PlayMapper: Illuminating Design Spaces of Platform Games

Vivek R. Warriar[1], Carmen Ugarte, John R. Woodward[2] and Laurissa Tokarchuk[3]
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
{v.r.warriar[1], j.woodward[2], laurissa.tokarchuk[3]}@qmul.ac.uk

***Abstract*—In this paper, we present PlayMapper, a novel variant of the MAP-Elites algorithm that has been adapted to map the level design space of the Super Mario Bros game. Our approach uses player and level based features to create a map of playable levels. We conduct an experiment to compare the effect of different sets of input features on the range of levels generated using this technique. In this work, we show that existing search-based techniques for PCG can be improved to allow for more control and creative freedom for designers. Current limitations of the system and directions for future work are also discussed.**

***Index Terms*—MAP-Elites, Procedural Content Generation, Platform Games, Evolutionary Algorithms, Illumination Techniques.**

I. Introduction

Procedural Content Generation (PCG) refers to using algorithms to create game content. Generating content is formalized as a search problem where the search space is all possible content that can be generated. Evolutionary search-based techniques attempt to find appropriate solutions, usually measured by the playability of this content. There exists a number of search-based strategies to find playable and diverse content for platform games [1], [2]. These systems are limited in terms of control. In grid-based puzzle games, a desired level of control can be introduced using evolutionary techniques. These can be used to arrange game levels ranging from easy to difficult in terms of game-play [9]. Search-based PCG systems can be used along with these approaches to generate content across a spectrum of difficulty for players. However, this must be conducted in addition to the search based PCG process which can be computationally expensive. Interestingly, this is a comparatively simple task for a human game designer. To address this gap we present PlayMapper (PM), a PCG system based on the MAP-Elites (ME) algorithm [3] built to generate levels for platform games. This pilot work is implemented in the Super Mario Bros. environment [4]. We refer to the representation or encoding of the solution as in genotypic space while the fitness of the solution is measured in-game simulation or phenotypic space. The choice of representation in the genotypic space and mapping between these 2 spaces will impact the quality of content generated by the algorithm. Fitness refers to what extent the level is playable. Following related work [2], we use an A* agent to measure the playability of generated levels.

The ME algorithm is an illumination technique [3] that aims to return the highest performing solution for each point in a designer-defined feature space. ME belongs to a broader family of techniques known as *Quality-Diversity* algorithms. As the name implies these techniques aim to return a set of diverse solutions of high fitness [10]. In games, this technique has been adapted for the bullet hell genre in the Talakat system [5]. This variant of ME has been also used to generate level segments of the Super Mario Bros. game [11]. PM is different as it does not use the constraint ME variant from [5], [11]. Additionally, we explore different behaviour dimensions than [11] and the level segments generated in their approach are of fixed size. To the best of our knowledge, PM is the first system that uses an illumination technique to generate platform game levels with promising control over the size of the levels generated.

In this paper, we describe the PM system in section II. Section III describes the study we conducted to evaluate the PM system. Results from the study are presented in Section IV. Finally, the implications of these results, limitations, and future work are discussed in section V. Initial results indicate that PM can be used to generate a diverse set of game levels across a number of designer-defined behavioural spaces. This is beneficial to the domains of both PCG and AI-assisted design for games.

II. PlayMapper

Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) is an illumination algorithm that aims to find the highest-performing solution for each point in a designer-defined feature space [3]. The algorithm returns a map of solutions along with a corresponding measure of fitness for each. The different axes of this map correspond to designer-defined features. For example, in this work, we explore the effects of using different features on the set of levels generated by the PM system (details provided in the next section). The current implementation of the system creates 2D maps. This can be extended to create maps of any dimension. This version of PM does not use crossover operators and relies on mutation.

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

The following subsections describe the main components of the system.

A. Designer-defined Features

Human-centric studies that evaluate the player experience of PCG systems indicate that there are 2 categories of features to consider: player behaviour features (PBFs) and controllable game features (CGFs) [6], [7]. PBFs refer to game metrics that have been calculated from a simulation-based evaluation with the A* agent or from gameplay data from people, while CGFs refers to a property of the level itself (unrelated to the game-playing agent). The PM system uses these features to return a set of playable levels. This is ideal because these features are of interest to designers when evaluating the player experience of game content [6], [7]. Furthermore, it is interesting to note that generating levels across a selected range of PBFs is not a simple task for even human designers. In this paper, PBFs are measured through A* agent gameplay. This is arguably a naive approach and research into more human-like agents will improve the PM system. The final diversity and playability of the levels generated are dependent on the choice of input features. An observation that has motivated this pilot study.

B. Mutation Operations

Mutation operations are important as they introduce genetic diversity into the search process. Mutation operations prevent the algorithm from converging onto a set of solutions of weak diversity (that is, when all levels produced by the PCG system are similar to each other). It is important to note that the mutation operators depend on the content representation or the genotype-to-phenotype mapping. Here, we follow the encoding described in [2]. A level is encoded as a 2D matrix of discrete symbols. Each symbol corresponds to an element in the level (for example, ground, coins, enemies, etc). This is a direct encoding as compared to other, more indirect ones such as grammars [1]. We have implemented 3 mutation operations:

1) Changing Mutator: This operation loops through each symbol within the encoding of a single level, and depending on a predefined probability, changes the symbol to another symbol used in the encoding. We use a changing mutation probability of 0.5%.
2) Growing Mutator: This operation loops through each column of the level encoding, and depending on a defined probability, creates a copy of the column and adds it to the end of the level. The goal of the operation is to allow for larger levels to be obtained from smaller levels. We have constrained this operation, limiting the level size to a specified range. We use a growing mutation probability of 5% and the size of the levels have been constrained to 146-444 columns in length.
3) Shrinking Mutator: This operation is similar to the growing mutator. The main difference is that as it loops through each column, there is a 5% probability that the column will be deleted. Similar constraints to the size of the level have been applied to this operation as well.

III. Experimental Settings

As mentioned in the previous section, the quality and diversity of the levels generated by the PM system are highly dependent on the choice of designer-defined features. A related heuristic to consider is the numerical range of the features in the map. We have chosen the following features and corresponding ranges in this study:

1) *Size* of the level, which is given by the number of columns in the encoding. The size range is 146-444 columns in length.
2) *Time* for the agent to complete level, which is normalized and ranges between 0-1.
3) *Speed* of agent. This is the ratio between the distance travelled and the time taken by the agent. This value is normalized and ranges between 0-1.
4) *Jump-entropy*, which is the ratio between the number of jumps and the total number of actions made by the agent across the level. We use this feature as a way of controlling the difficulty of the game (more jumps is considered more difficult). This range has been set between 0-0.1 (at 0.1 the Agent Jumps 10% of the time).

Speed, Time and Jump-entropy are PBFs while Size is a CGF. The ranges reported have been selected based on initial trials. The following subsections describe the experiment protocol and the quality metrics we have followed.

A. Experimental Protocol

We compare the quality and diversity of levels generated from different designer-defined feature maps. Since the entropy of player input has been a feature of interest in a previous MAP-Elites implementation for PCG in games [5], we keep Jump-Entropy consistent across one of the dimensions of the 2D map in all the study conditions. We compare the following 3 conditions: Size vs. Jump-Entropy, Time vs. Jump-Entropy and Speed vs. Jump-Entropy. These will be referred to as the Size condition, Time condition and Speed condition respectively. We initialize our algorithm with a starting population of human-designed levels from the video game level corpus [8]. The results reported are after 2000 iterations of a single run of the ME algorithm. Another important heuristic is the resolution of the map - we follow a 20×20 resolution.

B. Quality Criteria

We have adapted a subset of quality metrics described in [3] for this study. The following 2 metrics have been calculated:

1) Coverage: The ratio between the number of populated (non-empty) cells and the total number of cells in the map.
2) Reliability: The ratio between the number of cells populated **only by playable levels** and the total number of cells.

IV. Results

This section presents results across the 3 conditions of this study. Table I shows the scores for reliability and coverage at the beginning and the end of the run. The Size condition had

TABLE I
COVERAGE AND RELIABILITY SCORES BEFORE AND AFTER RUN

	Coverage		Reliability	
	Before	After	Before	After
Size Condition	0.045	0.213	0.017	0.123
Time Condition	0.035	0.510	0.012	0.068
Speed Condition	0.042	0.413	0.010	0.093

the least coverage and was most reliable. The Time condition had the highest coverage and was least reliable. This can be seen in the maps that have been visualized in Fig 1. The evolution of these scores over 2000 iterations is seen in Fig 2. A comprehensive visual inspection of the aesthetics of these levels is still in progress. Fig 3 shows some example snippets of levels that were generated. Initial inspection reveals that the visual aesthetic of the levels is different between conditions. We observe that none of them look human-designed. A number of broken pipes and redundant placements are seen across all 3 conditions. An example of a redundant placement would be placing a coin where there is no platform to collect it. In the size condition, we observe the expected step-wise change in level length while moving across the cells of the size dimension of the map (albeit some cells remain empty in 2000 iterations).

V. DISCUSSION

For this pilot study, we have reported results after 2000 iterations because we are interested in validating this approach. However, this has resulted in a large number of empty cells in each map. Having more iterations will ensure better coverage and reliability. These early results show that a promising level of control can be introduced to existing search-based PCG solutions. However, this is an early proof of concept implementation with a number of limitations. These are: (1) the visual aesthetic of the levels do not look human-designed (2) it is unclear which are the best features to use to create fun, diverse and controllable content (3) it is unclear how computationally intensive it will be to populate an entire map of playable levels.

The limitation of unsatisfying visual aesthetics is due to the direct encoding and the related mutation operations used in this study. We propose that a more indirect encoding (and complementary mutations) such as grammars [1] will fix this problem. Another interesting approach would be to match the expected visual patterns from Mario levels using the Deep Convolutional Generative Adversarial Network described in [2]. These are directions for potential future work.

Before we can address the visual style of the levels, the second and third limitations will need to be addressed with follow-up studies. This will enable us to make meaningful recommendations about what features to use and the number of iterations. This paper has partially explored the potential of using 4 features in the PM system. We are in the process of conducting a comprehensive analysis with two aspects of interest. First, we are interested to find a meaningful dimension of the map in our generation process. We currently use 2D. However, we are interested in exploring how 1D, 3D and 4D maps would impact the level output of the PM system. The second aspect of interest is, what constitutes the best sets of feature inputs for the PM system. In this study, we use level size and agent time, speed and jump-entropy (1 CGF and 3 PBFs) as features. In other work, we are also interested in how we can computationally model player experience using PBFs and CGFs [6]. With continued exploration in these directions, we believe that PCG systems will be able to generate content that maps to ideal player experiences across a multidimensional range of features of interest to game designers.

VI. CONCLUSION

PlayMapper is the first known implementation of an illumination technique for the platform game genre that controls the size of the generated levels. Although considerable future work must be conducted for it to be used commercially, initial results show the promising level of control it brings to the search-based PCG process. As AI techniques become more popular in game technology, the problem of blackbox systems will become more prevalent. We believe that PlayMapper and similar techniques will reduce the mystery behind existing blackbox PCG systems, making them more controllable and potentially more explainable.

ACKNOWLEDGMENT

The authors would like to thank Oliver Withington for insightful discussions. This work is supported by the EPSRC and AHRC Centre for Doctoral Training in Media and Arts Technology (EP/L01632X/1).

REFERENCES

[1] N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius and M. O'Neill, "Evolving levels for Super Mario Bros using grammatical evolution." 2012 IEEE Conference on Computational Intelligence and Games (CIG), Granada, 2012, pp. 304-311.

[2] Volz, Vanessa, et al. "Evolving mario levels in the latent space of a deep convolutional generative adversarial network." Proceedings of the Genetic and Evolutionary Computation Conference. ACM, 2018, pp. 221-228.

[3] Mouret, Jean-Baptiste, and Jeff Clune. "Illuminating search spaces by mapping elites." arXiv preprint arXiv:1504.04909 (2015).

[4] Togelius, J., Shaker, N., Karakovskiy, S. and Yannakakis, G.N., 2013. "The mario ai championship 2009-2012." AI Magazine, 34(3), pp.89-92.

[5] Khalifa, Ahmed, et al. "Talakat: Bullet hell generation through constrained map-elites." Proceedings of The Genetic and Evolutionary Computation Conference. ACM, 2018, pp. 1047-1054.

[6] Vivek R. Warriar, John R. Woodward and Laurissa Tokarchuk "Modelling Player Preferences in AR Mobile Games" accepted for publication in IEEE Conference on Games (COG) 2019.

[7] Pedersen, Christopher, Julian Togelius, and Georgios N. Yannakakis. "Modeling player experience for content creation." IEEE Transactions on Computational Intelligence and AI in Games 2.1, pp. 54–67, 2010.

[8] Summerville, Adam James, et al. "The vglc: The video game level corpus." arXiv preprint arXiv:1606.07487 (2016).

[9] Ashlock, Daniel, and E. J. Montgomery. "Applying an adaptive generative representation to the investigation of affordances in puzzles." 2019 IEEE Congress on Evolutionary Computation.

[10] Pugh, Justin K., Lisa B. Soros, and Kenneth O. Stanley. "Quality diversity: A new frontier for evolutionary computation." Frontiers in Robotics and AI 3 (2016): 40.

[11] Khalifa, Ahmed, Michael Cerny Green, Gabriella Barros, and Julian Togelius. "Intentional Computational Level Design." arXiv preprint arXiv:1904.08972 (2019).

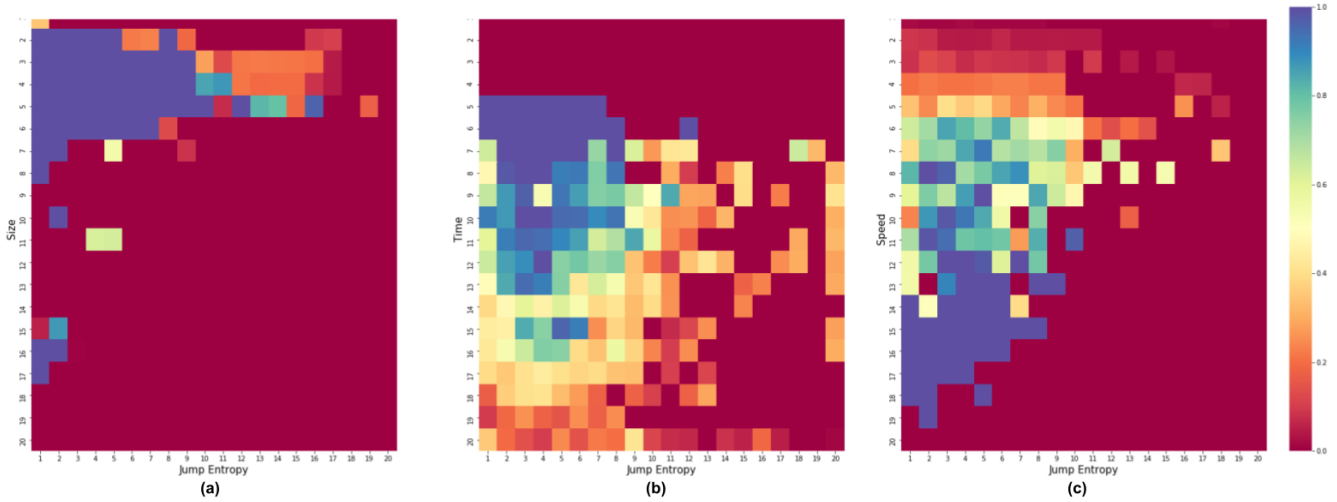

Fig. 1. The figures [a-c] show the visualized maps at the end of 2000 generations. The color of the cell indicates the fitness of the level contained within it. A cell with 0 fitness is an empty cell. [a] shows the size condition, [b] Shows the time condition and [c] shows the speed condition respectively.

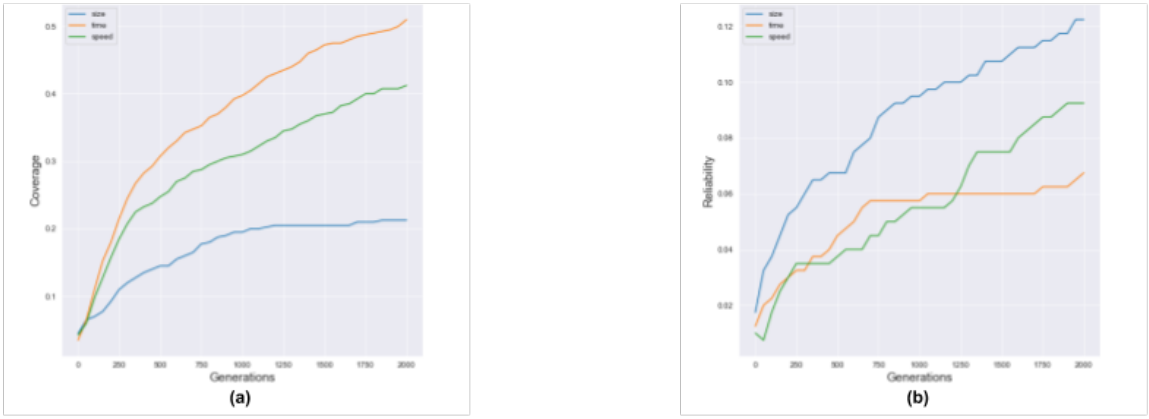

Fig. 2. The figures [a-c] show the visualized maps at the end of 2000 generations. [a] shows the size condition, [b] Shows the time condition and [c] Shows the speed condition respectively.

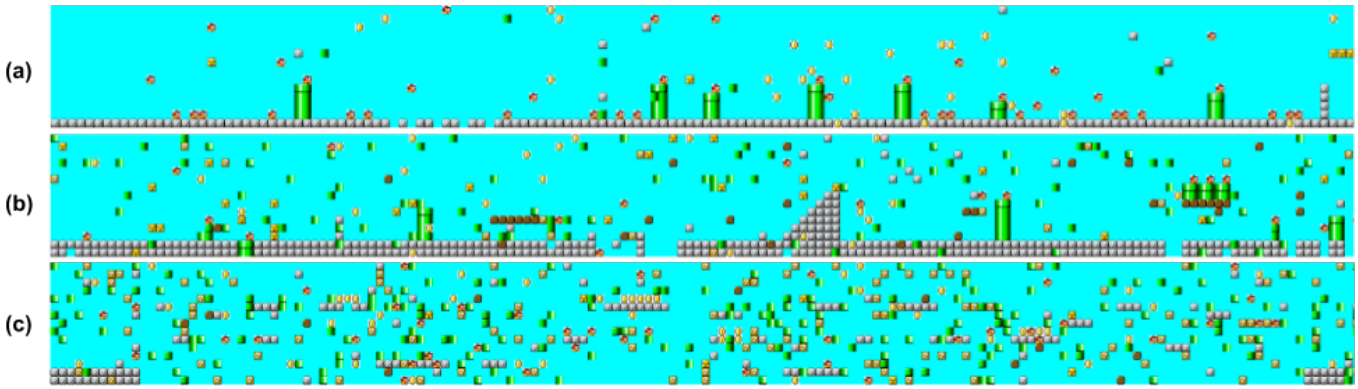

Fig. 3. The figures [a-c] shows example levels visualized from each of the conditions. [a] shows the Size condition [b] shows the Time conditions and [c]shows the Speed condition respectively. Note that only snippets of the levels are shown.