isaac* and *Enter the Gungeon* exemplify the use and success of this approach. PCG is based on the automatic creation of artifacts, such as environments, textures, and items, through algorithmic manipulation, mitigating the need for manual modeling and increasing replayability [Amato 2017].

In Dias et al. (2025), an approach was proposed whose central objective is the generation of interconnected and randomized maps. By defining an input parameter, referred to by the authors as energy, it becomes possible to control the degree of proximity and connectivity between the paths generated on the map.

In this regard, this work proposes the study and application of classical metaheuristics, such as the Genetic Algorithm (GA) [Goldberg 1989], which employs

evolutionary techniques based on biological evolution, as opposed to random search techniques, operating in highly chaotic and stochastic environments [Dias et al. 2025].

We propose to mitigate the difficulties encountered in Dias et al. (2025) regarding final topological refinement by optimizing and automating the process, aiming to generate dungeons with a specific level design. The proposed solution aims to expand integration, allowing the user to select a number of rooms and hubs, that is, rooms that connect to 3 or 4 others, in addition to a selection of more denser or sparser physical characteristics.

Thus, the main scientific contribution of this work consists of studying the applications of meta-heuristics in highly stochastic environments, as well as their feasibility when compared to standard techniques in the literature, and also in modeling a practical tool with low operational costs for developers.

2. Materials and Methods

The architecture of the proposed system is divided into two main layers: the first acts as the construction engine responsible for the procedural generation of maps, which was initially proposed by Dias et al. (2025), while the second aims to automate the selection of the generator's parameters. This automation is being analyzed using two distinct types of algorithms: a Random Search and a classical GA.

The execution flow begins with the parameterization of the system, in which the user defines the topological goals for the map that the generator must achieve. These guidelines consist of two numerical variables and one Boolean variable:

- **Rooms:** A numerical variable that visually represents the rooms on the map.
- **Hub:** Rooms that exhibit a high degree of connectivity, that is, rooms connected to three or more other rooms, are classified as Hubs. The user-defined numerical variable specifies how many rooms, out of the total generated, should fulfill this structural role.
- **Sparsity:** A *Boolean* control variable. Its function is to define the type of topology the final result will have. If this feature is disabled, the final map will tend toward more clustered and dense shapes. If enabled, the map will tend toward sparse shapes, featuring more corridors and a less dense environment.

Mathematically, the generation and reading of the resulting map are performed using matrices, which represent the existence of rooms as well as their connections [Dias et al. 2025].

2.1. Fitness Function

The function responsible for quantifying the quality of the solution initially extracts three direct metrics from the map: the number of rooms (R), the number of hubs (H), and the density of the bounding grid ($D = \frac{R}{w \cdot h}$), where w and h represent the maximum horizontal and vertical distances, respectively. The total cost to be minimized is given by Equation (1):

$$F(I) = \omega_r(R - R_t)^2 + \omega_h(H - H_t)^2 + P_d + P_e. \quad (1)$$

Here, R_t and H_t represent the numerical targets specified by the user, penalized quadratically by their respective weights ω_r and ω_h . To enforce the spatial constraint, the

density penalty $P_d = \max(0, D - L) \cdot \omega_p$ is applied, which is strictly triggered if the generated density D violates the tolerance limit L .

Finally, P_e acts as a final cost for the map. It adds a constant penalty ρ if the map does not meet the exact targets ($R \neq R_t \vee H \neq H_t$). The combination of the quadratic error with the barrier ρ creates a gradient of priorities: severe structural deviations are quickly discarded due to quadratic growth, while minor errors allow the algorithm to focus on refining the map density. Table 1 summarizes the coefficients used, taking into account the physical aspects of the generator.[Dias et al. 2025].

Table 1. Parameterization of the coefficients used in the Fitness Function.

Var.	Description	Value
ω_r	Room error weight	30
ω_h	Hub error weight	15
ω_p	Density penalty factor	80
ρ	Structural error penalty (P_e)	50
L	Sparsity / Compact maps limit	0.30 / 0.55

2.2. Evolutionary Algorithm

To generate and control the map, two algorithms were modeled that operate differently on the same reference individual, which is shown in Figure 1



Figure 1. Chromosome representation in the GA: parametric genes and their topological effects.

The first algorithm implemented is a genetic algorithm (GA) based on the classical theory [Goldberg 1989]. It operates with a population of 30 individuals over 40 generations. The evolutionary process employs standard operators from the literature: Roulette Wheel Selection, Single-Point Crossover, and Uniform Random Mutation, in addition to the application of strict elitism, ensuring the exclusive preservation of the best individual at each generational transition. Furthermore, the fitness function is applied uniquely to each solution set, and enhanced diversity maintenance techniques were not applied, given the initial need to study the application of meta-heuristics in their most classical form.

The second technique applied consists of a Random Search (RS). To ensure a comparison based on an equivalent computational budget, the RS stopping limit was set at exactly 1,200 independent evaluations of the fitness function; this value, in turn, is mathematically identical to the total number of maps generated by the GA. The RS operates by randomly sampling different combinations of genes within the permitted limits, without any population memory.

The algorithm's parameterization assumed chromosome limits of $E \in [2, 100]$ and $C \in [0.5, 1]$, based entirely on the physical operation of the generator [Dias et al. 2025]. For the remaining parameters, a mutation rate of 5% and a crossover rate of 85% were defined.

In accordance with the Code of Conduct for Authors in SBC Publications, it is declared that Artificial Intelligence tools (such as DeepL and Gemini) were used exclusively for translating the original manuscript from Portuguese to English and for grammatical revision of the text. No generative tools were employed in the creation of the scientific content, data analysis, or algorithmic development.

3. Results and Discussion

To evaluate the algorithm's effectiveness, a series of empirical tests was conducted. The experiments were run on a computer equipped with an 11th-generation Intel Core i5 processor, 16 GB of RAM, and the Windows operating system. For each defined target configuration, 30 independent runs were performed, and the average of the results was used as the basis for analysis.

The results presented in Table 2 highlight the behavior of the algorithms in a highly stochastic environment, where lower fitness values indicate solution superiority. The energy propagation-based map generator exhibits high variance [Dias et al. 2025], such that identical chromosomes can produce significantly different outputs. This characteristic introduces noise into the fitness evaluation, directly impacting the selection process in evolutionary metaheuristics. Rather than artificially mitigating this noise, the analysis demonstrated how the GA adapts its selection process to converge toward consistent solutions even under adverse conditions.

Table 2. Performance Comparison: GA vs RS in Selected Instances.

Target	Type	Min Fit		Avg Fit		Perfect		Time (s)	
		GA	RS	GA	RS	GA	RS	GA	RS
12r_5h	Dense	0.00	0.00	512.36	47.00	13	3	0.34	1.66
20r_8h	Dense	0.00	0.44	80.59	62.92	5	0	0.34	1.73
45r_10h	Sparse	125.56	203.86	292.66	640.38	0	0	0.33	1.71
60r_12h	Sparse	256.99	430.62	623.03	888.46	0	0	0.35	1.72

In the case of RS, the search space consists of only two variables. Coupled with this, the high number of attempts increases the probability of obtaining acceptable solutions. However, due to the noisy nature of the generator, obtaining optimal solutions, that is, solutions with exact values for the target metrics, remains infrequent.

For the GA, an analysis of Figure 2 indicates the presence of high variance across runs. This behavior is associated with the non-deterministic nature of fitness evaluation, which can lead the algorithm to follow non-viable directions based on noisy evaluations of individuals previously considered promising.

In terms of runtime, the GA outperformed the RS. This result can be explained by the GA's ability to explore more promising regions of the search space across generations. In contrast, the RS performs uniform sampling, frequently evaluating configurations with extreme parameters, which increases the computational cost of the generator.

Overall, both algorithms exhibited similar average performance in terms of error. An extended analysis of all types of tests conducted revealed that the GA achieved an overall average performance that was 8.6% better than the RS in terms of error minimization. This advantage held even though the GA exhibited a higher frequency

of outliers across generations. These results indicate the advantage of guided methods in stochastic search scenarios, even in the presence of high noise in the fitness function. Figure 3 shows the final results obtained in tests where the GA found optimal solutions, allowing for a visual analysis of the map generated by the evolution.

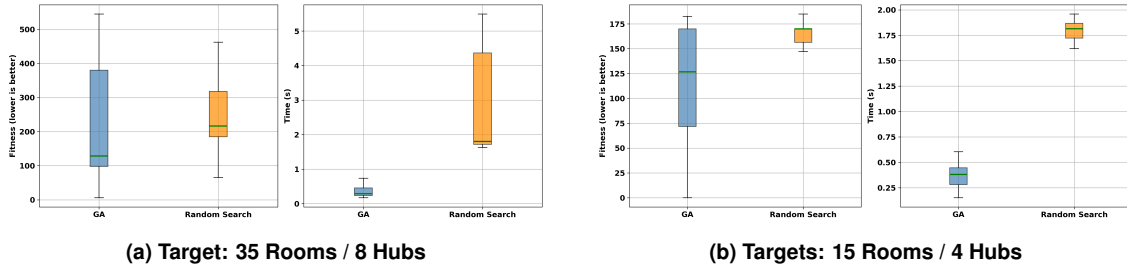


Figure 2. Comparative boxplot between GA and RS, evaluating fitness and time.



Figure 3. Maps with 100% accuracy (Blue: Rooms; White: Corridors; Yellow: Hubs).

4. Final Considerations

Finally, the results demonstrated that the evolutionary method exhibits better computational efficiency and a higher frequency of finding high-quality solutions when evaluating its applicability in real-world situations.

However, it was also concluded that the application of classical meta-heuristics in highly stochastic environments tends to result in variance in their final results. This occurs due to the noise factor present in the decoding of individuals, which can lead the algorithm to premature convergence.

Therefore, the modeling and application of robust evolutionary algorithms to noisy environments emerge as important directions for future work.

Acknowledgement

The authors acknowledge the support of the São Paulo Research Foundation (FAPESP) through research fellowships Grant No. 2025/17551-7 (R.D.M.S.) and Grant No. 2025/10220-5 (M.S.V.).

5. References

Amato, A. (2017). Procedural content generation in the game industry. *Game and Media Technology*, 1:1–9.

- Dias, M. V. P., Franco, A. d. O. d. R., da Silva, J. W. F., e Maia, J. G. R. (2025). Procedural map generation by energy propagation. In *Proceedings of the XXIV Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Salvador, BA, Brazil.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.
- Zhang, Y., Zhang, G., e Huang, X. (2022). A survey of procedural content generation for games. In *2022 International Conference on Culture-Oriented Science and Technology (CoST)*, pages 186–190.