2022 IEEE 8th Information Technology International Seminar (ITIS)
Surabaya, Indonesia, October 19 – 21, 2022

Using Genetic Algorithm for Wide yet Even Scattering of Game Objects: Applications on Irregular Levels and Involving Multiple Objects

Pratama Wirya Atmaja
Department of Informatics
University of Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
pratama_wirya.fik@upnjatim.ac.id

Sugiarto
Department of Data Science
University of Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
sugiarto.if@upnjatim.ac.id

2022 IEEE 8th Information Technology International Seminar (ITIS) | 979-8-3503-9819-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITIS57155.2022.10010220

***Abstract*—The rising costs of game development necessitate algorithmic and automatic methods for creating game content, also known as procedural content generation or PCG. Among many types of content, game levels are increasingly generated with PCG. The literature has recorded various level generation methods for specific games or contexts, yet no general-purpose one. We have previously addressed this gap by proposing a genetic algorithm-based method that scatters objects widely yet evenly on a level. The algorithm satisfies the player's needs for exploration. Our previous work acquired good results when applying the method on a simple and rectangular level. This paper presents two follow-up case studies that reflect real games: one with irregular levels and another with multiple object types. Our results show that our method fits the first case study; however, the irregular level's layout may yield unexpected results, such as the tendency of objects to appear along its edge. Meanwhile, we also find that objects of different types are best scattered separately, each under a specific parameter, to optimize their wide yet even distributions. We then discuss our results' implications for PCG research and how to generalize our method to generate other types of game content.**

Keywords—procedural content generation, game levels, object scattering, genetic algorithm, wide yet even distribution.

I. Introduction

A. Background and Problem Statement

Digital games have progressed rapidly worldwide [1], becoming the largest entertainment industry and penetrating many aspects of society, including education [2]. On the other hand, the growth has led to skyrocketing development costs [3], of which game content production is a significant part. Manual labor cannot keep up with the ever-increasing demand for content quantity and quality, necessitating algorithmic approaches backed by computing power, i.e., *procedural content generation* or PCG [4].

"Game content" refers to consumable parts of a game, ranging from directly-perceived ones such as animations and music to those buried in the code, e.g., game rules [5]. One of the essential content types is levels, which are distinct parts of a game world containing arrangements of game objects [6]. Initially practiced only in specific game genres such as roguelikes [7], procedurally generating levels for seemingly-infinite replayability is increasingly accepted in modern games [8]. It is also the most researched PCG subtopic [9].

To reach its goal, PCG for levels tries to acquire many unique levels that satisfy quality constraints. For that reason, it intersects with *stochastic optimization problems* [10], for which meta-heuristic approaches such as evolutionary algorithms (EAs) are popular choices. Indeed, from the regular genetic algorithm (GA) [11] to state-of-the-art variants like FI-2Pop and MAP-Elites [12][13], EAs have featured extensively in PCG literature.

So far, level generation methods are mostly specific to games and game genres, two of the most popular being platformers and roguelikes [9]. While research on cross-domain methods is underway [14], PCG for levels still relies on resources such as available game codebases [9], which puts less popular game genres at a disadvantage. Nevertheless, previous research has identified general quality metrics for levels, including area control, leniency, and exploration [15][16][17]. The third metric concerns the size of the area the player must visit to achieve their objectives [15][18]. Generally, the larger the area, the more fun the game is for the player.

We have previously researched a genreless way to scatter objects to optimize the exploration metric [19]. Our GA-based method always produces a wide yet even object distribution, i.e., it optimizes distances between the objects and overall space coverage. Our previous work empirically validated the method's viability in simple use cases involving one object type and a rectangular level. However, levels in games are typically more complex than that: they often have irregular shapes and contain various objects with different functions. Whether our method can optimally handle such levels should be investigated. Therefore, this paper reports our follow-up experiment on applying the method to such levels.

B. Contributions

Alongside our previous one, this work contributes a GA-based level generation method that works on any game genre, filling a clear gap in the PCG literature, which is still dominated by game- and genre-specific PCG methods. Additionally, this work presents principles for effective applications of the method on irregular levels and for scattering multiple object types, two use cases commonly found in games, therefore offering value to researchers and game developers.

II. PCG Method Specification

A. Specification of the Problem

In the context of our method, the "object scattering" problem is defined as the random placement of objects on discrete and two-dimensional "position slots." Regarding the exploration metric, we have established two quality criteria for the placement, the first being that visiting every placed object should require traveling a long distance. The second criterion is that the placement should equally favor every slot, meaning that repeated placements should not reveal any discernible object distribution pattern. One way to validate the criterion is by analyzing the slots' occupancies throughout the placements: a high variance between the occupancies indicates a pattern's existence.

Formally, the problem concerns *bijections* between a set O of objects and subsets of the set P of position slots. More specifically, some sets and a function are given: sets O and P where $|P| > |O|$, a set J of every $K \subset P$ where $|K| = |O|$, and a distance function D that maps J to a set R of all positive real numbers. With the sets and the function, generate a set A of every bijection G between O and $N \subset P$ such that:

1) For each G, $N \in$ an ordered pair $W \in D$ such that W has a high second tuple, i.e., the distance value; and

2) Given a function F that maps P to a set I of all positive integers according to how many times each position slot appears in every N, the integers must have a low variance.

B. Working Principles of the Method

As previously explained, our method is based on regular GA [20]. An exact yet brute-force method for object scattering that satisfies the quality criteria would have to check every result of the partial permutation between the position slots and the objects. With a moderate-sized set of slots (numbering at hundreds) and a reasonable set of objects (numbering at tens), the permutation count already surpasses 10 to the power of 100. Therefore, basing our method on GA is justified.

Each individual in the method's population is a position slot array, with objects occupying some of its cells. Equation (1) shows the fitness function, where a is the average distance between pairs of objects, thus representing the first quality criterion. However, maximizing a can result in the objects being placed only on the level's periphery, thus yielding an uneven distribution pattern that violates the second criterion. The function prevents it by balancing a with the object pair distances' standard deviation (s) multiplied by m. The right value of m pulls the objects to the center of the level while only slightly narrowing their distribution. Meanwhile, the longest distance between a pair of objects (r) decreases the function's output value to make it more manageable[1].

$$f = \frac{a - (s \cdot m)}{r} \tag{1}$$

[1] Our previous work incorrectly stated that r normalized the output value.

C. Known Characteristics

Our previous work's experiment applied our method to scatter objects on a rectangular level containing uniformly distributed position slots. We employed various dimensions of the level determined by the number of slots and the space between pairs of slots.

Our method's GA algorithm implemented the following parameters in the experiment. First, the algorithm ran for 100 iterations, and its population consisted of 40 individuals. Unlike regular GA, the algorithm did not shuffle its population as well as let mutated individuals produce any offspring; instead, the individuals replaced their old selves in the population. The algorithm evenly split the population into high-fitness and low-fitness parts. Ten random individuals from the low-fitness part and four others from the high-fitness one mutated in each iteration. The mutation process selected several objects in an individual and randomly repositioned them to unoccupied position slots. The algorithm never mutated the individual with the highest fitness value to prevent the value from decreasing.

In the experiment, we applied the method in batches, each comprising 400 executions of the algorithm. Fig. 1 shows how increasing the method's multiplier value affects the standard deviation of the occupancies of position slots in a 15x10 rectangular level receiving ten objects. The higher the multiplier, the lower the standard deviation until it hits a minimum at a 2.8 multiplier value. Thus, the value is the best since it optimizes the second quality criterion. Fig. 2 shows a heatmap of position slot occupancies under such an optimum multiplier value. Although the object distribution is not entirely optimal (as evidenced by a subtle pattern on the heatmap), it is the best our method can produce on a simple level with uniformly distributed position slots.

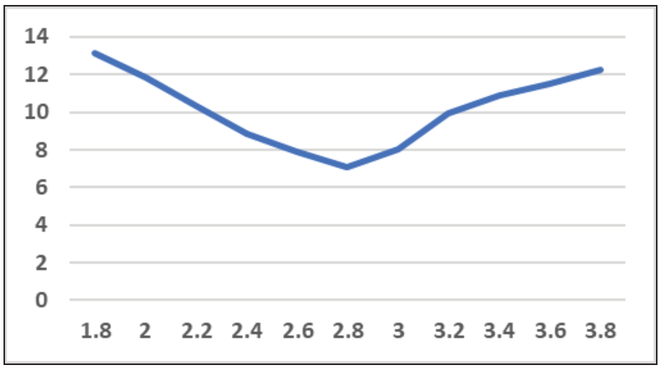

Fig. 1. The standard deviations of position marker occupancies in generation process batches with 15x14 position markers, 200 pixels spaces, ten scattered objects in each process, and 1.8 to 3.8 multiplier values

The experiment yielded two other interesting facts. First, the best value of the multiplier depends on the number of placed objects but not on the number of position slots or the inter-slot space. Fig. 3 shows the function between the number of objects and the best multiplier value. Equation (2) describes the function, which was approximated using an online curve fitting service[2]. Second, there is a *negative linear* relationship between the multiplier value and the average distance of sequentially visiting all objects: the higher the multiplier, the shorter the average distance.

[2] https://mycurvefit.com/

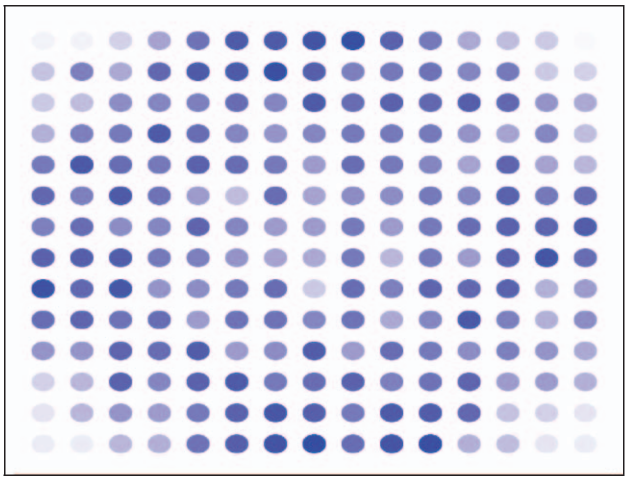

Fig. 2. An ideal heatmap of the occupancies of position slots on a 15x14 rectangular level

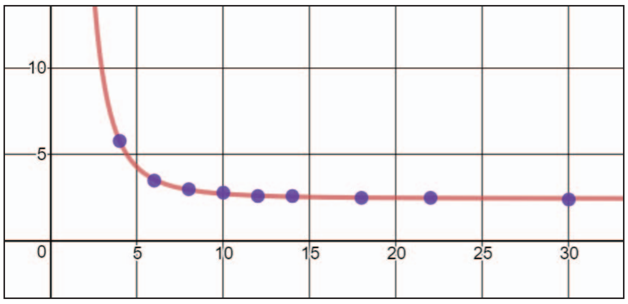

Fig. 3. The number of objects (x-axis), the best multiplier (y-axis), and the estimated function between them (the curve)

$$y = 2.45 + \frac{7045779}{1 + \left(\frac{x}{0.02}\right)^{2.71}} \tag{2}$$

III. Methodology of Follow-up Experiment

We will describe the specifications of two case studies in our follow-up experiment and our research questions.

A. *Specification of Case Study 1*

In the first case study, we applied the method to scatter objects on two irregular levels, each reflecting a realistic use case in games such as roguelikes. We scattered six, 10, and 20 objects on each level in batches of 400 applications of the method.

Fig. 4 shows the design of the first irregular level. It consists of three connected areas, with one area visibly larger than the others. Such a level can serve multiple purposes, such as facilitating an intense combat session. Meanwhile, Fig. 5 shows the second irregular level, which features several equally-small areas that can be connected with narrow corridors. Roguelikes such as Nethack [7] heavily feature procedurally-generated variations of such a level [18][21].

B. *Specification of Case Study 2*

Our second case study was on scattering objects of different types. The game literature recognizes many object types, such as player avatars, collectible items, and obstacles [22], which differ in function and thus may require separate treatments. For example, Ben Weber's Probabilistic Multipass Generator placed object groups in *Infinite Mario Bros*, a PCG-based variant of *Super Mario Bros*, in separate stages, e.g., platforms were placed first in the level, followed by collectible items [23]. Such a multi-stage placement process can ensure the correct spread of each object type and prevent it from clustering in a small area. Therefore, our second case study explored multi-stage object scattering processes, with each stage applying our method once to scatter one object type. The processes were done on the rectangular level from our previous work, and we limited the number of object types to three. Each stage would scatter 10 objects on the level; thus, each multi-stage process would fill the level with 30 objects in total. Each stage would employ a 2, 2.4, 2.8, 3.2, or 3.6 multiplier value, and different stages could apply the same value, allowing for 125 possible combinations of multiplier values. We would try all combinations to find the best one.

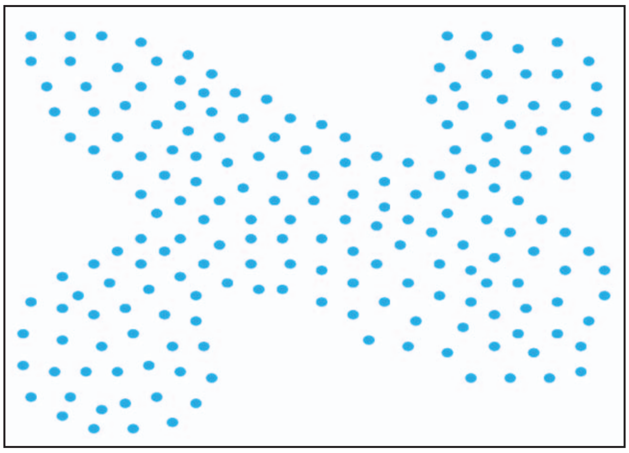

Fig. 4. The first irregular level for case study 1

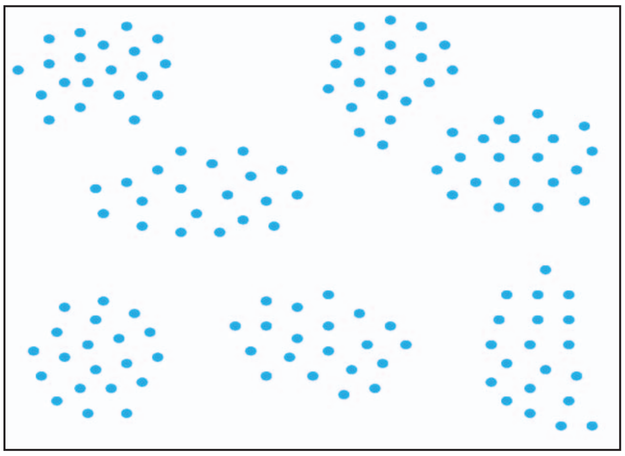

Fig. 5. The second irregular level for case study 1

C. *Research Questions*

To get concrete and quantitative results, we formulated the following research questions (RQs):

RQ1) Do optimal applications of the method produce different distribution patterns on irregular maps?

RQ2) Assuming the same object number, does the method require different multiplier values to scatter objects optimally on irregular levels?

RQ3) What is the relationship between the method's multiplier value and the standard deviation of its position slot occupancies? Is the relationship similar to that between the multiplier value and the standard deviation of a rectangular level's slot occupancies?
RQ4) What are the relationships between the average distance of sequentially visiting all objects in the irregular maps and other variables?
RQ5) How do the three-stage and single-stage scattering processes differ in object distribution patterns?
RQ6) What is the optimal multiplier value for each stage in a three-stage process?

IV. Results and Discussions

A. *Results of Case Study 1*

Fig. 6 and 7 show heatmaps of the occupancies of position slots in the first and second irregular levels, which show the optimal, wide-yet-even patterns after our method filled the levels with ten objects 400 times. Overall, the pattern on each level resembles the rectangular level's best one (Fig. 2): near-even except on some of the level's outermost areas. This finding answers RQ1 and validates our method's feasibility for irregular levels. However, (1) may not always give accurate multiplier values for the levels, e.g., the heatmap in Fig. 7 required a 2.6 instead of a 2.8 multiplier value. Therefore, the answer to RQ2 is, "It depends on the irregular level."

Regardless of the answer to RQ1, the irregular levels' heatmaps exhibit a unique pattern: some position slots on the levels' edges tend to be occupied more frequently than others. This pattern seems minor as it involves only a few slots; nevertheless, it may interest anyone applying our method.

Fig. 8 to 10 show the relationship between the multiplier value and position slot occupancies' standard deviation in case study 1 (RQ3) when the object number is 10, 20, and six. Interestingly, the standard deviation of level 2's position slot occupancies is much more volatile than that of level 1. We suspect level 2's heavily segmented layout to be the cause: since empty, slot-less space can be found everywhere in the level, objects tend to "jump" between disjoint areas instead of spreading smoothly across the level.

However, Fig. 9 and 10 also reveal something interesting: with the same level and object number, the method may yield multiple *minima* of slot occupancies' standard deviation. The phenomenon seems to depend on the level's layout since it only occurred on the first irregular level. We suspect that the level's layout, with a large blank space around its center, may prevent objects from clustering around the center when a high multiplier value is used. In such a scenario, some of the objects could have landed on farther position slots due to being unable to be placed around the center, causing the object distribution to resemble one under a lower multiplier value.

Lastly, regarding RQ4, we observed the same negative linear relationship as in our previous work: the higher our method's multiplier value, the shorter the average distance of sequentially visiting all objects in the level. This relationship was consistent across every irregular level and object number. In theory, the previous "multiple minima" phenomenon may disrupt the relationship since, when our method fails to put objects around the level's center, the average distance may become longer than it should be. However, our analysis did not reveal such a disruption.

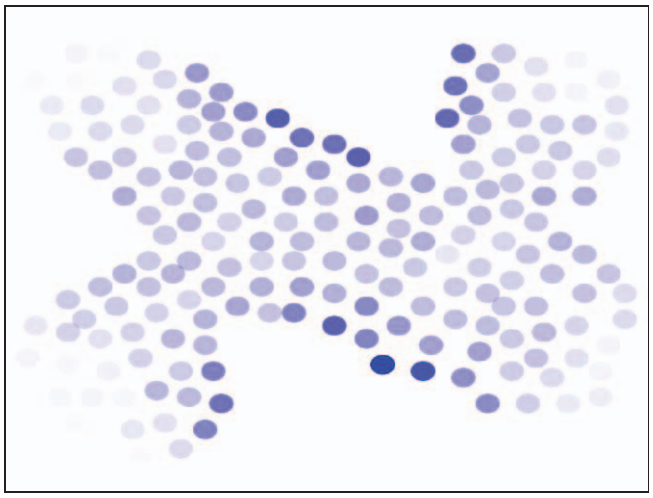
Fig. 6. An example of an ideal heatmap of the occupancies of position slots on the first irregular level when ten objects are scattered on it in 400 repetitions

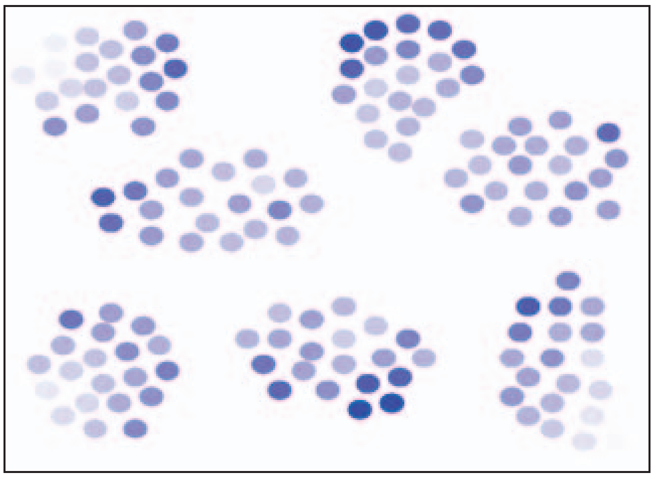
Fig. 7. An example of an ideal heatmap of the occupancies of position slots on the second irregular level when ten objects are scattered on it in 400 repetitions

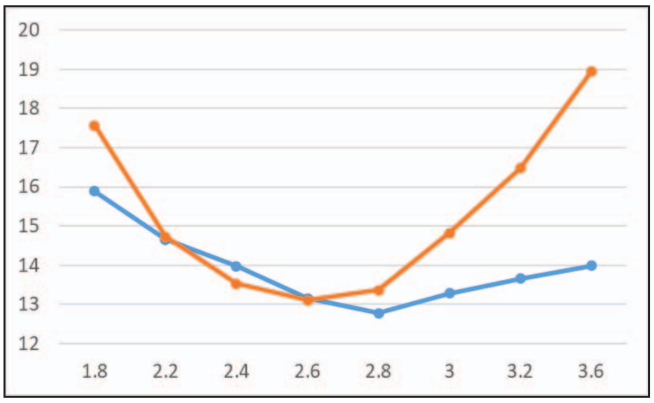

Fig. 8. The relationship between the multiplier value (x-axis) and position slot occupancies' standard deviation (y-axis) in case study 1 when the object number is ten. The blue line belongs to the first irregular level, whereas the orange line belongs to the second level.

B. *Results of Case Study 2*

Regarding RQ5, with the right combination of multiplier values, the multi-stage object placement process can surpass the single-stage one in effectiveness. The heatmap in Fig. 11 proves it by showing an object distribution pattern that is wider and more even than that in Fig. 2. We acquired such a pattern under the multiplier values of 2, 2.8, and 3.6 regardless of which stage applied which value. Thus, the order of the values does not seem to matter. Out of the 125 combinations of multiplier values, those involving the three values did produce the lowest slot occupancies' standard deviation.

Regarding RQ6, the multiplier value of 2 can make some objects populate the level's outer area, which a single-stage scattering process with the best multiplier value (e.g., 2.8 if the process scatters ten objects) seldomly targets. Meanwhile, the 3.6 multiplier value will do the opposite: filling the center area with objects. In conclusion, an effective multi-stage object scattering process employs the best multiplier value for the number of objects in each stage and values smaller and greater than the best one. We assume this principle also applies to multi-stage processes involving four or more object types.

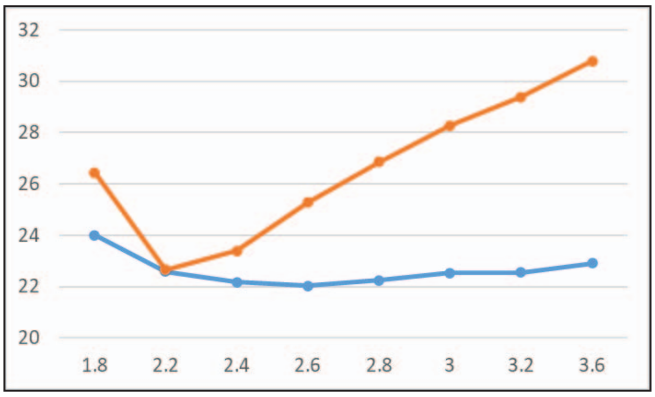

Fig. 9. The relationship between the multiplier value (x-axis) and position slot occupancies' standard deviation (y-axis) in case study 1 when the object number is 20. The blue line belongs to the first irregular level, whereas the orange line belongs to the second level.

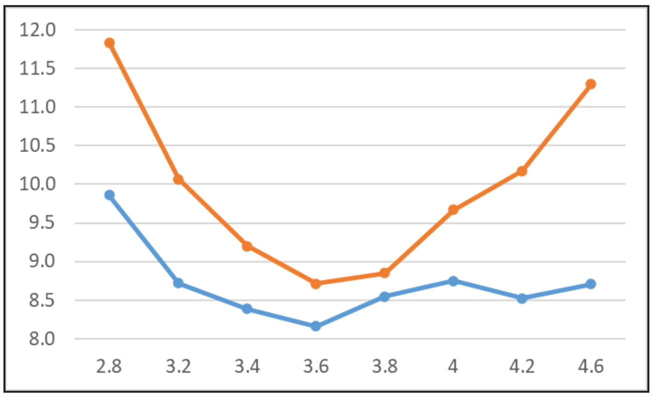

Fig. 10. The relationship between the multiplier value (x-axis) and position slot occupancies' standard deviation (y-axis) in case study 1 when the object number is six. The blue line belongs to the first irregular level, whereas the orange line belongs to the second level.

C. *Limitations*

The limitation of our findings concerns our method's stochastic nature, which caused the standard deviations of position slot occupancies in the first case study to fluctuate. To increase the reliability of our findings, we repeated each application batch 10 times and averaged the results. Unfortunately, statistics-wise, it is far from sufficient. Since the "population" of levels generated by a PCG method is practically infinite, to get results with a confidence level of 95% and a confidence interval of 5%, we would need to repeat each batch more than 300 times instead [24].

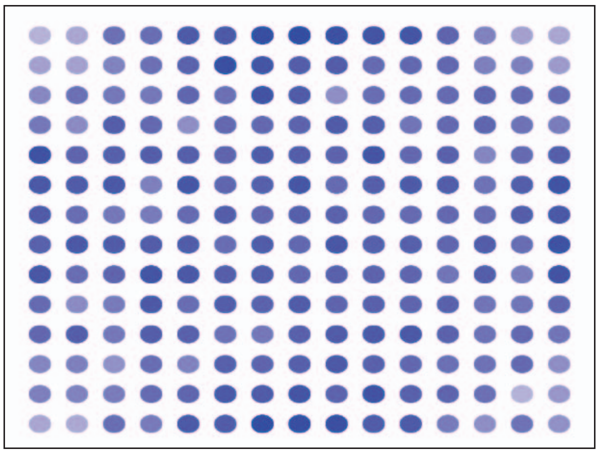

Fig. 11. A heatmap of position slot occupancies resulting from a multi-stage object scattering process employing multiplier values of 2, 2.8, and 3.6 for the first, second, and third stages, each scattering ten objects

V. Conclusions

This work continues our previous [19] in presenting a GA-based PCG method that scatters objects on a game level regardless of the game or its genre. We have shown how the method can satisfy the player's need for exploration by scattering objects widely yet evenly, even in realistic use cases involving irregularly-shaped levels and multiple object types. Compared to simple, rectangular levels, the method's parameters must be slightly altered on irregular ones, and quirks related to object placements on the levels may occur. Meanwhile, our method also scatters multiple object types reliably; moreover, we even acquired better results regarding space coverage by scattering the object types in separate stages and differentiating the stages' parameters.

Overall, our results show that general and genreless PCG methods are entirely possible, even when based on simple algorithms such as the regular GA. Still, we can improve some aspects of our method. In this work, we calculated the distances between object pairs as if there was nothing between them, which is not the case with irregular levels (there may be walls or other obstacles between objects). In our next experiment, we can apply more realistic ways of counting distances between objects, e.g., by encoding the position slots as graph nodes with limited edges between them. We can also investigate whether the principle of an effective multi-stage process is still valid when the stages scatter different numbers of objects.

By generalizing the bijection problem in our method, we can potentially apply our algorithm in many other contexts unrelated to game levels. A possible application in games, outside level generation, is to generate events between the player character and non-player ones (NPCs). In such a context, the game may code the relations between characters as a social graph. Our algorithm can then pair each event with one node in the graph (representing one NPC) and try to maximize the distances between chosen nodes while ensuring that, over multiple game sessions, in the long run, the pairing targets every NPC equally.

REFERENCES

[1] J. R. Whitson, "The New Spirit of Capitalism in the Game Industry," Television & New Media, vol. 20, no. 8, pp. 789–801, Dec. 2019.

[2] W. S. Ravyse, A. S. Blignaut, V. Leendertz, and A. Woolner, "Success factors for serious games to enhance learning: a systematic review," Virtual Reality, vol. 21, no. 1, pp. 31–58, 2017.

[3] E. Massarczyk, P. Winzer, and S. Bender, "Economic Evaluation of Business Models in Video Gaming Industry from Publisher Perspective," in Games and Learning Alliance. GALA 2019. Lecture Notes in Computer Science, vol 11899, pp. 479–489, 2019.

[4] A. Amato, "Procedural content generation in the game industry," in Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation, Springer International Publishing, pp. 15–50, 2017.

[5] J. Togelius, N. Shaker, and M. J. Nelson, "What is procedural content generation?," in Procedural Content Generation in Games, Springer International Publishing, pp. 1–3, 2016.

[6] A. Khalifa, F. de Mesentier Silva, and J. Togelius, "Level Design Patterns in 2D Games," in 2019 IEEE Conference on Games (CoG), pp. 1–8, Aug. 2019.

[7] J. Harris, Exploring Roguelike Games. CRC Press, 2020.

[8] J. Togelius, N. Shaker, and M. J. Nelson, "Games that use PCG," in Procedural Content Generation in Games, Springer International Publishing, pp. 4–5, 2016.

[9] A. Liapis, "10 Years of the PCG workshop: Past and Future Trends," in FDG '20: International Conference on the Foundations of Digital Games, Sep. 2020.

[10] J. C. Spall, "Stochastic Optimization," in Handbook of Computational Statistics, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 173–201, 2012.

[11] A. B. Harisa and W.-K. Tai, "Pacing-based Procedural Dungeon Level Generation: Alternating Level Creation to Meet Designer's Expectations," International Journal of Computing and Digital Systems, vol. 12, no. 1, pp. 401–416, 2022.

[12] A. Khalifa, G. Barros, M. C. Green, and J. Togelius, "Intentional computational level design," in GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference, pp. 796–803. 2019.

[13] A. Alvarez, S. Dahlskog, J. Font, and J. Togelius, "Interactive Constrained MAP-Elites: Analysis and Evaluation of the Expressiveness of the Feature Dimensions," IEEE Transactions on Games, vol. 14, no. 2, pp. 202–211, Jun. 2022.

[14] S. Snodgrass and A. Sarkar, "Multi-Domain Level Generation and Blending with Sketches via Example-Driven BSP and Variational Autoencoders," in FDG '20: International Conference on the Foundations of Digital Games, Sep. 2020.

[15] A. Liapis, G. N. Yannakakis, and J. Togelius, "Towards a generic method of evaluating game levels," in Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE, pp. 1–7, 2013.

[16] B. Horn, S. Dahlskog, N. Shaker, G. Smith, and J. Togelius, "A Comparative Evaluation of Procedural Level Generators in the Mario AI Framework," Foundations of Digital Games 2014, pp. 1–8, 2014.

[17] J. R. H. Mariño, W. M. P. Reis, and L. H. S. Lelis, "An empirical evaluation of evaluation metrics of procedurally generated mario levels," in Proceedings of the 11th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2015, pp. 44–50, 2015.

[18] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Evolving missions to create game spaces," in IEEE Conference on Computatonal Intelligence and Games, CIG, pp. 1–8, 2016.

[19] P. W. Atmaja and Sugiarto, "Generating Object Placements for Optimum Exploration and Unpredictability in Medium-Coupling Educational Games," in 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), pp. 1–6, Dec. 2020.

[20] J. Togelius and N. Shaker, "The search-based approach," in Procedural Content Generation in Games, Springer International Publishing, pp. 17–30, 2016.

[21] A. Liapis, "Multi-segment evolution of dungeon game levels," in GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference, pp. 203–210. 2017.

[22] G. Smith, M. Cha, and J. Whitehead, "A framework for analysis of 2D platformer levels," in Proceedings of Sandbox 2008: An ACM SIGGRAPH Videogame Symposium, Sandbox'08, pp. 75–80, 2008.

[23] N. Shaker et al., "The 2010 Mario AI Championship: Level Generation Track," IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 4, pp. 332–347, Dec. 2011.

[24] A. Hazra, "Using the confidence interval confidently," Journal of Thoracic Disease, vol. 9, no. 10, pp. 4124–4129, Oct. 2017.