Human Factors in Design, Engineering, and Computing, Vol. 238, 2026, 1689–1699
https://doi.org/10.54941/ahfe1008397

A Multitask Evolutionary Framework for Procedural Content Generation in Games

Hoang Long Nguyen and Taneli Nyyssönen

Department of Computing, Faculty of Technology, University of Turku, Finland

ABSTRACT

Procedural Content Generation (PCG) has emerged as a widely adopted technique used for creating content in various types of games. In recent years, numerous approaches have been proposed to address different PCG problems. Despite the diversity in their concept, most existing generators are designed for specific games, even though many games share structural similarities. In light of this reason, this paper investigates the application of Evolutionary Multitask Optimization (EMTO) for PCG. A Multifactorial Evolutionary Algorithm (MFEA)-based framework is introduced to systematically group PCG problems according to the compatibility of their search spaces, enabling multiple problems to be optimized simultaneously. To evaluate the effectiveness of the proposed framework, experiments were conducted on a range of PCG benchmark problems, along with comparisons to a single-task evolutionary approach. The results highlight both the potential and the challenges of applying EMTO to PCG, particularly with respect to task compatibility.

Keywords: Procedural content generation, Computer games, Multifactorial evolutionary algorithm, Evolutionary algorithm

INTRODUCTION

Game content plays a crucial role in maintaining player engagement, while modern games increasingly demand large, diverse, and customizable content. However, manual content creation is often expensive, time-consuming, and difficult to scale, making it a major bottleneck in game development (Hendrikx et al., 2013). To address these challenges, PCG has been introduced as a technique for generating game content using computer algorithms (Togelius et al., 2011) rather than manual approaches. Accordingly, PCG offers the potential to support scalable content production, enhance games' replayability, and facilitate game development in various aspects, especially saving both cost and time.

One common issue when generating game content via PCG is assessing the validity and suitability of generated instances for the game (Hendrikx et al., 2013), thereby evaluating the generator's efficiency. One attempt to standardize and automate this assessment was made in 2025 by Khalifa et al., with the proposal of the PCG Benchmark. The PCG Benchmark is a framework containing an extendable set of PCG problems. Each problem uses

Received September 23, 2026; Revised October 27, 2026; Accepted November 1, 2026; Available online December 1, 2026

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

the same evaluation scheme: it takes an array of content as input and outputs scores. As a result, the benchmark provides a controlled and reproducible environment for comparing generative methods.

One of the most popular algorithms that has been widely studied in the PCG research community are Evolutionary Algorithms (EAs). Thanks to their flexibility and ease of use, EAs have been applied to a wide range of PCG problems, such as dungeon and platformer level design, puzzle generation, and building layout generation (Hendrikx et al., 2013; Short et al., 2017). However, the weaknesses of conventional EAs are their large resource consumption and their typical design to solve one individual problem at a time (Gupta et al., 2015). In practice, many tasks, particularly those within the same game or domain, often share underlying structural patterns. Leveraging these shared characteristics could enable more efficient content generation. Meanwhile, a long-term goal of PCG research is the development of general content generators, which are systems capable of creating content for different games without being specifically tailored for each one (Hendrikx et al., 2013). Achieving this vision requires methods that can learn from multiple generative tasks simultaneously and generalize beyond a single game.

For these reasons, this paper explores the use of EMTO to tackle the PCG Benchmark, with MFEA (Gupta et al., 2015) adopted as a representative multitask evolutionary algorithm. MFEA can solve multiple problems simultaneously within a single evolutionary process. One key innovation of MFEA lies in the concept of encoding all solutions within a unique representation of population called Unified Search Space (USS). By placing individuals from different tasks into a common representation, MFEA allows the transfer of potentially useful genetic materials between related problems, thereby improving search efficiency and facilitating the achievement of optimal solutions for each task. This capability is particularly relevant for PCG, where multiple related content generation tasks often coexist, such as different modes or parameter settings within the same game, as well as structurally similar tasks across different games. Despite such advantages, directly applying MFEA to PCG remains challenging due to the heterogeneous and structured nature of PCG search spaces. To the best of our knowledge, no attempts have been made to apply EMTO as well as MFEA to the context of PCG in general.

The major contributions of this work can be summarized as follows:

- Design a multitask evolutionary framework for PCG.
- Implement MFEA as a representative multitask optimizer to demonstrate the effectiveness of the framework.
- Analyze and evaluate experimental results on the latest PCG Benchmark with multiple multitask scenarios.
- Compare with the previous EA single-task approach to examine the efficiency of the used algorithm.

The rest of this paper is organized as follows. Section II presents related works, while Section III provides a summary of the PCG Benchmark. Section IV elaborates on the proposed framework. Section V includes the

experiment setups and computational results, including a performance comparison with another algorithm. Finally, conclusions and future extension are discussed in Section VI.

RELATED WORKS

PCG is a research domain that has been studied for several decades, with numerous methods proposed for generating diverse types of game content (Hendrikx et al., 2013). Among these, Search-based Procedural Content Generation (SBPCG) has become a prevalent approach to many PCG problems (Zamorano et al., 2023). Introduced by Togelius et al., SBPCG formulates content creation as an optimization problem with a predefined goal or set of goals and a fitness function (Togelius et al., 2011), which allows search-based methods, such as EAs, to explore complex game content spaces. SBPCG has been applied to diverse PCG problems, with a typical example being generating levels for Super Mario Bros (SMB), a 2D platformer game developed by Nintendo (Nintendo, 1985). Previous work has explored multiple evolutionary approaches for this problem using pattern-based (Dahlskog et al., 2013), latent-space representations (Volz et al., 2018), and coevolution (Flimmel et al., 2021).

Recently, Khalifa et al., introduced the PCG Benchmark to facilitate systematic evaluation of generative algorithms across a diverse set of PCG problems (Khalifa et al., 2025). The authors conducted experiments with three algorithms, namely a random generator, an evolution strategy, and a GA, on the proposed benchmark. The findings demonstrated that GA generally outperformed the other algorithms.

MFEA has been widely used for optimization problems (Luo et al., 2025), including both continuous and discrete tasks. However, according to our literature review, there have been no efforts to apply MFEA to game research or PCG. This may be partially explained by the differences in how PCG tasks are represented and evaluated, making it complicated to construct a USS. With the introduction of the PCG Benchmark, applying MFEA to PCG tasks has become more practical, as the benchmark provides a platform that brings together diverse PCG problems with standardized evaluation procedures. Consequently, this paper presents the first investigation of applying MFEA to the PCG Benchmark.

PROCEDURAL CONTENT GENERATION BENCHMARK

The PCG Benchmark is a framework released in 2025 to support the performance evaluation of generative algorithms. It consists of twelve distinct PCG problems, each featuring its own representation and control parameters. These problems encompass a variety of games across different genres and objectives, including rule generation, pattern generation, platformers, roguelikes, puzzles, and more. The framework also provides three evaluation functions to assess quality, diversity, and controllability criteria of the generated content, along with three different fitness functions used to assess chromosomes during the evolution process (See Table 1).

Table 1: Fitness functions for chromosome evaluation in the PCG Benchmark.

Fitness Function	Optimization Priority
Quality Fitness (Q)	The fitness is calculated from the quality of its content.
Quality then Controllability Fitness (QT)	This fitness function prioritizes maximizing quality, followed by controllability only if the quality is optimal.
Quality, Controllability, then Population Diversity Fitness (QTD)	This fitness function seeks to optimize population diversity once an optimal solution regarding both quality and controllability is achieved.

Additionally, the PCG Benchmark introduces two types of possibility spaces, namely content and control spaces. These spaces define the search space for each problem, and they differ from one another. Hence, this equips the framework with the flexibility needed to handle various types of content in different games.

MFEA-BASED FRAMEWORK FOR PROCEDURAL CONTENT GENERATION

This section introduces our multitask approach for solving the PCG Benchmark in more detail. MFEA is implemented to generate content for a set of K PCG problems simultaneously. Each problem is defined as $P^i = \left(c_r^i, p_r^i, Q^i, T^i, D^i\right)$, where $i = 1 \ldots K$, c_r^i and p_r^i denote the content and control spaces, and Q^i, T^i, and D^i are quality, controllability, and diversity functions, respectively.

Construction of Unified Search Spaces

Traditional MFEA facilitates knowledge exchange by integrating all search spaces into a single USS. However, PCG problems in this benchmark often exhibit heterogeneous representations and various search space structures, with each task comprising two search spaces, i.e., a content space that specifies the structure of the generated artifact, and a control space that governs generation parameters under game-specific constraints. These two spaces differ in both semantics and structure and therefore cannot be meaningfully unified into a single search space.

To tackle this challenge, we propose a USS construction method specifically designed for PCG Benchmarks. Given a set of PCG problems $\{P^i\}_{i=1}^K$ as input, the framework first analyzes the representational compatibility of their search spaces, ensuring that only tasks with compatible semantic interpretations are unified into a shared search space. Compatibility is defined by preserving the minimal structural invariants necessary for valid decoding and meaningful genetic transfer.

1) Integer-valued spaces are considered compatible regardless of their ranges. The unified bounds are defined as: $l = \min_i l_i$, $u = \max_i u_i$, where $[l_i, u_i]$ is the range of task i.

2) Array-based spaces can be unified by adopting the maximum dimension across tasks. The unified array shape is computed as: $\mathrm{d} = \left(\max_i d_{i,1}, \max_i d_{i,2}, \ldots, \max_i d_{i,m}\right)$, where $d_{i,j}$ is the length along dimension j for task i, and m is the maximum number of dimensions among all tasks.
3) In contrast, dictionary-based spaces require consistent key sets, as mismatched keys would invalidate the semantic interpretation of generated content. For each key k, the value is treated as a separate subspace, and the unification is applied according to its type (integer-valued or array-based) as described above. Nested dictionaries are recursively unified using the same rules.

Based on these criteria, satisfied problems are grouped, while incompatible tasks are optimized separately. In this work, task compatibility is primarily determined by the content search space, since content representations must be structurally aligned to enable valid genetic exchange. If the corresponding control spaces are also compatible, those tasks are referred to as *fully compatible*. Otherwise, control parameters are evolved independently. For each group of fully compatible tasks, two independent content USS and control USS are constructed. Then, a single population is evolved to simultaneously solve multiple PCG tasks within one group. The overall USS construction is illustrated in Figure 1.

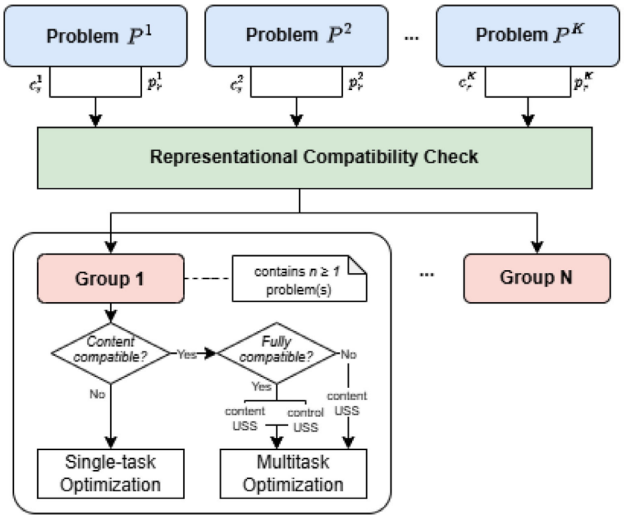


Figure 1: Overview of the proposed framework for unified search space construction.

Unified Chromosome Representation and Decoding

Let U^c and U^p denote the content and control USS constructed for a group of fully compatible PCG tasks, respectively. Each individual in the population is represented as: $I = (I^c, I^p)$, $I^c \in U^c$, $I^p \in U^p$, where I^c is the content vector and I^p is the control parameter vector. When a chromosome is initialized, these vectors are sampled from their respective USS.

For a task P^j, its solution $I^{(j)}$ is obtained by decoding a unified representation $I = (I^c, I^p)$ into its original search spaces (c_r^j, p_r^j). Decoding is applied independently to the content and control segments of the chromosome using the same mapping rules. For integer-valued spaces, decoding maps unified chromosome values into the valid range via modulobased projection. Binary spaces are treated as integer spaces with values restricted to {0, 1}. For array-based spaces, the decoding process first adapts the unified representation to the shape of the considered task by truncating each dimension to the required size. After that, each element in the resulting array is decoded according to the underlying value space. Finally, for dictionary-based spaces, decoding is applied independently to each key. Each key maintains its semantic meaning, and its associated value is decoded recursively based on the corresponding subspace.

Evolutionary Operator

Since chromosomes are encoded using varying structures, the generator adopts the uniform crossover and mutation operators employed in (Khalifa et al., 2025). Note that the operators only perform on both content and control parts when tasks are fully compatible; otherwise, the control parameter stays intact.

EXPERIMENTS AND RESULTS

Experimental Setups

In this section, we assess the effectiveness of our approach using the PCG Benchmark. As mentioned above, this benchmark includes 12 PCG problems, each containing 3 to 4 instances with varying sizes and constraints. Noting that in (Khalifa et al., 2025), the evaluation is conducted using a single instance per PCG problem. This paper, in contrast, considers all instances for each problem, as they can form a multitasking environment suitable for our proposed approach.

To evaluate the performance of the MFEA solver within the proposed framework, this paper adopts the single-task GA from (Khalifa et al., 2025) as a baseline, given its experimentally confirmed efficiency on the benchmark. We utilize the same parameters as the original paper to create a similar experimental environment. For the employed MFEA, the random mating probability (*rmp*) configuration was selected as 0.2, which is also consistent with commonly used values in the MFEA literature. Each algorithm was simulated 10 times on the same computer (Intel Core i7 - 2.10 GHz, 32 GB RAM). The evaluation budget is 20000 evaluations, with a population size of 100 individuals evolving over 200 generations (Khalifa et al., 2025). For GA, these evaluations are dedicated to a single problem instance. For MFEA, the same total budget is shared across K problems, resulting in a reduction of approximately 66%–75% in the number of evaluations allocated to each task. This setup allows us to assess whether inter-task knowledge transfer in MFEA can compensate for the significantly reduced evaluation budget per task. Regarding experimental criteria, we focus on the mean value of fitness results and the number of feasible, controlled, and unique solutions.

Experimental Scenarios

Two experimental scenarios were designed to investigate MFEA's performance and the knowledge transfer between tasks under different multitasking conditions.

- **Scenario 1: Fully compatible multitasking environment**: All instances belonging to the same PCG problem are optimized simultaneously. These instances share the same content and control representations but differ in problem size or control parameters. The results obtained are compared to GA.
- **Scenario 2: Content-compatible multitasking environment**: Instances from different PCG problems are optimized together when their content search spaces are compatible and of comparable size. Specifically, we consider grid-based problems, i.e., *Binary*, *Dangerous Dave*, *MiniDungeons*, *Sokoban*, and *Zelda*, which share some common design abstractions, such as reachability, obstacle placement, and player-goal interaction. Selected instances are further divided into subgroups according to their grid sizes. Noting that only grid-based problems are considered in this work, as other problems have different representations that are incompatible or meaningless to solve together.

Table 2 summarizes the multitask groupings used in the two experimental scenarios. Each row corresponds to a bundle of PCG Benchmark instances that are solved together within a single multitasking run. Instance identifiers within a bundle are separated by underscores. Examples of the best generated levels for all scenarios are found in the documentation[1].

Table 2: Multitask instance groupings for the experimental scenarios.

Scenario	Task Bundle
Fully compatible	12 same-problem bundles
Content-compatible	ddave-v0_mdungeons-v0_sokoban-v0_zelda-v0; binary-v0_ddave-large-v0_zelda-large-v0

Experimental Results

Performance Comparison on Fully Compatible Tasks

Figure 2 illustrates the improvement in maximum Q fitness across generations for the two algorithms. Both algorithms show generally competitive convergence trends in most problems. In contrast, there are clear distinctions in more difficult environments such as *SMB* and *Buildings*, where GA achieves significantly higher and faster fitness growth. The lower performance of MFEA may be related to the increased complexity of the USS, which comprises instances with widely differing level sizes. *Dangerous Dave* is one of the most volatile problems. Nevertheless, MFEA achieves notable improvements, suggesting that multitask optimization may provide beneficial knowledge transfer for this problem. Overall, MFEA tends to exhibit narrower shaded regions, suggesting more consistent performance.

[1]Available at https://github.com/longnh176040/MFEA_PCG

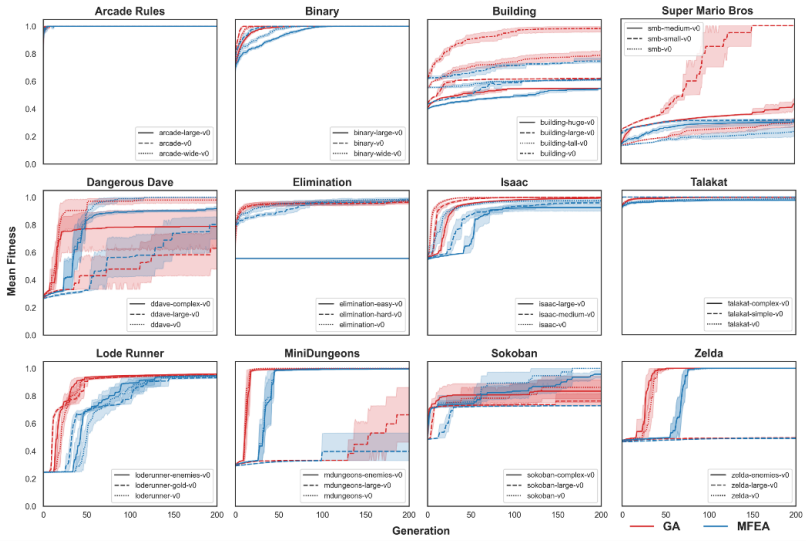


Figure 2: Evolutionary progress of the maximum quality fitness of GA and MFEA. Results are averaged from 10 runs, with 95% confidence intervals as the shaded area.

Figure 3 depicts the averaged number of feasible, unique, and controlled individuals across all instances of each task. Regarding the number of feasible solutions, GA maintains a slight advantage in *Arcade Rules*, *Binary*, *Isaac*, and *Talakat*, while MFEA remains highly competitive and outperforms GA in *Dangerous Dave* and *MiniDungeons*. Both algorithms show similar performance in generating individuals that meet controllability constraints. The most significant difference appears in the number of unique individuals. In almost every environment, MFEA produces a significantly higher number of unique solutions compared to GA. Most notably, under the QTD fitness scheme, MFEA achieves nearly 100% uniqueness for *Arcade Rules*, *Binary*, *Building*, and *Zelda*.

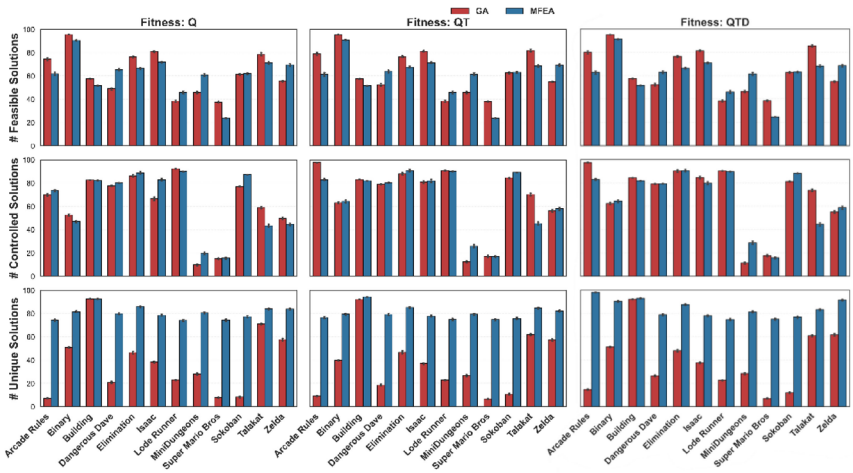


Figure 3: Number of solutions c_i in the 200^{th} generation that are feasible $q(c_i) = 1$, controlled ($t(c_i, p_i) = 1$), and unique ($d(c_i, \mathbb{C}) = 1$) compared to the final population $\mathbb{C}$. Results are averaged from 10 runs across all instances per task, with 95% confidence intervals shown as error bars.

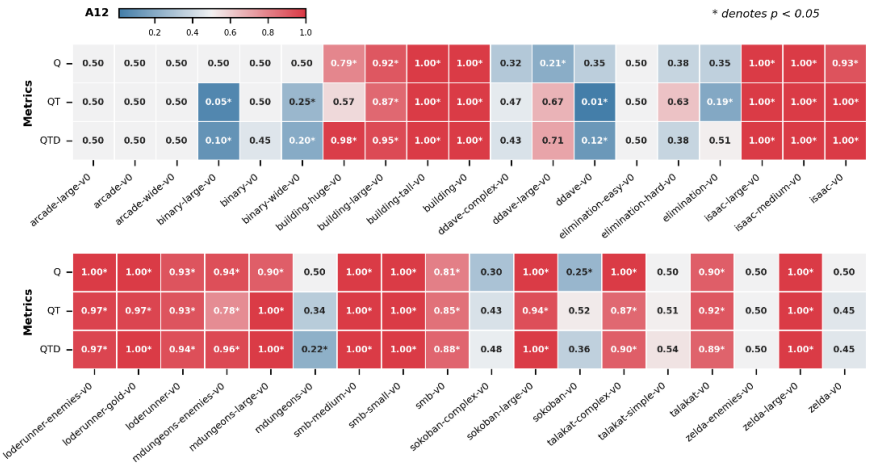


Figure 4: Statistical comparison of fitness performance via A12 effect sizes and Wilcoxon rank-sum tests. Red cells (A12 > 0.5) indicate that GA outperformed MFEA, while blue cells (A12 < 0.5) indicate the opposite. Statistically significant differences are marked with an (*).

Finally, the Wilcoxon rank-sum test and Vargha-Delaney A12 measure were conducted to further assess the performance differences between MFEA and GA (See Figure 4). The results show that in 14 to 16 out of 37 instances (approximately 40%), there is no statistically significant difference between MFEA and GA. GA's wins are concentrated in some problems, such as *Isaac*, *Lode Runner*, and *SMB*.

Overall, for Scenario 1, the results demonstrate that MFEA performs competitively with GA, even with a significant reduction (66% or 75% depending on problem) in evaluation budget. It achieves comparable or sometimes better solution quality while consistently maintaining higher population diversity, which shows its effectiveness as a strategy for PCG, where content variety plays a critical role.

Evaluate Multitask Learning on Content-Compatible Tasks

Figure 5 presents the improvement in maximum Q fitness across generations for two content-compatible environment bundles optimized using MFEA. In the bundle of small instances, improvements in *MiniDungeons* and *Zelda* occur after earlier progress in *Dangerous Dave*, suggesting successful knowledge transfer. Figure 6 shows some representatives of the best generated levels for this bundle. In contrast, the second bundle of three large instances shows prolonged stagnation in *Dangerous Dave* and *Zelda* despite rapid convergence on *Binary*. In comparison to Scenario 1, Scenario 2 exhibits signs of negative transfer. Specifically, the late improvements suggest that while knowledge transfer is possible between different games, the lack of task compatibility creates a transfer barrier that can only be overcome after significant independent evolution.

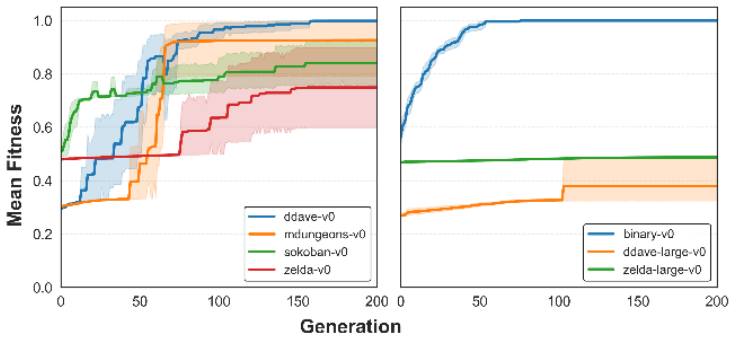


Figure 5: Evolutionary progress of the maximum Quality fitness of MFEA in Scenario 2. Results are averaged from 10 runs, with 95% confidence intervals as the shaded area.

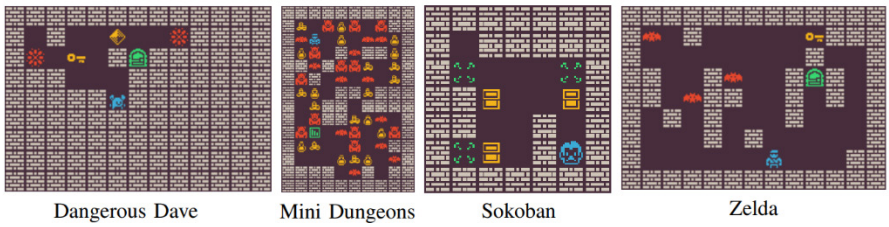


Figure 6: Representatives of the best generated levels optimizing the Quality fitness function in the bundle of small instances in Scenario 2.

CONCLUSION

This paper investigated the application of EMTO to the PCG Benchmark. An MFEA-based framework was introduced, which can systematically group PCG problems based on the compatibility of their content and control search spaces. To evaluate the proposed framework, experiments were conducted under multiple multitask scenarios, together with comparisons against a single-task GA baseline on the PCG Benchmark. Computational results showed that MFEA achieved competitive performance compared to the baseline, even with significantly lower budgets. Furthermore, MFEA presented an adequate capability of maintaining solution diversity. However, the results also demonstrated that the effectiveness of knowledge transfer was highly dependent on the compatibility of problems.

It is important to note that the proposed framework is not restricted to a specific multitask solver. The unified representation and compatibility-based grouping strategy can be integrated with other EMTO methods or population-based approaches. From a practical perspective, this approach is applicable for games that contain multiple related content components, such as large worlds with numerous sub-levels or collections of minigames. Future work will explore adaptive task grouping strategies, comparisons with higher evaluation budgets, and alternative multitask solvers to improve knowledge transfer in PCG.

REFERENCES

Dahlskog, S. and Togelius, J. (2013). Patterns as objectives for level generation. In: *Proceedings of the Second Workshop on Design Patterns in Games*. ACM.

Flimmel, J., Gemrot, J. and Cerny, V. (2021). Coevolution of AI and level generators for super mario game. *2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2093–2100.

Gupta, A., Ong, Y.-S. and Feng, L. (2016). Multifactorial evolution: Toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3), pp. 343–357.

Hendrikx, M., Meijer, S., Van Der Velden, J. and Iosup, A. (2013). Procedural content generation for games. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9(1), pp. 1–22.

Khalifa, A., Gallotta, R., Barthet, M., Liapis, A., Togelius, J. and Yannakakis, G.N. (2025). The procedural content generation benchmark: An open-source testbed for generative challenges in games. *Proceedings of the 20th International Conference on the Foundations of Digital Games*, pp. 1–12.

Luo, G. and Ma, X. (2025). A survey on evolutionary multitask optimization. *Journal of Artificial Intelligence and Technology*.

Nitendo, R&D (1985). *Super mario bros*.

Short, T. and Adams, T. (2017). *Procedural generation in game design*. Boca Raton: Crc Press, Taylor & Francis Group.

Togelius, J., Kastbjerg, E., Schedl, D. and Yannakakis, G.N. (2011a). What is procedural content generation? *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games - PCGames' 11*.

Volz, V., Schrum, J., Liu, J., Lucas, S.M., Smith, A. and Risi, S. (2018). Evolving mario levels in the latent space of a deep convolutional generative adversarial network. *Proceedings of the Genetic and Evolutionary Computation Conference on – GECCO'18*.

Zamorano, M., Cetina, C. and Sarro, F. (2023). *The quest for content: A survey of search-based procedural content generation for video games*.