An Evolutionary Approach for Procedural Content Generation in Cooperative Game Overcooked

In-Chang Baek[O], Tae-Gwan Ha, Kyung-Joong Kim*
Gwangju Institute of Science and Technology
{inchang.baek@gm., hataegwan@gm., kjkim@}gist.ac.kr

협력 게임 Overcooked에서 절차적 컨텐츠 생성을 위한 진화적 접근

백인창[O], 하태관, 김경중*
광주과학기술원

Abstract[1]

Recently, procedural content generation (PCG) shows a diverse approach to generating content in the game research community. The game map generation task has been one of the famous areas for PCG researchers. However, most works are studied on a single-player game, and few are studied on multi-player games. So, we opt to extend single-player studies into a multi-player problem using controllable PCG technique. In this paper, we propose an evolutionary approach to generate cooperative game maps for a multi-player game, Overcooked. Our contribution can be summarized into two-fold: (1) design a representative method to apply the procedural generation method for the game, and (2) generate maps with a parameterized fitness function. To demonstrate our method, we use a genetic algorithm as an evolutionary algorithm and analyze the results in qualitative and quantitative ways. In conclusion, our method successfully generates cooperative game maps with controllable parameters.

1. Introduction

Procedural content generation (PCG), which automatically generates game contents to guarantee cost-efficiency, has been studied for several years to generate a wide range of content such as images and music, and design architectures and circuits. In this community, various strategy has been used to enrich the content's quality and diversity. Among these strategies, controllable PCG allows the player or designer to utilize some set of parameters as the controller of the generator. Generators can be capable of producing content which layout diversely designed by the set of parameters. One of the approaches of controllable PCG is a genetic algorithm which generation occurs through the search space modified by the designer [1].

However, many of previous works are studied on single-player games, and few are studied on multi-player games. Compared with single-player case, a multi-player game has different types of relationships such as competition [2] and cooperation [3] so that content generator should be concerned with this interaction issue. For example, the generator can make the composition of the map become inefficient and asymmetric to intend more cooperative play from the players. Overcooked game, which goal is to prepare meals along with another player under a time limit, requires high-level joint strategy or motion coordination. In practice, several works employed this cooperative environment to improve multi-agent or human-robot coordination [3, 4, 5].

We suggest a new map generator for the Overcooked game based on a genetic algorithm in which genotypes represent the composition of the map. The algorithm is applied to research in the form of controllable PCG, which parameter of fitness function related to the composition of the map. The objective of this algorithm is to create various types of maps according to the set of fitness functions and parameters.

The experiment in this paper focuses on the analysis of the performance of the genetic algorithm as controllable map generation and the suitability of the evolutionary approach for multi-player games. We find the frequency of each block to ensure that the blocks are created with the condition we intend. Additionally, we visually check how the map is changed according to the set of parameters or the number of rooms.

2. Background

2.1 Genetic Algorithm

Genetic algorithm (GA) is one of the evolutionary algorithms for effectively searching for an optimal solution in several problems. To find an optimal, GA repeats four sequences (selection, crossover, mutation, and replacement) for a population. The population consists of various chromosomes as solution to the problem, and the optimality is calculated with a fitness function. Repeating the sequence, weak chromosomes are eliminated naturally, and the suboptimal chromosome becomes close to optimal with genetic variants. To apply this mechanism, we newly design our chromosome and fitness function for our task. From the perspective of controllable PCG, genetic algorithm has the advantage of being able to create various maps with its robust exploration.

2.2 Overcooked! Game

Overcooked! game is a multi-player video game available on online[2]. This game has a simple goal, cooking several foods in as many orders as possible under a set period time. To complete the order, each player should work together to prepare ingredients, cook them, and serve them

[1] This research was supported the National Research Foundation of Korea (NRF) funded by the MSIT (2021R1A4A1030075). *Corresponding Author
[2] https://www.team17.com/games/overcooked/

on a plate. That would be enough as is, but the difficulty comes from how limited player abilities are. Players can only be carrying one item at a time, and the player can't move around everything in the kitchen to expedite things. Also, the composition of the kitchen is designed in asymmetric ways, so their success or failure will almost entirely depend on how effectively the player can communicate with another player.

Employing this cooperative environment, several machine learning research used this game to evaluate their collaborative agents or coordination algorithms of agents [4, 5]. Instead of agent policies, another work [3] emphasized the significant effect of the environment on human-robot coordination. In this paper, the cooperative environment can be divided into compositions which can be numerically parameterize. Then, we can combine these various numbers of items to interact and paths to travel. For example, a combination of various item sets leads to different strategies, and the number of rooms induces new path planning and motion coordination for various player number.

3. Proposed Method

3.1 Layout Representation To apply PCG methods to 2D game maps, lots of studies have been used tabular representation, encoding visual 2D game layout to easily transformable data. As shown in Fig. 1, we define six game blocks into numbers and encode the game layout using them. We consider the game layout can be represented to the numeric matrix and flatten it into an array of a chromosome. Once we generate the game layout with a genetic algorithm, we can reshape to 1D array into a 2D matrix reversely and load it on the game.

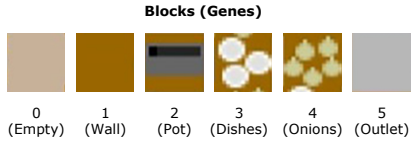


Figure 1 Six blocks to build the layout. We define the genotype into an integer value range from 0 to 5.

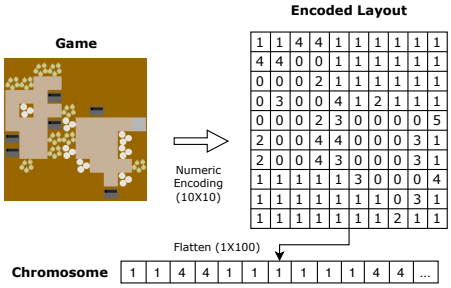


Figure 2 Representing a game layout into a chromosome.

3.2 Population Initialization The initial population is helpful to search for an optimal chromosome in GA. A carefully designed initializer remarkably reduces the searching space in this domain [1]. We also define an initializer to reduce searching space, initializing genes with a pre-defined rule (i.e., used domain knowledge) for selecting game blocks. Instead of uniform random distribution, the weights set to [0.7, 0.1, 0.05, 0.05, 0.05, 0.05] for six blocks, respectively. The probability is set how many each block is used in general. This method helps to generate reasonable maps in the lower evolution step.

3.3 Evolutionary Method As aforementioned, GA repeats four sequences to enhance a population. In each step, we select tournament selection, two-point crossover, and random-resetting mutation as our baseline. To evaluate individuals, we build a fitness function for generating the game map and can be found on Eq. 1~4. Fitness for conditional generation $fitness$ is a summation of room count f_{RC}, room size f_{RS}, and block count f_{BC}, respectively.

$$fitness = f_{RC} + f_{RS} + f_{BC} \quad (1)$$

$$f_{RC} = -|c - C| \quad (2)$$

$$f_{RS} = -stdev(s_1, s_2, s_3, \dots, s_c) \quad (3)$$

$$f_{BC} = -\frac{1}{4}\sum_{i=2}^{5}|b_i - B_i| \quad (4)$$

The important point to a generate cooperation map is to design game maps into separate rooms to make the interaction between players, and the room size should be enough to get around. To satisfy these conditions, we calculate the room counts and size of each individual (Fig. 3) and build f_{RC} to room count c close to C and f_{RS} lower variance of room sizes $(s_1, s_2, \dots)$ into zero.

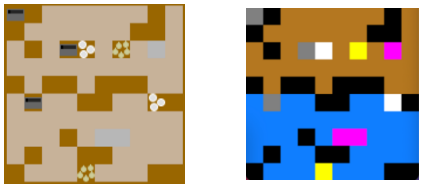


Figure 3 Divided rooms and placed blocks. Left is an original game screen and the right shows how we divide the map into two rooms.

Generating diverse maps is one of the main issues in PCG research. There are two ways in our problem, one is scaling the layout and the other is placing the resources (block 2, 3, 4, and 5) in various ways. In our paper, we fix the layout size to 10×10 and regulate the number of resources that makes diverse game strategy and difficulty. So, we build the term f_{BC} to regulate the four blocks with a block number i, current block count b_i and target count B_i. We build this equation similarly to f_{RC} and this term makes the population to fit the block count parameters. Additionally, we regularize this term by dividing to 4 blocks so that f_{BC} do not overwhelm other terms when types of blocks increase.

3.4 Heuristic Fixing To enhance the quality of generated maps, we remove unavailable resources (unreachable to both players), replacing the genes to *Wall* blocks as post-processing. This process is taken after the generation task is done and makes it fancier when visualizing the chromosome, not harming the generating process.

4. Experiments

The main goal of our experiment is to generate game maps with the number of players and resource placement in various settings. In this paper, we define five parameters that account for the number of rooms c and number of block 2~5 $B_{2\sim5}$. We set these parameters for $c = 2, 3, 4$ (2~4 players) and $B_{2\sim4} = 3, 5, 7$ as shown in Fig 4. B_5 is always set to 1 because there is no need for variations in this game. We conduct the experiment in the following steps: (1) Create a population that has 100 chromosomes. Each individual has 100 genes and initializes with weighted random function in Section 3.2. (2) Conduct the GA

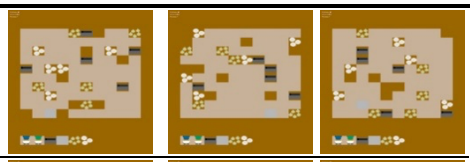
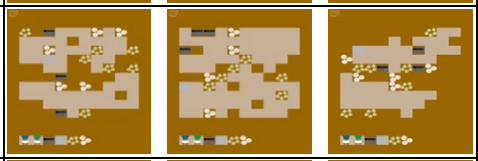
#Rooms	#Blocks	3 of Generated Results	#Rooms	#Blocks	3 of Generated Results
$C = 2$	$B_2 = 3$ $B_3 = 3$ $B_4 = 3$		$C = 1$	$B_2 = 5$ $B_3 = 5$ $B_4 = 5$	
$C = 2$	$B_2 = 3$ $B_3 = 5$ $B_4 = 7$		$C = 2$	$B_2 = 5$ $B_3 = 5$ $B_4 = 5$	
$C = 2$	$B_2 = 7$ $B_3 = 7$ $B_4 = 7$		$C = 3$	$B_2 = 5$ $B_3 = 5$ $B_4 = 5$	

Figure 4 Generated maps for each parameter. Left-side images show variations of block counts (pot, dish, and onion) and right-side images are variations on room count (number of players).

process by calculating our fitness function in Section 3.3. (3) Repeat previous steps during 500 generations. (4) Save the best chromosome in the population with parameters. We repeat the above process a hundred times for six parameter sets. Additionally, probability of crossover set to 0.9 and probability of mutation to 0.01 for GA hyper-parameters.

5. Result and Discussion

In this section, we discuss how the maps are generated in qualitative and quantitative ways. There are a total of 6 cases for comparison, 3 cases on variations on block count, and remains for variations on room count. We visualize the best chromosomes in each experiment in Fig. 4 and summarize the quantitative results in Fig. 5 and 6.

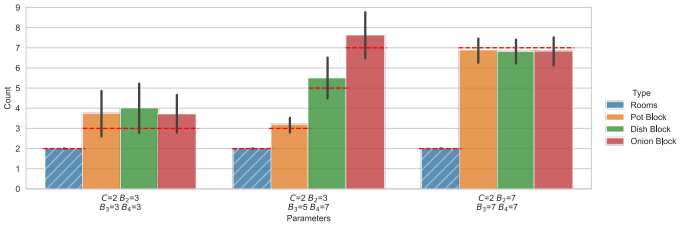


Figure 5 Quantitative results for variations on block counts. Red dashed lines mean our experimental parameter.

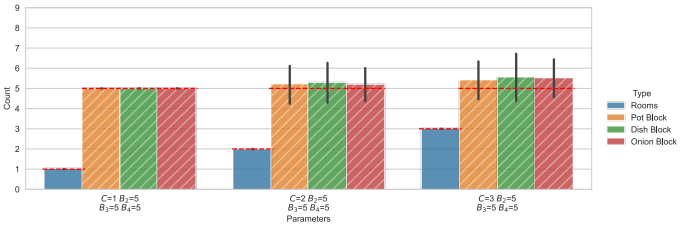


Figure 6 Quantitative results for variations on room count.

As quantitative analysis, the room counts (C) strictly fit our parameters in well, and block count (B_i) is approximately closed to ours, even if the condition B_i is different from each other. It is because the term f_{RC} accounts larger potion than f_{BC}, and it means that we can utilize these sub-optimal chromosomes as variously generated maps when we make a constraint on player number.

However, as shown in Fig. 4, the quality of maps is getting low when the constraint values are increasing, in both cases of C and B_i. In the case of B_i, the increasing resource blocks placed between the rooms, and disturb the interaction between players. In the case of C, each room gets far away so that makes the maps un-playable when room count is increasing (especially on $C = 3$). In our empirical analysis, it's easier to make room small when making the term f_{RS} lower, in terms of the variance. To solve this problem, it may need an additional fitness term to make the outputs more feasible.

6. Conclusion and Future Work

We propose a new approach for generating map layouts with a genetic algorithm. To generate the game maps, some parameters for multi-player and placing game resources are formularized into fitness function. And it successfully generates diverse layouts with some conditions that regulate difficulty and strategic diversity. As our limitation, we simply assume that the cooperation is accrued from an asymmetric condition between players and designed into fitness function with our domain knowledge. In future work, we will regard several cooperation methods and generate maps regarding various playing strategies. Also, generating more feasible (playable) maps could be valuable work.

References

[1] Lucas Ferreira, Claudio Toledo, *et al.* "A Search-based Approach for Generating Angry Birds Levels." *Proceedings of 2014 IEEE Conference on Computational Intelligence and Games* (2014): 1-8.

[2] Georg Volkmar, Nikolas Mähllmann, *et al.* "Procedural Content Generation in Competitive Multiplayer Platform Games." *International Conference on Entertainment Computing and Serious Games* (2019): 228–234.

[3] Fontaine, Matthew C., *et al.* "On the Importance of Environments in Human-Robot Coordination." *arXiv preprint arXiv:2106.10853* (2021).

[4] Carroll, Micah, *et al.* "On the utility of learning about humans for human-ai coordination." *Advances in Neural Information Processing Systems* 32 (2019): 5174-5185.

[5] Sarah A Wu, *et al.* "Too many cooks: Bayesian inference for coordinating multi-agent collaboration." *Topics in Cognitive Science* (2021), 13(2): 414–432.