Procedural Maze Generation with Considering Difficulty from Human Players' Perspectives

Keita Fujihira, Chu-Hsuan Hsueh, and Kokolo Ikeda

School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan
{keita.fujihira, hsuehch, kokolo}@jaist.ac.jp

Abstract. In video game development, creating maps, enemies, and many other elements of game levels is one of the important process. In order to improve players' game experiences, game designers need to understand players' behavioral tendencies and create levels accordingly. Among various components of levels, this paper targets mazes and presents an automatic maze generation method with considering difficulty based on human players' tendencies. We first investigate the tendencies using supervised learning and create a test player considering human-likeness by using the tendencies. The test player simulates human players' behaviors when playing mazes and judges difficulty according to the simulation results. Maze evaluation results of subject experiments show that our method succeeds in generating mazes where the difficulty estimated by the test player matches human players'.

Keywords: Procedural content generation · RPG · Maze · Difficulty.

1 Introduction

Artificial Intelligence (AI) has succeeded in various fields; and for games, AI has been applied not only to make computer players but also to generate game content, well known as Procedural Content Generation (PCG). PCG has attracted attention from academia and industry, and was mainly researched for popular game genres such as platformer games (e.g., Super Mario Bros) [1] and shooting games [2]. In contrast, research for role-playing games (RPG) is relatively few.

RPG is a classical genre that includes famous titles such as The Elder Scrolls[1] and Fallout[2] series. In addition to defeating final bosses, players may have various purposes, and these factors are often reflected in game design. One example is "finding and collecting items on the maps." Since it is not exciting that rare items are easily found, game designers may design game maps that navigate players away from rare items. Another example is "going to the next town." Although exploring game maps may be enjoyable, getting lost for a long time is frustrating. Meanwhile, too explicit navigation (e.g., maps containing a single

[1] https://elderscrolls.bethesda.net/en
[2] https://fallout.bethesda.net/en/

path or flashing marks showing the correct directions) harms the play experience. Therefore, it is better that RPG maps leave freedom for players while implicitly control their behaviors. These are key challenges in game design, known as "Narrative Paradox" [3]. Game designers usually create maps with such complicated mechanisms manually. To develop a large-scaled game, creating maps need many game designers and time, making development costs high.

The main goal of this research is the automatic generation of such maps. As the first step, we target on simple two-dimensional mazes that contain only passages and walls but no enemies, items, or NPCs. We investigate human players' behavioral tendencies when playing such mazes. In this paper, we aim to know the difficulty of mazes from human players' perspectives and generate mazes accordingly. We analyze human players' path selection tendencies by supervised learning and use the learned model to create a test player for maze generation. In evaluation experiments, the results that human players play the generated mazes are consistent with the results predicted by the test player.

The rest of the paper is organized as follows. Section 2 introduces work related to PCG and maze generation. Section 3 and Section 4 present investigation of human players' tendencies using supervised learning and maze generation approach using the test player, respectively. Section 5 shows the results of the subject experiments on maze evaluation. Finally, Section 6 makes conclusions.

2 Related Work

PCG is the algorithmic creation of game content with limited or indirect user input [4]. A large variety of content such as maps [5], puzzles [6], and NPCs [7] is covered as targets of PCG. Among approaches for PCG, the following four are representative. Constructive PCG generates content via rules that are usually hand-crafted. Search-based PCG optimizes generated content through repeats of generation and evaluation [8]. PCG via Machine Learning trains generation models using existing game content to generate new one [9]. PCG via Reinforcement Learning is a very recent approach that trains generation models by reinforcement learning and does not require existing game content [10, 11].

This paper focuses on the generation of maze maps. Simple Constructive PCG algorithms include digging method [12], extending method [13], and toppling method [14]. Some researchers further considered difficulty and used Search-based PCG for maze generation [15–17]. For example, Kwiecień [17] proposed a method to generate challenging mazes based on the cockroach swarm optimization algorithm, and defined maze complexity by path complexity and the number of branch points. The path complexity depended on the length of a solution path and the number of its direction changes. However, none of them explicitly considered human players' behaviors. Even though a maze looks complicated with many branch points, players do not always get lost. Our approach differs from these papers' in that ours considers difficulty from human players' perspectives.

3 Investigation of Human players' Tendencies

Many mazes have been released and played around the world. Mazes can be classified into various types according to the number of dimensions, uniqueness of solution path, etc. In this paper, we target two-dimensional mazes with unique solution paths. It is important to grasp human players' behavior tendencies to adjust difficulty from humans' perspectives. Section 3.1 presents subject experiments on collecting human players' behaviors. Section 3.2 employs supervised learning on path selection probability to investigate the tendencies.

3.1 Subject Experiments

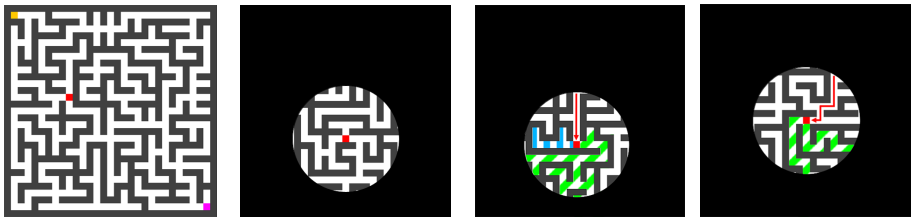
We generated the mazes using classical algorithms [12–14] for experiments, and the settings are listed as follows:

- The maze size is 31×31 (Small), 41×41 (Medium), or 51×51 (Large).
- The maze consists of only passable cells (passage) and impassable cells (wall).
- Since the algorithms place two consecutive passages at a time, each passage must locate at a cell whose x-, y-, or both coordinates are even numbers, assuming that the top-left corner is (1, 1).
- A player's goal is to get from a starting point to an end point.
- In cases that players do not visit a cell more than once, the solution path that connects the starting point to the end point is unique.
- The starting point and the end point are located at the upper left and lower right in the maze, respectively.
- The maze does not contain loops, cycles, and isolated cells.
- A player can move only one cell vertically or horizontally per action.
- The time limit is set for playing a maze, depending on the maze size: 80 seconds for Small, 100 seconds for Medium, 150 seconds for Large. If the player does not reach the end point within the time limit, the game is over.
- A player has either one of the following two ranges of view.
  - Wide view: A player can view the entire maze, as shown Fig. 1(a).
  - Narrow view: A player can only view inside a circle centered on himself/herself, as shown Fig. 1(b). The diameter of the circle is half of the side length of the maze (e.g., $31/2 = 15.5$ cells for Small).

A total of twenty players (males and females in twenties to forties) participated in the experiments. The participants included both players who were interested in playing games and players who were not. The participants played 21 mazes with different maze sizes and view ranges.

3.2 Prediction by Supervised Learning

Human players' tendencies are investigated by using supervised learning on probabilities of selecting proceeding directions at *branch point*. Note that not all intersections are branch points, which we will define soon later. For a given cell in



(a) (b) (c) (d)

Fig. 1. Small size mazes with (a) the wide view and (b) the narrow view, and intersections that (c) is not and (d) is a branch point, where the yellow, pink, and red cells are the starting point, the end point, and the player, respectively.

a maze, we define a proceeding direction to be *uncertain* if any part of the succeeding paths is invisible within the range of the narrow view. In Fig. 1(c), going right leads to succeeding paths in green oblique line. It can be seen that some paths go to areas outside of the narrow view. Thus, the proceeding direction of right is uncertain. In contrast, going left leads to succeeding paths fully visible in the narrow view (blue vertical lines) and is not uncertain. We then define a *branch point* as an intersection connected to two or more uncertain proceeding directions. In Fig. 1(d), the player (red cell) is at a branch point, which is connected to two uncertain proceeding directions. Note that branch points in both wide and narrow view mazes are defined in the same way.

Learning Settings We collected branch points and the players' selection proportions of proceeding directions from the mazes in the subject experiments. For example, assume that players can go right or go down at some branch point. If 16 out of 20 players went right, the selection proportion of going right at this branch point was $16/20 = 0.8$. Note that we only counted data from the first time that players reached the branch points. In the supervised learning, the input was maze features related to branch points, including a proceeding direction, and the output was the predicted selection proportion. We extracted 23 input features, as explained in Appendix A. From the subject experiments, We collected 147 and 197 branch points for wide and narrow view mazes, respectively. Since the amount of data was small, we doubled the data by creating copies that swapped the x-axis and y-axis values of the original maze data. The prediction models were built based on the LightGBM[3] with leave-one-out cross-validation.

Learning Results Fig. 2 shows the prediction results. Note that selection proportions of the proceeding directions at a branch point were predicted separately and the sum of the proportions might not be 1.0. Thus, We further normalized the selection proportions at each branch point to make them a probability distribution (i.e., summing to 1.0). Root-mean-square errors between the predicted

[3] https://lightgbm.readthedocs.io/en/latest/

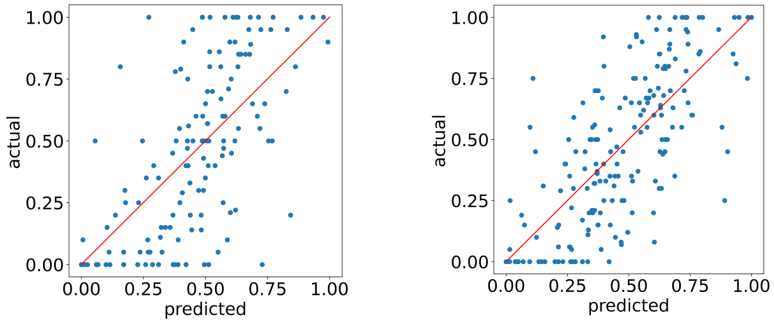


Fig. 2. Prediction results (x-axis: predicted probability, y-axis: actual probability). (a) wide view maze, (b) narrow view maze.

values and actual values were 0.26 (wide view) and 0.21 (narrow view). Although the prediction accuracy still had room to improve, the Pearson correlation coefficients between the predicted values and actual values were 0.66 (wide view) and 0.74 (narrow view). The former had a moderate positive correlation, while the latter had a highly positive correlation. We concluded that the prediction models were reliable to some extent, especially the narrow view one.

We further analyzed the players' tendencies in both wide view and narrow view mazes and had interesting findings that were also reflected in the prediction models. Due to page limit, we only present the results of narrow view mazes. We observed that (1) the players tended to go down or right instead of going up or left and that (2) when both right and down directions were available, the players tended to go straight. Such tendencies are shown in Fig. 3, the statistics on the players' selection proportions of proceeding directions according to the shapes of branch points. To keep the figures simple and easy-to-read, data with selection proportions less than 0.5 are excluded. Also, data with sample numbers less than 20 are excluded since they are not reliable enough.

We considered that tendency (1) is because the end points of all mazes in the experiments were fixed at the bottom-right corner. When further looking into the prediction model, the "cos_goal" feature had the biggest contribution on prediction (a feature importance of 0.35). With a higher "cos_goal" value, the proceeding direction was likely promising for the players. For example, if a player is at some branch point in the upper-right area, going down leads to the highest "cos_goal" value and is reasonable for the player to select. Thus, we concluded that tendency (1) was adequately reflected in the prediction model. As for tendency (2), we considered that going straight is easier to operate than changing directions for human players. This tendency was also reflected in the prediction model, where the "is_straight" feature contributed the second most (a feature importance of 0.15).

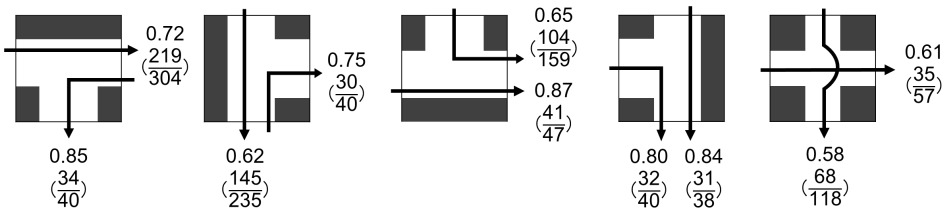


Fig. 3. Human players' selection proportions of the proceeding directions.

4 Maze Generation and Selection Approach

For generated mazes, we aim to evaluate the difficulty from human players' perspectives. With a prediction model (e.g., the narrow view one in Section 3), Section 4.1 creates a test player simulating human players. Section 4.2 then defines difficulty and generates mazes accordingly.

4.1 Test Player Considering Human-likeness

The test player's action selection is broadly divided into two cases according to whether the passage is a branch point or not (defined in Section 3.2). For a passage that is not a branch point, the test player proceeds as follows. The test player directly goes to the end point when the end point is in the view range and can be arrived. Otherwise, the test player moves forward and never step into dead ends (e.g., never going left in the situation of Fig. 1(c)).

The following two exceptions are introduced to the above rules to make the test player more human-like. First, the test player stops proceeding and returns to the last passed branch point when it goes left on the horizontal edges of the mazes or goes up on the vertical edges of the mazes. Since all mazes have end points fixed at the bottom-right corner and do not contain loops, going left or up on the edges never leads to solution paths. The investigation in Section 3 also supports that human players have such a tendency. Second, for narrow view mazes, the test player returns to the last passed branch point upon knowing that the succeeding paths lead to dead ends.

For a passage that is a branch point, the action selection at the first time visiting and at those returning from dead ends is different. When visiting a branch point for the first time, the test player selects directions according to the probability distribution from the prediction models in Section 3.

When the test player returns to a branch points from some dead ends, we predict the selection proportions again for directions that the paths have not been traversed. When the test player is not on the solution path[4], the predicted proportions less than 0.5 are changed to 0.0, which is to prevent the test player's behaviors being too different from human players'. If all the not-traversed directions have proportions of 0.0, the test player returns to the last passed branch

[4] We create the test player assuming that the shortest solutions of mazes are known.

point. The reason for not applying this rule to solution paths is to make sure that the test player can reach the end points. If at least one direction can be selected, the proportions are normalized to sum to 1.0 no matter whether the branch point is on the solution path or not. Taking a T-shaped branch point on the solution path as an example, assume that the predicted proportions for going right and down are 0.6 and 0.3, respectively. The test player goes right with a probability of $0.6/(0.6+0.3) \approx 67\%$ and goes down with 33%.

4.2 Maze Difficulty Evaluation using Test Player

We use the test player in Section 4.1 to simulate human players' behaviors when playing mazes and evaluate difficulty according to estimated step numbers. In more detail, for each maze, we know in advance the shortest number of steps to the goal, denoted by $n_{shortest}$. After the test player clears the maze, we calculate the total number of steps cost by the player, denoted by n_{total}. We then define *the number of extra steps* n_{extra} as $n_{total} - n_{shortest}$. With higher n_{extra}, we consider that the maze is more difficult. Since the test player involves randomness, we let it play each maze several times (10 in this paper) and judge the difficulty according to the results of all trials. We generate mazes automatically using the digging method [12] and sample five difficulty groups as follows:

- $\text{easy}_{\text{lowSD}}$ (low average n_{extra} with low standard deviation).
- $\text{moderate}_{\text{lowSD}}$ (moderate average n_{extra} with low standard deviation).
- $\text{moderate}_{\text{highSD}}$ (moderate average n_{extra} with high standard deviation).
- $\text{difficult}_{\text{lowSD}}$ (high average n_{extra} with low standard deviation).
- $\text{difficult}_{\text{highSD}}$ (high average n_{extra} with high standard deviation).

5 Subject Experiments on Maze Evaluation

We conducted subject experiments to see whether the maze difficulty for human players are well predicted by the test player. Section 5.1 presents settings of the experiments and Section 5.2 shows the experimental results.

5.1 Experiment Settings

A total of 10 players (males in twenties) participated in the experiments. The participants were different from those in Section 3, for the sake of fair evaluation. The maze size was 51×51 and players had the narrow view. We generated 30,000 mazes and let the test player with narrow view prediction model play each maze 10 times. A maze with an average n_{extra} in $[0, 50)$ was classified as easy, $[150, 250)$ as moderate, and $[350,)$ as difficult. In each class, mazes with the top-7 and the bottom-7 standard deviations of n_{extra} were selected as highSD and lowSD, respectively.

Fig. 4(a) and 4(b) show examples of $\text{easy}_{\text{lowSD}}$ and $\text{difficult}_{\text{lowSD}}$ mazes with the test player's trajectories of one trial, where cells in gray are on the solution path and those in red oblique lines are not. Fig. 4(c) shows the trajectories by one of the human players. The participants played the 35 mazes in the order shown in Table 1.

(a) (b) (c)

Fig. 4. Example mazes of (a) $\text{easy}_{\text{lowSD}}$ and (b)(c) $\text{difficult}_{\text{lowSD}}$, where (a) and (b) show the test player's trajectories in gray (on the solution path) and in red oblique lines (not on the solution path), and (c) shows one human player's trajectories.

5.2 Experiment Results

Table 1 shows the means and the standard deviations of the participants' n_{extra} for each maze. Generally, the difficulty judged by the test player matched human players' (i.e., difficult > moderate > easy). When focusing on mazes with lowSD, the Student's t-test ($p < 0.05$) showed that the results were statistically significant. Namely, $\text{difficult}_{\text{lowSD}}$ had the average n_{extra} higher than $\text{moderate}_{\text{lowSD}}$ ($p = 0.0016$), and $\text{moderate}_{\text{lowSD}}$ higher than $\text{easy}_{\text{lowSD}}$ ($p = 0.0083$). We also compared the standard deviations within the same difficulty. We considered that mazes with lower standard deviations are preferred since it means that even different human players may still experience similar difficulty. For mazes with lowSD and highSD, the results of Student's t-test ($p < 0.05$) showed statistical significance as follows: $\text{moderate}_{\text{highSD}}$ had the average standard deviation higher than $\text{moderate}_{\text{lowSD}}$ ($p = 0.0142$), and $\text{difficult}_{\text{highSD}}$ higher than $\text{difficult}_{\text{lowSD}}$ ($p = 0.0057$). The results demonstrated that our method succeeded in generating mazes where the difficulty matched human players' in terms of n_{extra}.

Interestingly, some $\text{easy}_{\text{lowSD}}$ mazes were actually difficult for human players and vice versa, as shown in Table 1. We suspected that the accuracy of the prediction model was insufficient in some situations and the test player somewhat lacked the consideration for humans' assumption and misunderstanding. For example, assume a maze has a very long solution path. When human players cannot reach the end point for a long time, they may return to previous branch points even when they are on the solution path. In contrast, the test player did not have such hesitation. We expected that the difficulty evaluation would fit human players' behaviors better if we introduced more human-like characteristics like this into the test player.

6 Conclusion and Future Work

In this paper, we proposed a procedural maze generation method with considering difficulty from human perspectives, which generated mazes by the following two steps. The first was investigation of human players' tendencies. We employed

Table 1. The means and standard deviations (SD) of n_{extra} by human players.

$\text{easy}_{\text{lowSD}}$			$\text{moderate}_{\text{lowSD}}$			$\text{moderate}_{\text{highSD}}$			$\text{difficult}_{\text{lowSD}}$			$\text{difficult}_{\text{highSD}}$		
order	mean	SD	order	mean	SD	order	mean	SD	order	mean	SD	order	mean	SD
5	200	194	1	71	43	3	53	113	4	299	173	2	257	180
9	72	131	10	241	197	6	243	206	7	364	123	8	231	162
14	69	44	12	285	114	11	195	209	13	371	121	17	447	162
18	77	116	19	220	137	15	135	214	16	406	150	20	368	200
21	175	151	23	228	140	24	232	201	22	262	133	25	265	234
26	48	47	29	178	152	27	228	180	28	517	159	30	104	173
33	27	25	32	235	42	31	314	291	34	374	77	35	140	184
Avg.	95	101	Avg.	208	118	Avg.	200	202	Avg.	307	134	Avg.	259	185

supervised learning and built models for prediction of human players' path selection probabilities. The prediction results had moderate or highly positive correlations to human players' selections. The second was the creation of a test player based on the model. The test player's results of playing maze were used as a measure of difficulty. Then, we conducted subject experiments to evaluate whether the maze difficulty are suitable for human players. The experiments showed that difficulty estimated by the test player matched human players' playing results.

The followings discuss several promising research directions. We expect to improve the difficulty evaluation by making the test player consider more human-like characteristics. Also, the efficiency of maze generation has room to improve. The current approach may generate many mazes before obtaining one with a specific difficulty. We consider that combining other algorithms such as simulated annealing and local search can help improve the efficiency. In addition, we plan to apply the approach to generate mazes containing more elements such as enemies and items for application to RPGs and compare our approach to existing ones.

A Appendix

The following are 23 features, mainly related to each branch point b.

- **maze_size:** The maze size (Small, Medium, or Large).
- **x, y:** The x- and y-coordinates of b.
- **ent:** The direction from which players enter b.
- **proc:** The proceeding direction at b.
- **proc_up, proc_down, proc_left, proc_right:** Whether each direction is an uncertain proceeding direction.
- **dist_wall_up, dist_wall_down, dist_wall_right, dist_wall_left:** Distance from b to the edge in each direction.
- **is_straight:** Whether ent and proc are in a straight line or not.
- **straight_depth:** The number of passages that follow the straight line.
- **promisingness:** The number of passages on the succeeding paths of proc.
- **promisingness_sum:** The sum of the promisingness values of all directions, excluding ent, at b.

- **general_direction:** The general direction of the paths extending from proc.
- **num_branch:** The number of other branch points in the range of narrow view. This feature was used for both wide and narrow view mazes.
- **dist_start:** The minimum number of passages from the starting point to b.
- **avg_step:** The number of steps taken to reach b. We used the average number of steps of the experiment participants.
- **cos_start:** The $cos\theta$ of the angle between the starting point and proc.
- **cos_goal:** The $cos\theta$ of the angle between the end point and proc.

References

1. Volz, V., et al.: Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network. In: 2018 Genetic and Evol. Comput. Conf., pp. 221-228 (2018)
2. Cardamone, L., Yannakakis, G. N., Togelius, J., Lanzi, P. L.: Evolving Interesting Maps for a First Person Shooter. In: 2011 Eur. Conf. on the Appl. of Evol. Comput., pp. 63-72 (2011)
3. Louchart, S., Aylett, R.: Solving the Narrative Paradox in VEs - Lessons from RPGs. IVA 2003 Lecture Notes in Artif. Intell., **2792**, pp. 244-248 (2003)
4. Togelius, J., Kastbjerg, E., Schedl, D., Yannakakis, G.N.: What is Procedural Content Generation?: Mario on the Borderline. In: 2nd Int. Workshop on Procedural Content Gener. in Games, pp. 1-6 (2011)
5. Togelius, J., et al.: Multiobjective Exploraiton of the StarCraft Map Space. In: 2010 IEEE Conf. on Comput. Intell. and Games, pp. 265-272 (2010)
6. Oikawa, T., Hsueh, C.H., Ikeda, K.: Improving Human Players' T-Spin Skills in Tetris with Procedural Problem Generation. In: Adv. in Comput. Games, pp. 41-52 (2020)
7. Soares, E. S., Bulitko, V.: Deep Variational Autoencoders for NPC Behaviour Classification. In: 2019 IEEE Conf. on Games, pp. 1-4 (2019)
8. Togelius, J., Yannakakis, G. N., Stanley, K. O., Browne, C.: Search-Based Procedural Content Generation: A Taxonomy and Survey. IEEE Trans. on Comput. Intell. and AI in Games, **3**(3), pp. 172-186 (2011)
9. Summerville, A., et al.: Procedural Content Generation via Machine Learning (PCGML). IEEE Trans. on Games, **10**(3), pp. 257-270 (2018)
10. Nam. S., Ikeda, K.: Generation of Diverse Stages in Turn-Based Role-Playing Game using Reinforcement Learning. In: 2019 IEEE Conf. on Games, pp. 1-8 (2019)
11. Khalifa, A., Bontrager, P., Earle, S., Togelius, J.: PCGRL: Procedural Content Generation via Reinforcement Learning. In: 16th AAAI Conf. on Artif. Intell. and Interactive Digit. Entertainment, pp. 95-101 (2020)
12. Algoful, https://algoful.com/Archive/Algorithm/MazeDig. Accessed Sep 2021
13. Algoful, https://algoful.com/Archive/Algorithm/MazeExtend. Accessed Sep 2021
14. Algoful, https://algoful.com/Archive/Algorithm/MazeBar. Accessed Sep 2021
15. Susanto, E.K., Fachruddin, R., Diputra, M. I., Herumuti, D., Yunanto, A. A.: Maze Generation based on Difficulty using Genetic Algorithm with Gene Pool. In: 2020 Int. Seminar on Appl. for Technol. of Inf. and Commun., pp. 554-559 (2020)
16. Adams, C., Louis, S.: Procedural Maze Level Generation with Evolutionary Cellular Automata. In: 2017 IEEE Symp. Ser. on Comput. Intell., pp. 1-8 (2017)
17. Kwiecień, J.: A Swarm-Based Approach to Generate Challenging Mazes. Entropy, **20**(10), 762 (2018)