2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)

Generating Object Placements for Optimum Exploration and Unpredictability in Medium-Coupling Educational Games

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

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

***Abstract*—Educational games are powerful tools in the era of Education 4.0. However, their applications are hampered by many issues, including high development costs. Procedural content generation (PCG) and medium coupling are two potent methods for reducing an educational game's development cost; however, their joint application is under-studied. In this study, we explored the topic through a PCG application for generating placements of objects representing elements of learning content in a game map. The content consisted of correct answers, which the player must gather in the correct order, and wrong answers, which the player must avoid. We employed an evolutionary algorithm in three stages of the generation: it first generated a minimal set of correct answer element objects (CAEOs), followed by generating a copy of the set, and then a set of wrong answer element objects (WAEOs). We employed and tested a fitness function that calculated the mean and standard deviation of object pair distances. The results show that the generation algorithm is capable of generating CAEOs and WAEOs distributions that are unpredictable and encourage exploration. We find that multiplying the fitness function's standard deviation variable by a specific value, which depends on how many objects are to be generated, is crucial to the algorithm's success. We also discuss the limitations of this study and directions for future ones.**

Keywords—procedural content generation, educational games, medium coupling, object placements, evolutionary algorithm.

I. Introduction

Education in the era of Industrial Revolution 4.0 is a wilderness full of risks and excitements, and educational games are among the tools capable of taming it [1]. On the one hand, the games offer numerous significant benefits over the more traditional teaching and learning methods, such as inducing the Flow experience [2] and facilitating learning through analogies [3]. On the other hand, their adoption by educational institutions is not without difficulties. A particular problem hindering the applications of the games is development cost [4]. The *medium coupling* approach is one that structures an educational game to preserve its educational strength and, at the same time, reduce its cost. On the other hand, *procedural content generation* (PCG) [5] can greatly save the game developer's resources by automating their content production process. Taken together, the medium coupling and PCG can potentially synergize for even greater cost reduction effects; however, such synergy is still under-explored.

This study, therefore, aims to explore how to apply PCG and medium coupling together in educational games. Specifically, we explore the application of PCG to generate placements of objects representing the learning content of a medium-coupling educational game.

A. Literature Review

Educational games are a subset of serious games, which are games with two purposes: to entertain their players and to achieve more "serious" goals [6]. In educational games, the "serious goals" are, of course, related to teaching and learning. From the definition, it follows logically that an educational game consists of not only *game content* but also *learning content* [7][8]. The two contents are not separate entities but must be integrated and function as one. The literature of educational games recognizes two opposing manners of integration: the *intrinsic* [9][10], also known as "tight coupling," and the *extrinsic* [11], also known as "loose coupling." An example of an intrinsically-integrated game is an action game where the player's attacks and the enemies' weaknesses obey specific mathematical rules. On the other hand, a game that occasionally presents quizzes unrelated to, and even interrupts, its gameplay is a perfect example of an extrinsically-integrated one. The consensus on the integration is as follows: the more intrinsic the integration, the stronger the educational effects of the game, and (unfortunately) the higher its development cost and complexity.

Other than the extrinsic and intrinsic ones, there exists a third content integration approach. Although it has no agreed-upon name in the literature, we term it "medium coupling" in this study. If the extrinsic and intrinsic integrations represent two extremes, then the medium coupling sits somewhere between those two in the "spectrum." The medium coupling has the advantage of presenting educational content as a part of the gameplay while keeping the development cost and complexity manageable. It has been utilized in the works of Rosyid et al. [12], Garneli et al. [13], Beserra et al. [14], and Atmaja et al. [15], and also in commercial products such as *Toon Math* (https://play.google.com/store/apps/details?id=com.closeapps.mathrun).

Rather than integrating the game's core mechanics with the learning content, a medium-coupling game assigns its learning content's elements to specific in-game objects. Rosyid et al. [12] experimented with an educational dungeon crawling game about chemical molecules, where the chemical elements of the molecules were scattered in the game world as pickable items. Each element of the learning content (a chemical element) was assigned to one game content's element (a pickable item). The pickable items then effectively served as an "interface" through which the player would interact with the learning content's elements. Picking the wrong items, or the right items in the wrong order, would result in a wrong molecule.

978-1-7281-9673-2/20/$31.00 ©2020 IEEE

Under medium coupling, elements of different learning contents can be assigned to game objects without drastically changing the game's inner working. The easy replacement of learning content is the main factor of medium coupling's cost reduction strength. The game developer only needs to ensure the correct arrangement of the game objects. The arrangement determines the game's difficulty level and fun factor, and a wrong arrangement may even make the game impossible to finish. To save resources, the game developer can theoretically apply PCG to automate the arrangement process.

PCG itself has been harnessed in its various forms in educational games. Hooshyar et al. [16] experimented with a data-driven PCG to generate content for an English-learning game. Jemmali et al. [17] employed a grammar-based approach to generate a puzzle programming game's levels. Park et al. utilized Answer-Set Programming [18] and Generative Adversarial Networks [19] to generate levels for another programming game, this time of the 3D adventure genre. However, the use of PCG for medium-coupling educational games has yet to be explored.

Meanwhile, game researchers have extensively studied game maps (also known as "levels"), as good game maps are essential to the quality of a game [20]. The procedural generation of game maps is a vibrant subset of PCG [21], and many quality metrics have been proposed to guide the generation toward more playable results [22][21][23]. The metrics include area control, exploration, and leniency. Of particular interest in this study is exploration, which is defined as how large the area a player must visit to achieve their objectives on the map [22][24]. Generally, the larger the area, the more fun the game is for the player.

Algorithms for object placements covering areas as large as possible are not novel. *Uniform scattering* is a type of such algorithm [25], which has found use in robotics [26]. Researchers such as Koay [27] have explored methods for uniform point distribution on geometrical shapes. However, PCG also needs to surprise the player, as unpredictability is a desirable factor in games [28]. Therefore, the utilized algorithm must not pursue maximum space coverage by sacrificing unpredictability.

PCG in games commonly achieves such unpredictability through stochastic algorithms. On the other hand, the exploration metric's current usage is still limited to PCG for discrete game elements such as mission graphs [24] and tiles in strategy games [29]. To the best of our knowledge, a general-purpose object placement algorithm supporting both exploration and unpredictability does not yet exist in the literature.

B. Contributions

This study presents two contributions, the first being a novel, general-purpose evolutionary algorithm for object placements optimizing both unpredictability and space coverage. The second contribution is a three-stage PCG method for generating placements of objects representing learning content in medium-coupling educational games, which uses the evolutionary algorithm in each of the stages.

II. Research Methodology

In this study, we employed a three-stage PCG to generate placements of objects in a game map. Each object represented either a correct or a wrong answer element. Regarding player exploration and unpredictability, we established two criteria for our algorithm, the first being the player should, on average, traverse a sufficiently long distance to collect answer element objects. The second criterion was that the object placements should not favor only some areas on the map.

According to the PCG taxonomy in [30], our algorithm was *generic*, *stochastic*, *automatic*, and followed the *generate-and-test* approach. Our algorithm also generated *necessary* content for the game, although the generation could be either *online* or *offline*. We experimented on the algorithm to understand the configuration in which it would work best.

A. Correct and Wrong Answer Element Objects

In this study, a correct answer consisted of an ordered string of 10 different elements. A minimal set of correct answer element objects (hereafter abbreviated CAEOs) would also consist of 10 members, with each representing one of the answer elements. The wrong answer element objects (hereafter abbreviated WAEOs), on the other hand, could represent anything else with regards to the learning content. A game map would contain more than one copy of each CAEO to allow the player to collect CAEOs more easily.

B. Game Map

We used rectangular game maps, with each containing uniformly-distributed position markers. The algorithm would create CAEOs and WAEOs and place each of them on one of the markers. The maps were of varying dimensions determined by the number of position markers and the space between each marker pair. Fig.1 shows a game map example. We experimented with different spaces and numbers of markers to understand their effects on the PCG.

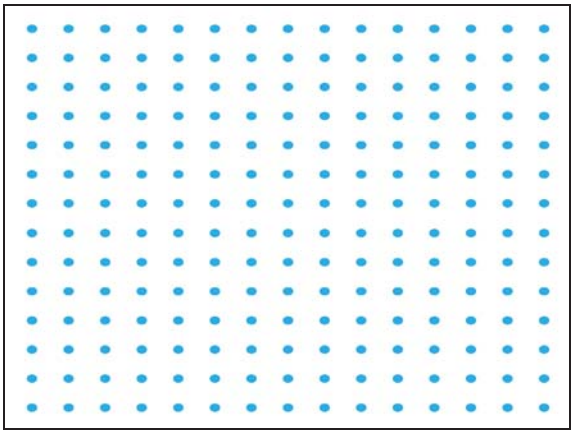

Fig. 1. An example of a game map with 15x14 position markers (blue dots) with a 200 pixels space between each other.

C. Generation Algorithm

To generate a set of answer objects with a desired distribution across the game map, we implemented a three-stage generation process. The first stage generated a minimal set of CAEOs (hereafter termed initial CAEOs); the second stage added a copy of the minimal CAEO set (hereafter termed additional CAEOs); the last stage spread WAEOs in the remaining space in the map. Generating the initial

CAEOs first and the additional CAEOs second was to ensure the correct distribution of the CAEOs and prevent them from clustering in a small area.

In the first stage, the initial CAEOs would appear at ten random position markers in the game map. In the second stage, the additional CAEOs would appear at 10 of 170 available markers not already occupied by the initial CAEOs. Likewise, the generation of 30 WAEOs in the last stage would utilize the remaining 160 markers.

Each stage in the generation process employed the same core algorithm, which was based on the evolutionary search algorithm described in [31] with some adjustments. First, we did not shuffle the population. Second, a mutated individual did not produce any offspring but instead replaced its old self in the population. The evolutionary algorithm in each stage ran for 100 iterations, and its population consisted of 40 individuals. Each individual was a game map containing several relevant objects (e.g., 10 CAEOs in the first stage of a generation process).

The population was evenly split in the middle into a high- and low-fitness part. The algorithm mutated ten random individuals from the low-fitness part and four others from the high-fitness one in each iteration. The mutation process would select several CAEOs or WAEOs in an individual and reposition them to random, unoccupied position markers in the game map. Every individual had a chance to mutate in each iteration, with one exception: the algorithm never selected the individual with the highest fitness value to prevent the value from decreasing.

D. Fitness Functions

Equation (1) shows our fitness function, which calculated the distances between pairs of generated CAEOs or WAEOs. A long average distance (a) would require the player to traverse a long distance to collect CAEOs, thus facilitating exploration. However, merely maximizing the average distance would result in predictable CAEO distributions, and that is why we also employed the object pair distances' standard deviation (s) to balance it. How strong the effect of the standard deviation was determined by a multiplier (m). We experimented with different multiplier values for each game map to find the best one. Meanwhile, the longest distance between a pair of WAEOs or CAEOs (r) was used to normalize different maps' fitness values.

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

E. Experiment Specification

We experimented with the generation algorithm to understand three things: (1) the object distribution tendencies of the generation algorithm, (2) the ideal specification of the three-stage PCG for optimum unpredictability and player exploration, and (3) how far the distance the player must traverse, under an ideal specification of the PCG, to collect CAEOs. To achieve the first goal, we conducted generation process batches, with each batch consisting of 400 generation processes. As the goal did not require complete sets of CAEOs and WAEOs, the generation processes consisted of only the first stage. In the batches, we adjusted four variables: the multiplier in the fitness function, the number of position markers, the space between marker pairs, and the number of generated objects. The results of the generation process batches were then visualized as scatter plots with varying alpha values of the dots: more transparent dots indicated seldom-occupied position markers and vice versa. To measure the quality of object distribution in each batch, we analyzed position markers' occupancies by calculating their standard deviation. A desirable distribution would be one with a low value of the standard deviation, as the CAEOs would evenly occupy the position markers within the 400 generation processes, therefore assuring the unpredictability aspect.

The second goal was achieved by generating complete sets of CAEOs and WAEOs under varying multiplier values of the fitness function. Meanwhile, the third goal was, once again, achieved through a generation process batch. We calculated the distances for collecting complete CAEO strings to get the mean and standard deviation of the distances in each generation process and the entire process batch. As we needed the initial and the additional CAEOs for the calculation, the process batch consisted of the first and second generation stages. With two sets of CAEOs, with each set consisting of 10 CAEOs, there would be 1024 possible paths for collecting a CAEO string in each generation process and a total of 409,600 paths in the whole process batch.

III. Results and Discussions

A. Analysis of Generated Object Distributions

Our experiment to understand the algorithm's object distribution tendencies uncovered valuable insight. There is one multiplier value producing the best object distribution for each specific number of generated objects. The multiplier does not seem to depend on other variables; i.e., it is the same across maps with different position marker numbers and spaces between markers.

Fig. 2 shows a chart with nine dots for nine generated object numbers (x values) and their best multipliers (y values). We used an online curve fitting service (https://mycurvefit.com/) to estimate the function that maps object numbers to the best multipliers. Equation (2) shows the function, which acquires a 0.99 coefficient of determination. It seems that the best multipliers stabilize around 2.4 for large numbers of generated objects.

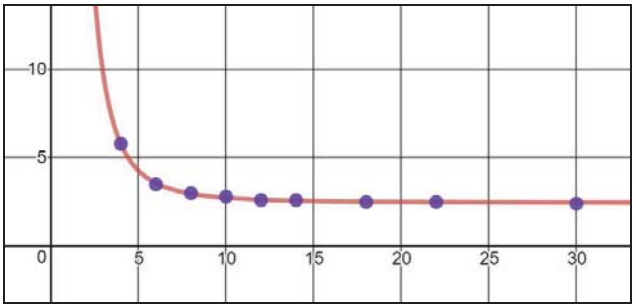

Fig. 2. Object numbers (x values), the best multipliers (y values), and the estimated function between them (the curve).

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

Fig. 3 shows the standard deviations of position marker occupancies in 11 generation process batches with 15x14 position markers, 200 pixels spaces, ten generated objects in each process, and 1.8 to 3.8 multipliers. We can see that the 2.8 multiplier produces the best object distribution, i.e., one with the lowest position marker occupancies' standard deviation. Changing the number of markers or the spaces between them will increase or decrease the standard deviations; however, the lowest one will still occur at the multiplier of 2.8.

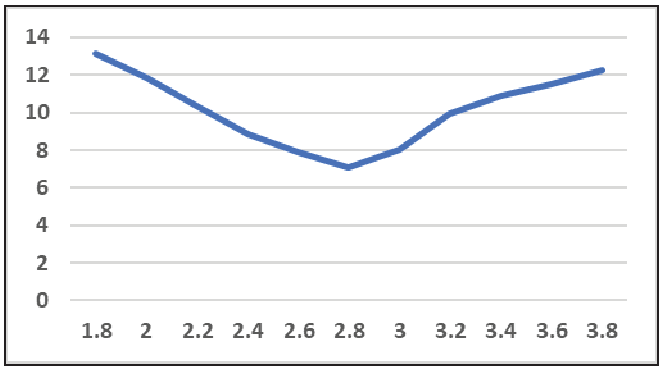

Fig. 3. The standard deviations of position marker occupancies in generation process batches with 15x14 position markers, 200 pixels spaces, ten generated objects in each process, and 1.8 to 3.8 multipliers.

Fig. 4 to 6 show the results of generation process batches, where each process generated ten objects on a map with 15x14 position markers and 200 pixels spaces between them. Fig. 4 shows an object distribution under a multiplier of two. We can see that the generation processes favor scattering objects along the perimeter, resulting in CAEOs that are predictable to a degree. As Fig. 5 shows, a quick and straightforward solution to the predictability is increasing the standard deviation multiplier. With a 2.8 multiplier, the distribution no longer skews toward the perimeter and can already satisfy both player exploration and unpredictability. However, Fig. 6 shows that increasing the multiplier further would be counterproductive, as the generation processes would then put the objects in the inner area too often.

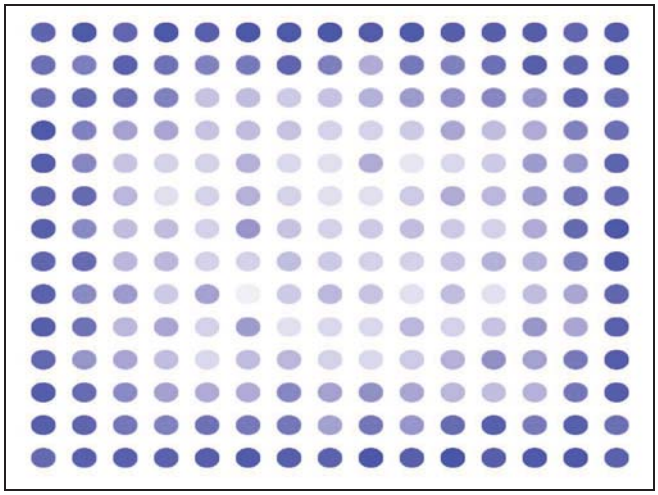

Fig. 4. The distribution of objects generated in 400 generation processes, with each process generated ten objects under a multiplier of two. The map contains 15x14 position markers with 200 pixels spaces.

Meanwhile, Fig. 7 and 8 corroborate our finding of the best multipliers dependent only on generated object numbers. The game maps in Fig. 5, 7, and 8 have different position marker dimensions and spaces between markers; regardless of that, the 2.8 multiplier generates objects with similar distributions on the three maps.

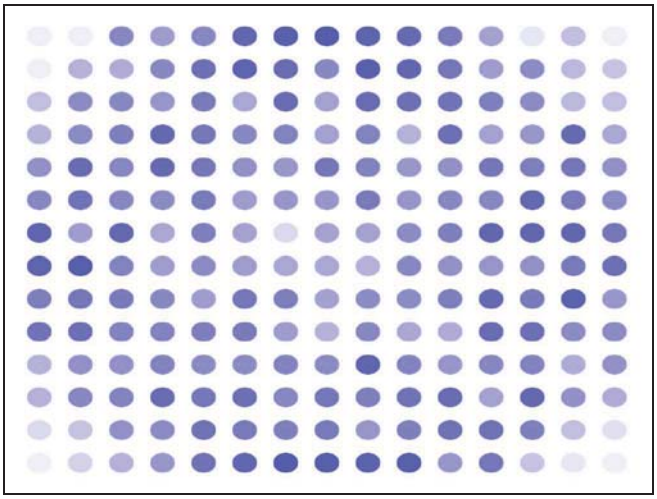

Fig. 5. The distribution of objects generated in 400 generation processes, with each process generated ten objects under a 2.8 multiplier. The map contains 15x14 position markers with 200 pixels spaces.

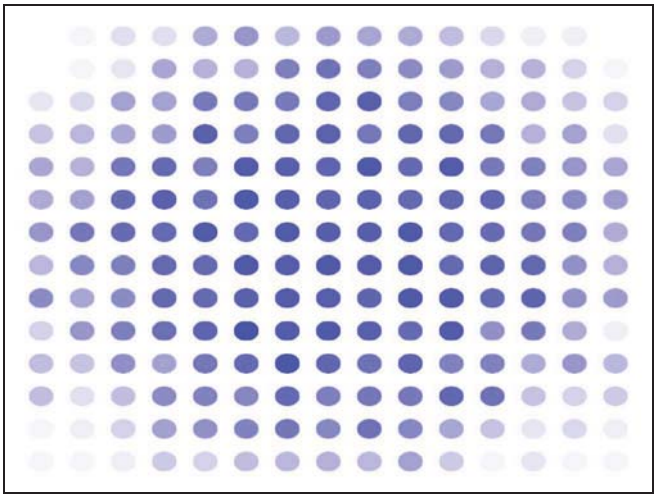

Fig. 6. The distribution of objects generated in 400 generation processes, with each process generated ten objects under a 3.6 multiplier. The map contains 15x14 position markers with 200 pixels spaces.

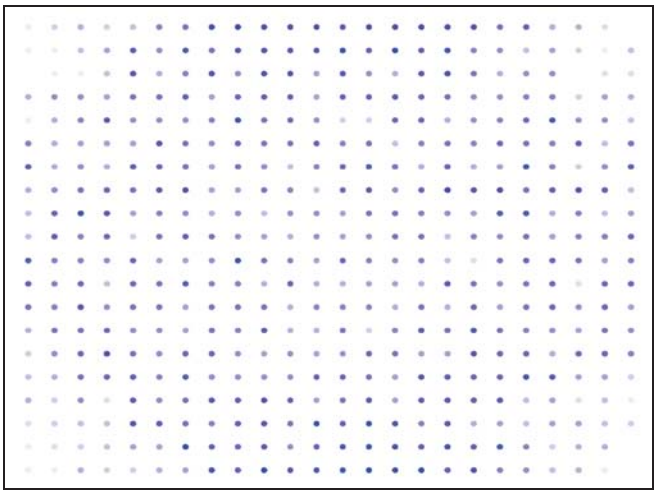

Fig. 7. The distribution of objects generated in 400 generation processes, with each process generated ten objects under a 2.8 multiplier. The map contains 24x20 position markers with 500 pixels spaces.

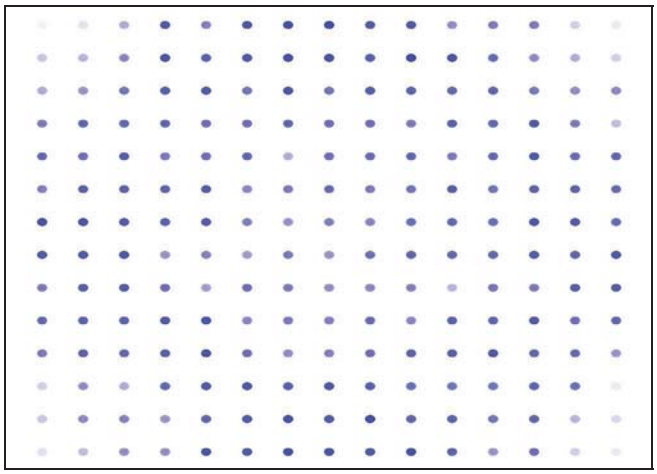

Fig. 8. The distribution of objects generated in 400 generation processes, with each process generated ten objects under a 2.8 multiplier. The map contains 15x14 position markers with 500 pixels spaces.

B. *Generating Complete Sets of Answer Element Objects*

Fig. 9 shows an example of a complete set of CAEOs and WAEOs generated on a 15x14 position markers and 200 pixels spaces map and under different multipliers. For the first and second stages, we employed a 2.8 multiplier, whereas a 2.4 one was used in the third stage for the WAEOs. As the 2.8 multiplier pays slightly less attention to the map's perimeter, the 2.4 one can complement it nicely.

Ultimately, however, which multiplier to use in which stage may depend on the game developer's intention. Even if the 2.8 multiplier works best for the game map, it does not mean the algorithm must always employ it. Using lower or higher multipliers occasionally can be an excellent way to surprise and excite the player.

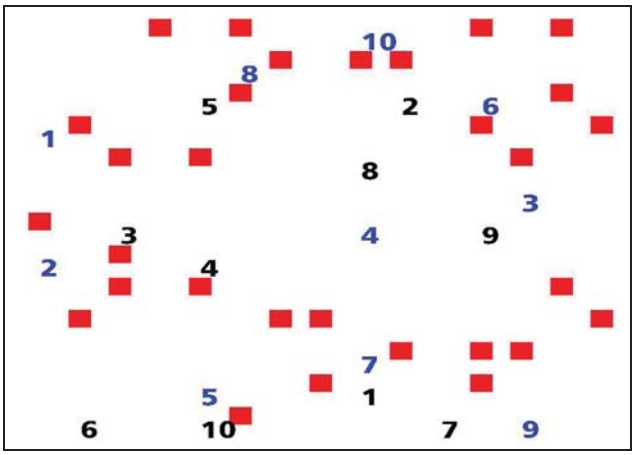

Fig. 9. An example of a complete set of procedurally-generated answer element objects. The initial and additional CAEOs are the blue and black letters, respectively, and the WAEOs are the red boxes. The map is of the 15x14 position markers and 200 pixels spaces specification. The CAEOs were generated under a 2.8 multiplier, whereas the WAEOs were generated under a 2.4 one.

C. *Traversal Distances for Collecting CAEOs*

Fig. 10 shows the average distances, among 400 generation processes, that a player must traverse to collect ten CAEOs in their correct order. As before, each generation also used a 2.8 multiplier and was done on the 15x14 markers and 200 pixels spaces map. The aggregated mean of the distances within the 400 generations is 13,314 pixels, which slightly exceeds the game map's perimeter. The average distance indicates the generation algorithm's capability of presenting exploration opportunities to the player. Meanwhile, the standard deviation of the aggregated distance data is 2,145 pixels. The value amounts to only 16% of the average and indicates the reliability of the average.

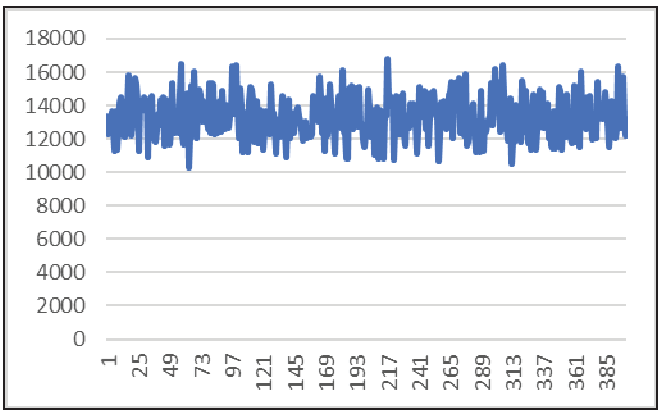

Fig. 10. Average traversal distances, in pixels and among 400 generations, for collecting ten CAEOs in the correct order. The generation was done under a 2.8 multiplier on a map with 15x14 markers and 200 pixels spaces.

D. *Limitations*

This study possesses three limitations. First, we employed simple rectangular game maps with no obstacles, forbidden areas, or other elements that may influence the gameplay. They contrast the majority of game maps in commercial games, which are more complex and varied. Second, we do not make any assumptions about the characteristics of the generated objects; whether they are capable of any movements or actions is outside this study's scope. Allowing the objects to move freely (e.g., making them enemies of the player) may make their initial placements in the map less relevant, thus limiting the applicability of the generation algorithm. Finally, we refrain from making any claim on the algorithm's implementation readiness, as we did not experiment on its practical aspects, e.g., execution speed.

IV. Conclusions

Educational games, while being potent educational tools, suffer from development cost problems [4]. In this study, we have experimented with a cost-reduction method for educational games, which is in the form of procedural content generation (PCG) for medium-coupling educational games, which are those that assign their learning content's elements to their game objects. We have proposed and tested a three-stage PCG for placements of objects representing correct and wrong answers to certain learning content. Our experiment employed game maps containing position markers spread evenly across the maps. The PCG method generated and placed correct and wrong answer element objects (CAEOs and WAEOs, respectively) stochastically on the map. The generation process would try to position the objects unpredictably while also requiring player exploration; for that reason, the process consisted of three stages for generating an initial set of CAEOs, an additional set of CAEOs, and a set of WAEOs.

The generation process' evolutionary algorithm employed a fitness function calculating the average distance between position markers as a positive variable and the distances' standard deviation as a negative variable. We have proven that the generation algorithm, with a specific

multiplier for the fitness function's standard deviation, can generate and distribute objects evenly across the game map (i.e., facilitating player exploration) while ensuring the distribution's unpredictability. We have found that the correct multiplier value depends on the number of objects generated on the map but not on the map itself. Through curve-fitting, we have estimated the function that maps generated object numbers to the correct multipliers. We have also found how mixing different multipliers for the three stages is beneficial in generating complete sets of CAEOs and WAEOs with good distributions. Furthermore, we have proven that the generation process can ensure average traversal distances for the player to collect CAEOs that are both reliable and sufficiently long.

This study, ultimately, is only preliminary. It only delves into a particular subset of the broader topic of PCG for medium-coupling educational games. Regardless of that, we hope that this study can start discussions on medium-coupling games and PCG for object placements. In the future, we will explore other related and more complex generation processes. Graph-based procedural generation of objects [24] will be beneficial for game maps with varied topologies. Concerning the learning content, we will explore how to generate objects representing more complex answer elements, e.g., structured as trees or directed graphs. It will also be fruitful to explore further our evolutionary algorithm, e.g., its usage in more general contexts in games and beyond. Refining the fitness function to achieve better object distributions is also a worthwhile topic.

REFERENCES

[1] F. Almeida and J. Simoes, "The role of serious games, gamification and industry 4.0 tools in the education 4.0 paradigm," *Contemporary Educational Technology*, vol. 10, no. 2, pp. 120–136, 2019.

[2] C. C. Chang, C. Liang, P. N. Chou, and G. Y. Lin, "Is game-based learning better in flow experience and various types of cognitive load than non-game-based learning? Perspective from multimedia and media richness," *Computers in Human Behavior*, vol. 71, pp. 218–227, 2017.

[3] W. Martin, M. Silander, and S. Rutter, "Digital games as sources for science analogies: Learning about energy through play," *Computers and Education*, vol. 130, pp. 1–12, 2019.

[4] E. A. Boyle *et al.*, "An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games," *Computers and Education*, vol. 94, pp. 178–192, 2016.

[5] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 9, no. 1, 2013.

[6] F. Laamarti, M. Eid, and A. El Saddik, "An Overview of Serious Games," *International Journal of Computer Games Technology*, 2014.

[7] S. Arnab *et al.*, "Mapping learning and game mechanics for serious games analysis," *British Journal of Educational Technology*, vol. 46, no. 2, pp. 391–411, 2015.

[8] M. B. Carvalho *et al.*, "An activity theory-based model for serious games analysis and conceptual design," *Computers and Education*, vol. 87, pp. 166–181, 2015.

[9] A. Echeverría, E. Barrios, M. Nussbaum, M. Améstica, and S. Leclerc, "The atomic intrinsic integration approach: A structured methodology for the design of games for the conceptual understanding of physics," *Computers and Education*, vol. 59, no. 2, pp. 806–816, 2012.

[10] J. Habgood and S. E. Ainsworth, "Motivating children to learn effectively: Exploring the value of intrinsic integration in educational games," *Journal of the Learning Sciences*, vol. 20, no. 2, pp. 169–206, 2011.

[11] O. Ku, S. Y. Chen, D. H. Wu, A. C. C. Lao, and T. W. Chan, "The effects of game-based learning on mathematical confidence and performance: High ability vs. low ability," *Educational Technology and Society*, vol. 17, no. 3, pp. 65–78, 2014.

[12] H. A. Rosyid, M. Palmerlee, and K. Chen, "Deploying learning materials to game content for serious education game development: A case study," *Entertainment Computing*, vol. 26, pp. 1–9, 2018.

[13] V. Garneli, C. Sotides, K. Patiniotis, I. Deliyannis, and K. Chorianopoulos, "Designing a 2D Platform Game with Mathematics Curriculum," in *Games and Learning Alliance. GALA 2019. Lecture Notes in Computer Science, vol 11899*, Springer, Cham, 2019, pp. 42–51.

[14] V. Beserra, M. Nussbaum, R. Zeni, W. Rodriguez, and G. Wurman, "Practising arithmetic using educational video games with an interpersonal computer," *Educational Technology and Society*, vol. 17, no. 3, pp. 343–358, 2014.

[15] P. W. Atmaja, F. Muttaqin, and S. Sugiarto, "Facilitating educational contents of different subjects with context-agnostic educational game: A pilot case study," *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 6, no. 1, pp. 53–65, 2020.

[16] D. Hooshyar, M. Yousefi, M. Wang, and H. Lim, "A data-driven procedural-content-generation approach for educational games," *Journal of Computer Assisted Learning*, vol. 34, no. 6, pp. 731–739, Dec. 2018.

[17] C. Jemmali, C. Ithier, S. Cooper, and M. S. El-Nasr, "Grammar Based Modular Level Generator for a Programming Puzzle Game," in *Experimental AI in Games: An AIIDE 2020 Workshop*, 2020.

[18] K. Park, B. Mott, W. Min, E. Wiebe, K. E. Boyer, and J. Lester, "Generating Game Levels to Develop Computer Science Competencies in Game-Based Learning Environments," in *Artificial Intelligence in Education. AIED 2020. Lecture Notes in Computer Science, vol 12164*, 2020, pp. 240–245.

[19] K. Park, B. W. Mott, W. Min, K. E. Boyer, E. N. Wiebe, and J. C. Lester, "Generating Educational Game Levels with Multistep Deep Convolutional Generative Adversarial Networks," in *2019 IEEE Conference on Games (CoG)*, 2019.

[20] A. Khalifa, F. de Mesentier Silva, and J. Togelius, "Level Design Patterns in 2D Games," in *2019 IEEE Conference on Games (CoG)*, 2019.

[21] 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.

[22] 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*, 2013.

[23] 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*, 2015, pp. 44–50.

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

[25] L. Barriere, P. Flocchini, E. Mesa-Barrameda, and N. Santoro, "Uniform scattering of autonomous mobile robots in a grid," *International Journal of Foundations of Computer Science*, vol. 22, no. 03, pp. 679–697, Apr. 2011.

[26] P. Poudel and G. Sharma, "Time-optimal uniform scattering in a grid," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*, 2019, pp. 228–237.

[27] C. G. Koay, "A simple scheme for generating nearly uniform distribution of antipodally symmetric points on the unit sphere," *Journal of Computational Science*, vol. 2, no. 4, pp. 377–381, Dec. 2011.

[28] N. Shaker and M. Abou-Zleikha, "Alone we can do so little, together we can do so much: a combinatorial approach for generating game content," in *AIIDE'14: Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2014, pp. 167–173.

[29] A. Liapis, G. N. Yannakakis, and J. Togelius, "Designer modeling for Sentient Sketchbook," in *2014 IEEE Conference on Computational Intelligence and Games*, 2014.

[30] J. Togelius, N. Shaker, and M. J. Nelson, "A taxonomy of PCG," in *Procedural Content Generation in Games*, Springer International Publishing, 2016, pp. 7–10.

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