Level Generation with Constrained Expressive Range

Mahsa Bazzaz
Northeastern University
Boston, MA, USA
bazzaz.ma@northeastern.edu

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

Abstract

Expressive range analysis is a visualization-based technique used to evaluate the performance of generative models, particularly in game level generation. It typically employs two quantifiable metrics to position generated artifacts on a 2D plot, offering insight into how content is distributed within a defined metric space. In this work, we use the expressive range of a generator as the conceptual space of possible creations. Inspired by the quality diversity paradigm, we explore this space to generate levels. To do so, we use a constraint-based generator that systematically traverses and generates levels in this space. To train the constraint-based generator we use different tile patterns to learn from the initial example levels. We analyze how different patterns influence the exploration of the expressive range. Specifically, we compare the exploration process based on time, the number of successful and failed sample generations, and the overall interestingness of the generated levels. Unlike typical quality diversity approaches that rely on random generation and hope to get good coverage of the expressive range, this approach systematically traverses the grid ensuring more coverage. This helps create unique and interesting game levels while also improving our understanding of the generator's strengths and limitations.

Keywords

video games, pcg, expressive range, constraint-based generation

ACM Reference Format:
Mahsa Bazzaz and Seth Cooper. 2025. Level Generation with Constrained Expressive Range. In *International Conference on the Foundations of Digital Games (FDG '25), April 15–18, 2025, Graz, Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3723498.3723845

1 Introduction

In the field of procedural content generation for game levels [18], expressive range has long been the primary tool for qualitative exploration. In procedural content generation via machine learning, expressive range is used to compare the range of a trained content generator to that of the original corpus, with the goal being for the generator to replicate the range properties of the original dataset. Expressive range examines the variety of content that can be created by a generator, according to selected "metrics" that can be computed about the content, commonly using these metrics to place content into discrete "cells" in a two-dimensional histogram.

FDG '25, Graz, Austria

ACM ISBN /25/04
https://doi.org/10.1145/3723498.3723845

Inspired by the quality diversity (QD) paradigm [4], we highlight the importance of first finding diverse solutions and then improving their quality. This idea comes from the QD principle of rewarding diversity to uncover paths toward high-performing areas in the search space [5].

In this work, we use the expressive range of a generator as this conceptual space of potential creations and extend it to explore unique and interesting level designs. To achieve this, we utilize a prioritized random selection to explore underrepresented cells in a two-dimensional expressive range defined by density and difficulty. We employ a constraint-based level generator to create examples from these underrepresented cells.

By guiding the generator toward unexplored regions of the expressive range, our approach encourages the creation of levels that differ from those typically produced. This not only helps diversify the generated content but also enables a deeper understanding of the generator's capabilities and limitations.

The contributions of this paper are:

- Utilizing the expressive range space as the conceptual space of possible game creations.
- Developing a pipeline for sampling from cells of expressive range space.
- Discovering unique and interesting levels by exploring the extended expressive range of the corpus.

2 Related Work

2.1 Expressive Range Analysis

The concept of expressive range was first introduced by Smith and Whitehead [15] as a means of visualizing generative spaces and assessing the variety and quality of levels produced by generators. They employed metrics such as linearity, leniency, and density to analyze the output tendencies of their system, focusing on characteristics like the types of levels that are likely to be generated and those that are difficult or impossible to produce.

Building on this foundation, Summerville [17] introduced corner plots to visualize multiple metric pairs simultaneously, overcoming the previous limitation of only being able to examine two metrics at a time.

Traditional expressive range and corner plots have become fundamental methods for evaluating procedural content generation systems. For instance, Kreminski et al. [8] utilized expressive range as an evaluation metric to determine how effectively a creative tool supports a wide variety of outputs and how well the interface facilitates both system-initiated and user-driven design choices. Similarly, Alvarez et al. [1] employed expressive range to evaluate the diversity and variety of solutions generated within the feature space of the map-elites algorithm. Summerville and Mateas [19]

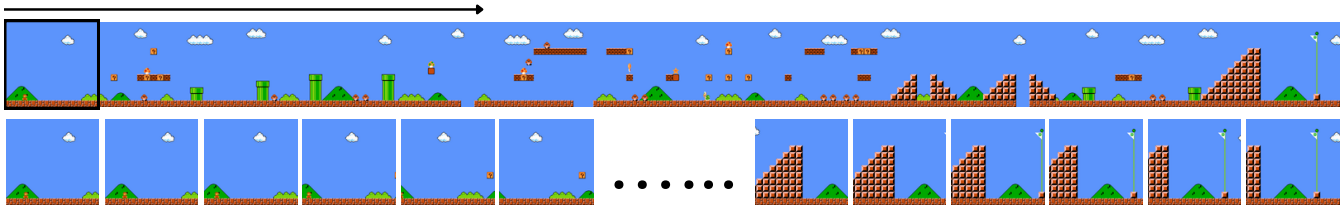


Figure 1: Sliding window horizontally across the level 1-1 of Super Mario Bros. to create the training set.

utilized corner plots to illustrate the expressive range of individual metrics along the diagonal and the interplay of metric pairs in the off-diagonal cells. This approach allowed the authors to evaluate whether the generated levels mirrored human-designed levels in specific metrics and preserved a similar overall distribution of characteristics.

The effectiveness of these evaluations, however, can depend on the chosen metrics. There has been some work to enhance this approach (for example Withington and Tokarchuk [21]) by focusing on better capturing meaningful variations in generated content by refining metric selection. By choosing metrics that are relevant and aligned with the intended creative outcomes, it can be possible more accurately and insightfully evaluate PCG systems.

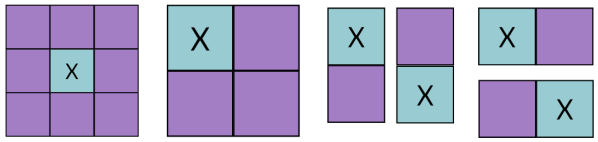
Moreover, other research has tried to define new evaluations that are more dynamic than expressive range. For example, Hervé et al. [7] introduced generative shift analysis as a method for evaluating and comparing procedural content generators. This approach emphasizes analyzing shifts in output distributions when the generative system or its parameters are altered, which is in contrast to the expressive range, which measures content variety through static scores.

Expressive range analysis has been used in design in a few studies. For example, during the development of the game That's Me TV, Guzdial et al. [6] used expressive range visualization to adjust the game's mechanics until they matched his design goals. Similarly, Madkour et al. [9] used expressive range as a way for designers to directly interact with their probabilistic graph grammar system, allowing them to define their desired generative space by selecting a region on a plot.

2.2 Quality Diversity

Quality Diversity (QD) algorithms are optimization methods designed to find many high-quality solutions that are also diverse [5]. Unlike traditional methods that focus on a single best solution, QD algorithms aim to explore a wide range of possibilities. They reward both quality (how well a solution performs) and diversity (how different the solutions are from each other). By encouraging exploration and ensuring quality through techniques like local competition or constraints, QD algorithms are especially useful for creative tasks [13].

The expressive range, representing the conceptual space of possible levels that can be generated, creates a partitioned space where diversity can be enforced within a grid of cells. Each cell represents a distinct area in the behavior space, characterized by specific



(a) *ring* (b) *block2* (c) *nbr-plus*

Figure 2: Pattern templates. The blue tiles are "input" tile locations, and the purple are "output" tile locations; a certain input tile constrains the corresponding output tiles to be those that were seen in the example level(s). Note that `diamond` `ring` and `block2` jointly constrain the output tiles all at once, while `nbr-plus` constrains each output tile independently.

properties, and contains all individuals that exhibit certain behaviors. Then the simplest way to enhance the quality is using local competition between individuals within the same niche.

3 Methodology

3.1 Metric Pair Selection

We use the two scores of density-difficulty to analyze the 2D expressive range of a corpus of levels. This score combination has been widely adopted in previous works [12, 14]. We implemented the same definition of these scores from previous work:

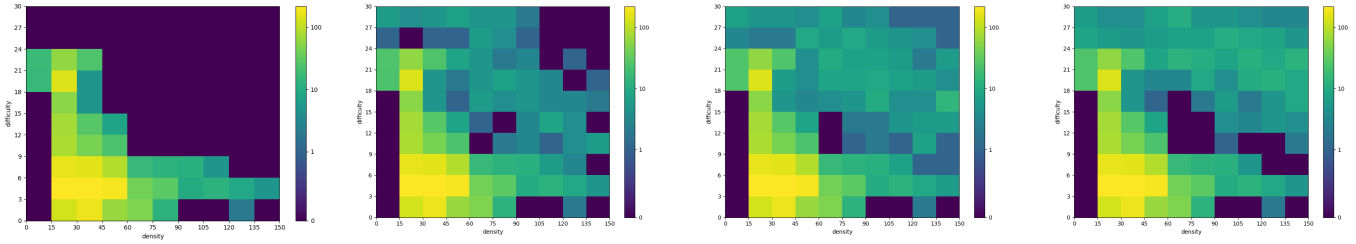
Density: the number of tiles in a segment that aren't background tiles.

Difficulty: the number of tiles of a segment that are occupied by any enemy or are hazard tiles (including gaps).

3.2 Constraint-Based Generator

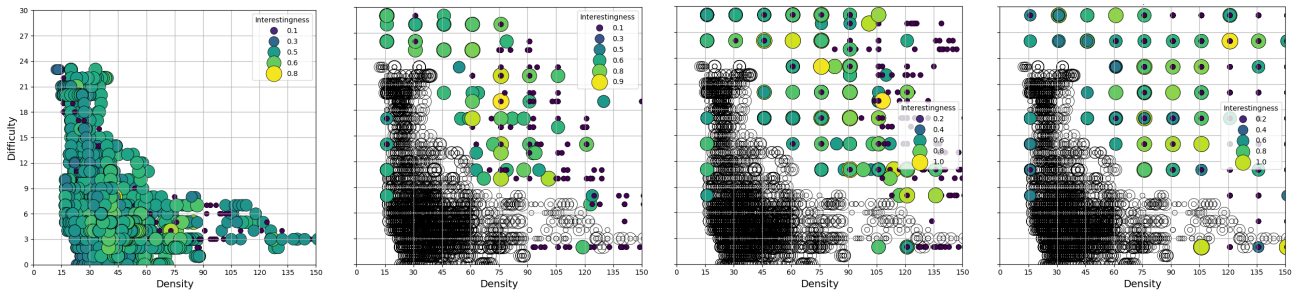
In this work, we use the Sturgeon constraint-based level generator [2] as a dependable tool for procedural generation. Sturgeon leverages a straightforward, mid-level API to define constraints on Boolean variables, which it translates into low-level constraint satisfaction problems. As a result, Sturgeon ensures that the generated levels meet the specified constraints precisely, without any errors or noise.

Sturgeon enforces local tile patterns as constraints, making sure the generated levels replicate the local structures of example levels. This is achieved by using pattern templates that create levels by using rules that guide how tiles should be arranged, rather than placing them randomly. Each pattern template consists of a set of input and output patterns, where specific input patterns limit the types of output patterns that can appear. In this work, we experimented with



(a) *initial corpus* (b) *extended corpus - ring* (c) *extended corpus - block2* (d) *extended corpus - nbr-plus*

Figure 3: Density-difficulty expressive range of expanded corpus with each pattern template. The color bar is based on SymLogNorm allowing linear behavior around zero and logarithmic beyond.



(a) *initial corpus* (b) *extended corpus - ring* (c) *extended corpus - block2* (d) *extended corpus - nbr-plus*

Figure 4: Scatter plot displaying the (normalized) interestingness of generated levels across the expressive range, with the size and color of each point reflecting its level of interestingness. Uncolored points represent levels from the initial corpus in extended corpuses.

Pattern Template	Total Attempts	Successful Attempts	Failed Attempts	Timed Out Attempts	Average Solve Time (s)	Average Fail Time (s)	Average Time (s)
`ring`	200	177	2	21	150.09	73.73	374.61
`block2`	294	282	2	10	8.36	19.97	309.44
`nbr-plus`	315	313	2	0	0.98	30.78	10.57

Table 1: Comparison of performance of different pattern templates in terms of successful attempts (resulted in level generation) within a fixed time limit.

different pattern templates (`ring`, `block2`, and `nbr-plus`; described below) to analyze their performance in exploring the generator's expressive range.

For this work, we used Super Mario Bros. [10] levels from The VGLC [20] to extract these patterns. Additionally, Sturgeon can impose constraints on overall tile counts, allowing the generated levels to maintain a similar overall appearance to the example. We also utilized these constraints on overall tile counts to enforce specific metric values to sample from the 2D expressive range of possible game creations.

4 Experiments

4.1 Domain

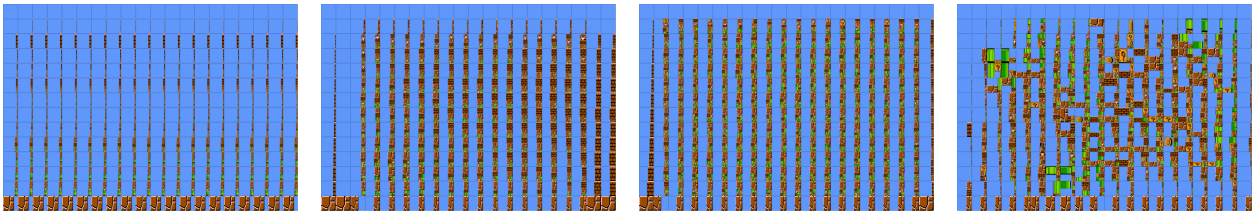
To create the initial corpus of Super Mario levels, we applied a sliding window of size 20×14 horizontally across the level, producing 2,302 smaller segments with 11 tile types. Figure 1 shows a visualization of this process.

4.2 Exploring the Expressive Range Space

To apply Sturgeon's capability of imposing constraints on overall tile counts, we need to frame the density-difficulty constraint as a tile count problem for Sturgeon. This is straightforward for both density and difficulty metrics, as they are computed based on tile counts. We aim to satisfy the constraint that the total number of non-background tiles equals the desired density, while the sum of enemy and hazard tiles corresponds to the desired difficulty.

In this work, we experimented with the `ring`, `block2`, and `nbr-plus` pattern templates. Figure 2 illustrates the `ring` pattern template is the most constrained template between these three followed by `block2` and then `nbr-plus`. More constrained pattern templates (e.g. `ring`) impose greater structure on level generation, making it harder to fill gaps and create complete levels. Therefore, a trade-off exists between maintaining structure and finding feasible solutions when selecting different pattern templates.

To compare the different pattern templates based on their speed in exploring the expressive range space, we allocated the same



(a) *initial* (b) *ring* (c) *block2* (d) *nbr-plus*

Figure 5: Visual exploration of initial levels of the corpus and the generated levels with different pattern templates.

amount of time (12 hours) for all three pattern templates. The exploration process uses a prioritized random selection that focuses on underrepresented cells within the space — cells containing fewer than 10 levels. At each step, cells with the fewest levels are selected from uniformly at random. Underrepresented cells can be selected multiple times until they reach the 10-level threshold. If an attempt to generate a level in a cell fails, or if the cell has already reached the limit, it is added to a blocklist. This prevents the process from getting stuck in cells that are either impossible to generate or have already met their level count.

A pattern template that successfully generates a level meeting the density-difficulty constraint, or identifies the absence of solutions for that setting, can explore the space more efficiently. In addition to the overall allocated time, we have set a 15-minute timeout for individual level generation attempts to prevent solvers from getting stuck on more challenging density-difficulty settings.

This approach is somewhat similar to reversing the typical flow of quality diversity searches. Instead of generating levels and hoping they cover a wide expressive range, we systematically explore the space to ensure full coverage. To investigate this further, we also run the constraint-based generator without any specific constraints, allowing it to create levels randomly, without considering density or difficulty. We then compare how well these purely random levels cover the density-difficulty space.

To assess the quality of the generated levels, we introduce a quantitative measure of "interestingness". Using Sturgeon, we determine the player's path in each generated segment and examine two key factors: path length (which reflects gameplay duration) and the number of jumps (which reflects the player's engagement). By adding these values together, we estimate how interesting the levels in each explored cell are. This quantitative measure is inspired by gameplay features detected in previous works, that significantly correlate to being "fun"[11].

The definition of "interestingness" is not a definitive measure of the quality of the generated levels, but it provides insight into which levels are engaging and playable, rather than bland or unplayable.

5 Results

5.1 Exploring the Expressive Range Space

As anticipated, the various template patterns demonstrated differing performances in exploring the expressive range of games. Figure 3 illustrates how effectively each pattern explored the space. The `ring` pattern, due to its more restrictive constraints, was less successful in covering all cells. However, as the patterns relaxed these constraints, their ability to find level solutions for each cell improved.

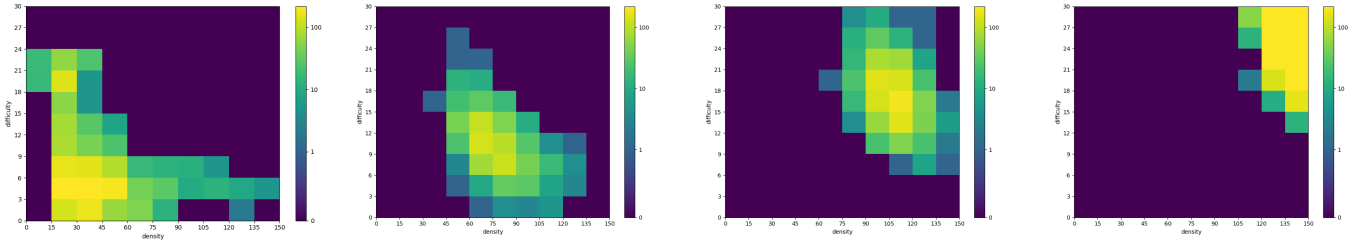
As shown in Table 1, the `nbr-plus` pattern template achieved the lowest solve time, while the `ring` pattern template had the highest. This outcome supports the expectation that less-constrained pattern templates are faster at finding solutions, highlighting the trade-off between maintaining an ideal structure and optimizing speed. This trade-off is further evident in the samples generated during the exploration of the expressive range, as shown in Figure 8.

Comparing the expanded expressive range of the systematic traversal in Figure 3 with the fully random level generation in Figure 6 highlights a significant difference in coverage between these two approaches. Interestingly, random generation for each pattern template tends to cover entirely different areas of the expressive range. This is likely because different pattern templates are better suited to capturing specific structures in levels, making them more effective for generating different densities and difficulties.

Additionally, we compare the (normalized) interestingness of levels generated through systematic traversal (Figure 4) with those created via random generation (Figure 7). The interestingness values have been normalized between the minimum and maximum values across all values to make sure the figures are easily comparable. We observe some differences. For instance, it is clear that the `nbr-plus` pattern template struggles to produce levels that are both playable and interesting without guidance on the appropriate proportions of density and difficulty — or, more broadly, without additional constraints on the level.

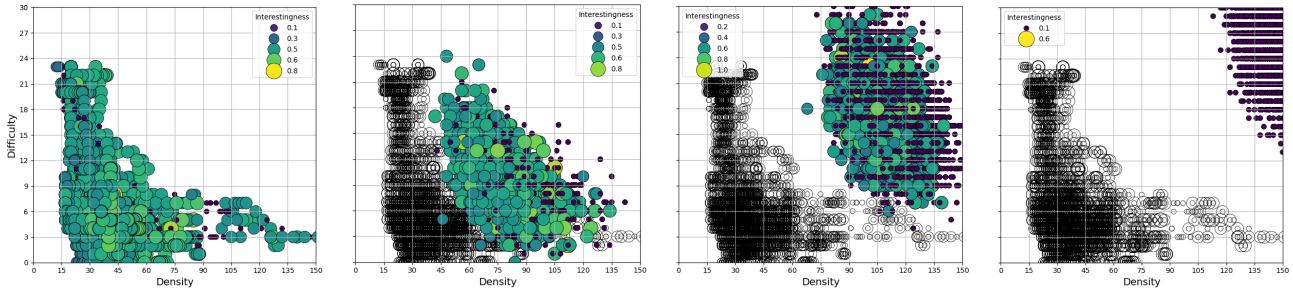
5.2 Visual Exploration

We used the visual explorer tool designed by Cooper et al. [3] to provide a quick visual summary of both the initial and extended corpora. Inspired by Oskar Stålberg [16], this tool represents all available tiles within each cell, scaled according to their frequency. Comparing the visual exploration of the initial levels of the corpus and the generated levels with different pattern templates in Figure 5 shows some immediate differences between the levels that come from underrepresented cells of expressive range and the rest. For example, the levels generated for underrepresented cells show noticeable variations in pipe positions and heights. This trend is more noticeable in patterns that are less restrictive (like `nbr-plus`) as they allow more uncommon structures in levels.



(a) *initial* (b) *ring* (c) *block2* (d) *nbr-plus*

Figure 6: Density-difficulty expressive range of RANDOM generated corpus with each pattern template. The color bar is based on SymLogNorm allowing linear behavior around zero and logarithmic beyond.



(a) *initial* (b) *ring* (c) *block2* (d) *nbr-plus*

Figure 7: Scatter plot displaying the (normalized) interestingness of complete randomly generated levels across the expressive range, with the size and color of each point reflecting its level of interestingness. uncolored points represent levels from the initial corpus in extended corpuses.

6 Discussion

This work is an initial attempt to create a systematic method that allows Sturgeon to explore its expressive range in a controlled manner, using an approach inspired by quality diversity, but with full control over the exploration. In the future, we plan to build on this by adding more complex metrics, like linearity, which go beyond counting tiles and are harder to define as a satisfaction problem for the generator.

We believe that this controlled exploration of the expressive range can offer value in the context of generative model training. By curating specific datasets through this approach, we can ensure that the data used for training is more diverse and tailored to particular characteristics or qualities.

7 Conclusion

In this work, we use the expressive range of a level generator as a conceptual space to explore potential level designs. To investigate underrepresented areas in terms of density and difficulty, we employed a prioritized random selection strategy with a constraint-based generator. By applying different pattern templates to learn from initial levels, we evaluated the efficiency of level generation based on time, success and failure rates, and the interestingness of the generated levels. This approach allows us to interact with the generator, covering the expressive range systematically (rather than randomly) to produce diverse and interesting game levels.

Acknowledgments

Support was provided by Research Computing at Northeastern University (https://rc.northeastern.edu/) through the use of the Discovery Cluster.

References

[1] Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. 2020. Interactive constrained map-elites: Analysis and evaluation of the expressiveness of the feature dimensions. *IEEE Transactions on Games* 14, 2 (2020), 202–211.

[2] Seth Cooper. 2022. Sturgeon: Tile-Based Procedural Level Generation via Learned and Designed Constraints. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 18, 1 (Oct. 2022), 26–36. https://doi.org/10.1609/aiide.v18i1.21944

[3] Seth Cooper, Faisal Abutarab, Emily Halina, and Nathan R Sturtevant. 2023. Visual Exploration of Tile Datasets. In *EXAG@ AIIDE*.

[4] Li Ding, Jenny Zhang, Jeff Clune, Lee Spector, and Joel Lehman. 2024. Quality diversity through human feedback: towards open-ended diversity-driven optimization. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 441, 19 pages.

[5] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. 2019. Procedural Content Generation through Quality Diversity. *2019 IEEE Conference on Games (CoG)* (2019), 1–8. https://api.semanticscholar.org/CorpusID:195848208

[6] Matthew Guzdial, Devi Acharya, Max Kreminski, Michael Cook, Mirjam Eladhari, Antonios Liapis, and Anne Sullivan. 2020. Tabletop Roleplaying Games as Procedural Content Generators. In *Proceedings of the 15th International Conference on the Foundations of Digital Games* (Bugibba, Malta) *(FDG '20)*. Association for Computing Machinery, New York, NY, USA, Article 103, 9 pages. https://doi.org/10.1145/3402942.3409605

[7] Jean-Baptiste Hervé, Oliver Withington, Marion Hervé, Laurissa Tokarchuk, and Christoph Salge. 2023. Exploring Minecraft Settlement Generators with Generative Shift Analysis. *arXiv preprint arXiv:2309.05371* (2023).

[8] Max Kreminski, Isaac Karth, Michael Mateas, and Noah Wardrip-Fruin. 2022. Evaluating Mixed-Initiative Creative Interfaces via Expressive Range Coverage Analysis.. In *IUI Workshops*. 34–45.
[9] Abdelrahman Madkour, Stacy Marsella, and Casper Harteveld. 2022. Towards Non-Technical Designer Control over PCG Systems: Investigating an Example-Based Mechanism for Controlling Graph Grammars. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (Athens, Greece) *(FDG '22)*. Association for Computing Machinery, New York, NY, USA, Article 39, 12 pages. https://doi.org/10.1145/3555858.3555895
[10] Nintendo. 1985. Super Mario Bros. Game [NES].
[11] Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. 2009. Modeling player experience in Super Mario Bros. In *2009 IEEE Symposium on Computational Intelligence and Games*. 132–139. https://doi.org/10.1109/CIG.2009.5286482
[12] Anurag Sarkar and Seth Cooper. 2020. Sequential segment-based level generation and blending using variational autoencoders. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*. 1–9.
[13] Anurag Sarkar and Seth Cooper. 2021. Generating and Blending Game Levels via Quality-Diversity in the Latent Space of a Variational Autoencoder. In *Proceedings of the 16th International Conference on the Foundations of Digital Games* (Montreal, QC, Canada) *(FDG '21)*. Association for Computing Machinery, New York, NY, USA, Article 3, 11 pages. https://doi.org/10.1145/3472538.3472545
[14] Anurag Sarkar, Zhihan Yang, and Seth Cooper. 2020. Controllable level blending between games using variational autoencoders. *arXiv preprint arXiv:2002.11869* (2020).
[15] Gillian Smith and Jim Whitehead. 2010. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (Monterey, California) *(PCGames '10)*. Association for Computing Machinery, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/1814256.1814260
[16] O. Stålberg. 2018. Wave Function Collapse in Bad North. https://www.youtube.com/watch?v=0bcZb-SsnrA YouTube video.
[17] Adam Summerville. 2018. Expanding Expressive Range: Evaluation Methodologies for Procedural Content Generation. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 14, 1 (Sep. 2018), 116–122. https://doi.org/10.1609/aiide.v14i1.13012
[18] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270.
[19] Adam J. Summerville and Michael Mateas. 2016. Super Mario as a String: Platformer Level Generation Via LSTMs. https://dl.digra.org/index.php/dl/article/view/752. In *Proceedings of DiGRA/FDG 2016 Conference*. DiGRA, Tampere.
[20] Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Ontañón Villar. 2016. The VGLC: The Video Game Level Corpus. *Proceedings of the 7th Workshop on Procedural Content Generation* (2016).
[21] Oliver Withington and Laurissa Tokarchuk. 2023. The Right Variety: Improving Expressive Range Analysis with Metric Selection Methods. In *Proceedings of the 18th International Conference on the Foundations of Digital Games* (Lisbon, Portugal) *(FDG '23)*. Association for Computing Machinery, New York, NY, USA, Article 18, 11 pages. https://doi.org/10.1145/3582437.3582453

A Appendix

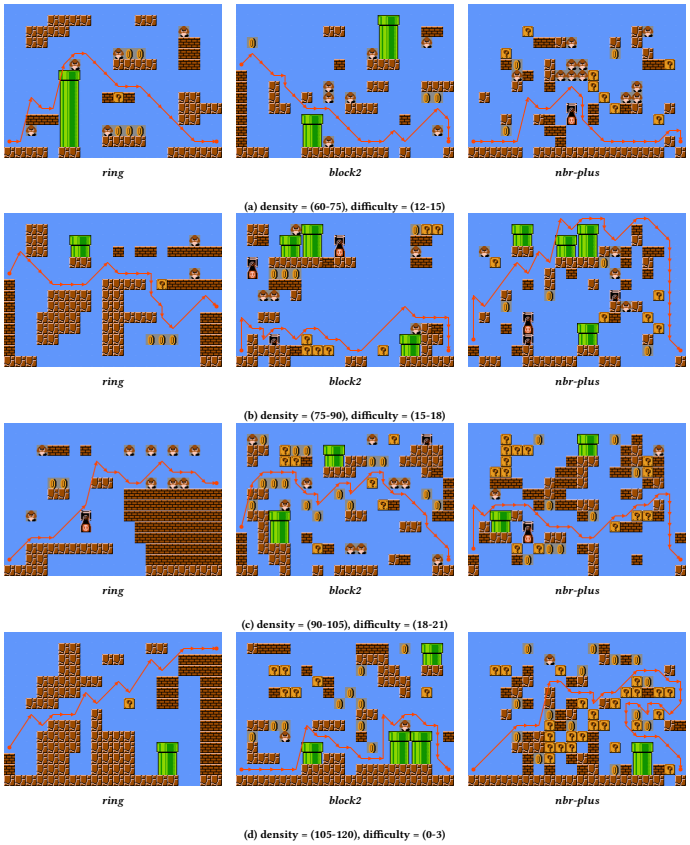


Figure 8: Samples of generated levels with constraint-based level generator (Sturgeon) while exploring the same under-represented cells in expressive range space with different pattern templates.