

# Procedural generation of levels for the angry birds videogame using evolutionary computation

Jaime Salinas-Hernández  
Department of Graduate Studies  
Tijuana Institute of Technology  
Tijuana, México  
salinas.jaime1217@gmail.com

Mario Garcia-Valdez  
Department of Graduate Studies  
Tijuana Institute of Technology  
Tijuana, México  
mario@tectijuana.edu.mx

JJ Merelo  
Department of Graduate Studies  
Universidad de Granada  
Granada, Spain  
jmerelo@ugr.es

**Abstract**—In this paper an evolutionary computation-based system is proposed to generate levels for the Angry Birds video game, the levels themselves are composed of a set of birds that the player can use, a certain amount of pigs that must be destroyed in order to progress to the next level and a given number of pieces that conform structures that may or may not protect the pigs from the player. The focus in this stage is the generation of diverse structures that can be played in the game having the additional characteristics of being interesting and playable. We propose a multi-layered search, first constructing composite structures from basic components, to then build more complex structures building from these components, following an open-ended evolution approach; in which the evolution of structures is not guided towards a single objective but rather is free to evolve and generate novelty or diversity. The objective is then that the proposed levels are evaluated by how complex they become and how different they are from the rest of the group. The experiments conducted show that a evolutionary approach allows the generated levels to have a level of uniqueness because on most of the cases the way they are presented on screen is a way a normal human would have trouble to create but at the same time are interesting to see.

**Index Terms**—genetic algorithm, procedural generated content, open-ended evolution, evolutionary computation

## I. INTRODUCTION

Procedural content generation is an interesting subject on the entertainment industry, primarily in video game development because of the savings in time and human resources when generating content for different video game projects [1], [2]. In this paper an evolutionary content generation system is proposed to generate levels for the popular Angry Birds video game, taking inspiration in concepts of open-ended evolution. Angry birds is a game developed by Rovio Entertainment in which the objective is to destroy structures placed on the map as well as eliminating green pig-like entities in order to obtain the highest score possible, the game uses a gravity and collision system that allows the structures to fall and be destroyed by means of damage as well as the use of three different construction materials *ice*, *wood* and *stone* each one with a different resistance, ranging from fragile, normal and sturdy in that order. The game mechanic is to use a slingshot to shoot a certain number of birds with different abilities, to destroy both the structures and the objectives. A high score is achieved by doing the most destruction with the least birds possible [3].

The main objective of this project is to generate levels that can be played by naïve or intelligent agents as well as physical players. The levels themselves need to be complex in the sense that interesting structures can be created, but at the same time, it is possible to complete in some way [4], [5].

Procedural content generation provides a peculiar case of study because of the way content has to be made for a each particular type of game. In this use case we have those restrictions dictated by the game, and also those imposed by a competition track [6]. provide an interesting The challenge for the participating algorithms is to be flexible on the pieces that can be used and the running time of the algorithm, as they are determined at runtime. In order to tackle this case study, we proposed the use a genetic algorithm capable of creating game levels having the objective of being both *interesting* and *playable*.

In this work we found that an evolutionary algorithm can be used to create this type of content not only because it is able to create levels that are interesting and diverse but because an evolutionary algorithm is capable of creating things a normal human could have trouble imagining of.

We propose a basic Open-Ended Evolution (OEE) algorithm, a variant of genetic evolution algorithm where the objective differs from that of a regular genetic algorithm, in this case the final objective is not to reach a final state of the evolution (an optimal solution) but to continue evolving and creating new types of entities, perhaps more complex on each new generation. Solution entities have the possibility to evolve not only inside the frame of their type, but they can create entirely new groups [7]. Taylor et al. [8], [9] defines OEE as a system capable of creating novelty beyond a point where nothing changes on the current group of interest rather than just converging to a stable or quasi-stable state. For this project, OEE is defined simply as a genetic evolution process with different classes of objects, with compositional and taxonomic relationships between them. Objects of each class can evolve in parallel or as in this case, independently of each other. The idea is based on the object oriented model, in which complex behavior can be model using these principles. The rest of the paper is as follows.

Section 2, contains a quick overview of some aspects used by different authors in order to tackle this problem, their design

and basic functionality, in Section 3 the description of the current state of the project is explained, from the basis of the development, modification to the original proposed method as well as different approaches that were taken to solve the problem. finally in Section 4 we provide our conclusions about the development of this paper as well as improvements that can help to refine the proposed work.

## II. STATE OF THE ART

In this Section we present PCG works relevant to the Level generation for the Angry Birds video game. Renz et al. [6], [10] organized an AI competition with the aim of developing intelligent agents that could play new Angry Birds levels better than the best human players. The competition was then collocated in other AI conferences. In order to train and test these agents, more novel and some times difficult levels must be generated. A related competition was then proposed by Stephenson et al. [4] this time with the goal of developing a level generator for the game. Several generators have been proposed in the following years of competition. M. Kaidan et al. [11] propose a model in which the generation of content is controlled by the skill of a player while actively playing the game, the moment the player finishes a certain level the algorithm obtains the score and generates the next level to be played. Y. Jiang et al. [12] proposed the use of predefined letter style patterns and combined to create a small pool of words and letters used to be combined in order to create levels with text on them, the participants would play the game and after that they would be asked if they liked the layout of the levels. A search based approach has been also used in several works.

## III. PROPOSED METHOD

In this Section we describe the approach used in this work, explaining first, the restrictions of the game. The game contains a total of 11 block shapes as shown in Figure 1. These basic blocks cannot be modified, it is not possible to add more or modify the existing ones, with this in mind the purpose is to create composites of these blocks in different locations and angles in order to create structures that can be used as building blocks for more complex structures.

### A. Composite Generation

Since there is no way to *add* a new composite to the pool of blocks, then a way to combine both is needed, in order to do this first the original 11 pieces of the game will be converted into composites of a single element. A composite is defined as a lists of blocks, then a *composite* of one element will be created for each piece. Composites are the first layer the building blocks for the structures to come. A composite could include different types of blocks, rotated at a certain angle and compose a basic structure. An ideal composite could support other composites on top with out falling apart, but this is not required. A composite is measured using maginary bounding box, defined by calculating the lengths of all the pieces, finding the borders of said pieces and then finding the points that could define a box around all of them, then use this defined box to

calculate the height and width of the new composite and then calculate from the center point of said box the offsets from the center of the group to the center of each respective piece as shown on figure 3, this information is needed in order to add it as a new composite so it can be used to place the pieces in their respective new positions when the files needed for the simulation are being created. This is similar to creating a group of elements in a drawing application.

Good composites can be found using an evolutionary approach proposed earlier by Calle and Merelo, where basic structures are generated by searching a space of falling blocks and selecting those that remain standing. This algorithm is not capable of creating complex strucrees, but it can generate very novel basic structures. We can see that the evolution at this level can benefit from having more freedom in how basic structures are built. We do not want to form piles or sequences of blocks, this will be done at a higher level in the composite hirarchy.

At this level the composite generator has the following steps:

- 1) Generate the first generation of the population, consisting of randomly generated list of blocks.
- 2) Run a Simulation.
- 3) Evaluate each individual of the population.
- 4) If a member of the population has remaining pieces obtain the location and angle of the pieces, assign a fitness.
- 5) Do the selection, crossover and mutation operations.
- 6) Repeat the cycle.
- 7) From the best individuals select composites. Several algorithms could be proposed for this step, for instance selecting clusters of blocks.

Another option is to just manually generate interesting composites to be used by the algorithm. We could say that the previous approach is an option for a procedural generation of composites.



Fig. 1. List of pieces in the angry birds game.

The Angry Birds level generation track is a competition where the objective is to generate levels that are *fun* and *enjoyable* however they also have to be challenging but not impossible. However the rules for the competition specify that a configuration file must be used to generate the levels of the game in a limited amount of time, the information on the configuration file is first the number of *different* levels that must be generated, the second value is the different materials *not allowed* to be used, for instance a group of generated levels must not have circles which composition material is ice or stone, moreover it does not have to integrate a triangle piece with ice, stone or wood material so this combinations must

Using this same distribution a new evaluation will be added in which according to how much of the area is filled a score will be given and this score will be used as a new sorting value at the moment of selecting individuals for the crossover operation, since as shown on figure ?? there is a possibility that a huge mask is given to an individual it can be possible that the amount of pieces will not be able to cover the full area given, so in order to prevent this kind of problem we also proposed to give each mask a value that refers to the minimum amount of pieces and height required to fill the most of the mask, then check the amount of pieces and height before simulation for a single individual and then randomize which mask of the ones that can be used will be assigned to that individual, this way the results shown on figure 6 are obtained, this however gave us a new problem because as shown in the figure and on figure 5 as well, there were instances in which a certain combination of pieces could be too big for the area and this resulted in a

level that had structural flaws from the beginning which could provoke the entire structure and immediately nearby others to fall at the beginning of the simulation however this allowed us to realize that predefined composites could provide problems to the generated structures therefore this approach was not the best for this problem.

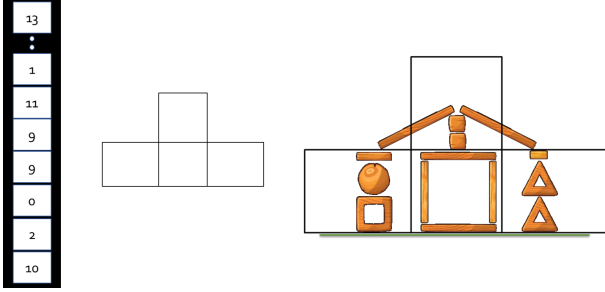


Fig. 5. Rule of thirds applied to an individual.

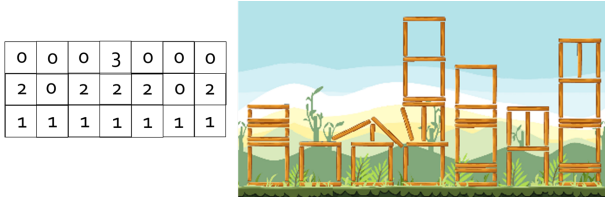


Fig. 6. Results obtained with rule of thirds.

### C. New approach

The first difference that the new approach has is that the masks that are assigned to each individual are no longer pre-generated ones, this is because we decided that pre-generating them would not allow the generator to *evolve* because the individuals would be limited to a certain number of possible combinations which will in turn prevent diversity in the population, so in order to prevent this the modification to this section allows the individuals to generate a mask in which a number of columns is defined by default but the amount of pieces in each one is random, however the total of assigned pieces in all columns must be the same as the total pieces allowed in each individual which is a number assigned before beginning the system, an example of this can be seen in figure 7 where the total of pieces in an individual is 18 and these pieces are distributed on all the columns available. This is further explained in this section.

The competition also requires that a generator follows a certain ruleset this is emulated by providing a file to the system to simulate the ruleset provided at the competition, this *text file* (*parameters.txt*) provides the specifications that must be met in the competition, the parameters in the file are as follows:

- Total required levels.
- Restrictions to pieces or piece-material combinations that cannot be placed or used in the levels, in this case if the file specifies that a *CIRCLE* made out of *ICE* cannot be

placed, so the system must be able to verify that said combination is not used.

- Total number of pigs, this value can be a single value or a range of values that represent how many pigs must be placed on a level.
- The amount of time in which the system must give an output.

			0			
4		1	2	6		2
1		5	8	9		3
0	0	4	3	2	4	5

Fig. 7. New chromosome chain.

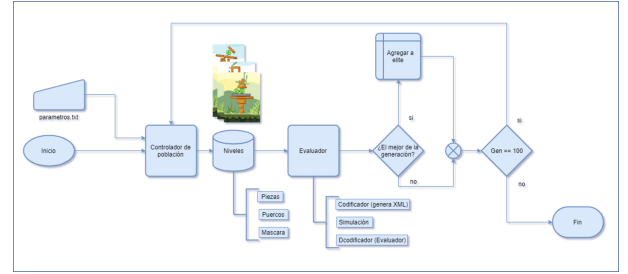


Fig. 8. New system diagram based on the parameter file and the new chromosome representation.

Using this as a base a new system diagram is proposed as shown in figure 8 in which the file with parameters is integrated with the input data the system requires, the way the individuals of the population are further presented and the way the simulation phase is taken, the method to represent the individuals is proposed by dividing the individual on three layers represented as follows:

- Element layer: also known as pieces layer, this layer represents the elements assigned to a particular individual by an array of numbers from 0 to infinite, where each number represents a reference to the class of a piece in the selection pool including the ones generated by the OOE algorithm, the representation of this is shown in figure 4.
- Mask layer: this layer is represented by a 7 digit array in which the value assigned in each column represents the amount of pieces from the element layer that will be assigned to said column, a representation of this is shown on figure 9.
- Enemy layer: also known as pig layer, this layer is represented by two elements a single numeric value representing the total amount of pigs assigned to the level and an array of locations for the placement of this same pigs, a representation of it is shown on figure 10.

In order to select the parents of the next generation the parents are ordered by their fitness, then the system can choose between 2 options for the selection, tournament selection or roulette selection:

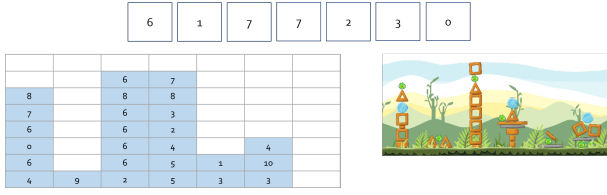


Fig. 9. Mask array with column and fenotype representation.

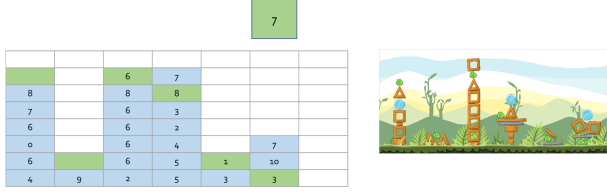


Fig. 10. Amount of pigs in a level, their location relative to the mask layer (green) and fenotype representation.

- Tournament selection: in order to select the parent by this method the system checks the amount of crossovers assigned, then calculates the ammount of parent required to compete in pairs, then by doing a descendent selection the slots for the tournament are filled and the winners are calculated in each group.
- Roulette selection: this method of selection was first decided to be used directly by giving a proportional amount of chances to be selected according to the fitness of a individual, however since one of the objectives is to obtain variety of levels the roulette must first be ranked in order to ensure that low fitness individuals can be selected as parents.

As for the crossover operations the system currently uses two types, single point crossover and double point crossover, however this crossover operation is not only applied to the piece array of an individual, but it is also applied to the mask of said individual, this way the mask can ensure that some pieces of some childs in the population are now placed in the level, this can bring the possibility that a mask that uses less pieces that it normally as can obtain a better fitness than one that uses all of them. An example of this two crossover types can be seen in figure 11 in which the mask of an individual is combined with another and the resulting mask for the childs have different amount of pieces to be placed.

The mutation operations of the generator work with the pieces layer of an individual, the four types of mutations used are:

- Add or remove elements from the piece list
- Change the piece that is being used in a point of the list
- Change the material a piece is made offsets
- Change the x and y coordinates of a piece

This mutations do not have to be used all at once on a single individual but depending on the probabilities they can happen all at once, since most of the pieces on a individual are mostly the same type of material then at least the material mutation can occur on all or most of the individuals to ensure that

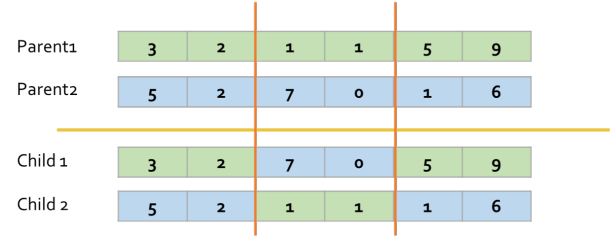
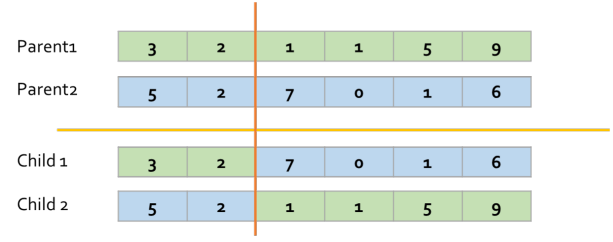


Fig. 11. Single point crossover(top) and double point crossover(bottom) applied to an individual mask.

all levels will not have all the pieces with the same material which could render a level "boring" by the fitness function.

Since this group of mutations further ensures diversity in the populations a different way to evaluate the individuals is needed, by using the previous proposed method as a base a way to evaluate the "diversity" and "uniqueness" of a individual, for this we first remove the  $x$  and  $y$  position as well as the  $angle$  errors in order to add two new evaluation formulas as follows:

- Hamming distance: a simple explanation for the *Hamming distance* is that it is a calculation applied to a pair of strings where the objective is to find the minimum required number of substitutions required in order to convert a string exactly the same as the other, its formula is shown in equation 1, since the formula itself locates the minimum number of substitutions then we compare the individual with the rest of the population trying to find the maximum number of substitutions required efectively using the formula to give a better fitness to those individuals that are different from the group.
- Shannon entropy: the shannon entropy is used on information theory to calculate the disorder or uncertainty, in this case it will be used to determine if a level is "boring" or "interesting" based on the amount of entropy calculated according to the pieces used in a level where a low entropy is boring because if a determined level as a lot of repeated pieces then it will look simple, the formula used is shown in equation 2 where the elements calculated are the pieces of a level and their appearance probability against all others.

$$d = \min \{ d(x, y) : x, y \in C, \text{if } x \neq y, 1 \text{ else } 0 \} \quad (1)$$

$$S = -\sum_i P_i \log P_i \quad (2)$$



Finally figure 12 and figure 13 represent the way all the layers of an individual are combined to create a level after all the selection, combination and mutation operations, first in figure 12 the composites layer is obtained from an individual, by a background process the width and height of all composites are obtained, then using the mask from the same individual the composites are placed in the order they are in their respective list (left to right in the level) in case a mask is not able to accommodate all the composites the remaining ones are not placed nor taken into account on the evaluation process, in the opposite case that there are not enough composites to fill a mask then the remaining areas are not taken into account as well, then in figure 13 the previous combination of layers is evaluated by checking which placed composites can hold a pig in the center or on the top of it, then the pigs are placed in the same order the composites are placed (left to right) in the locations where one is able to be placed.

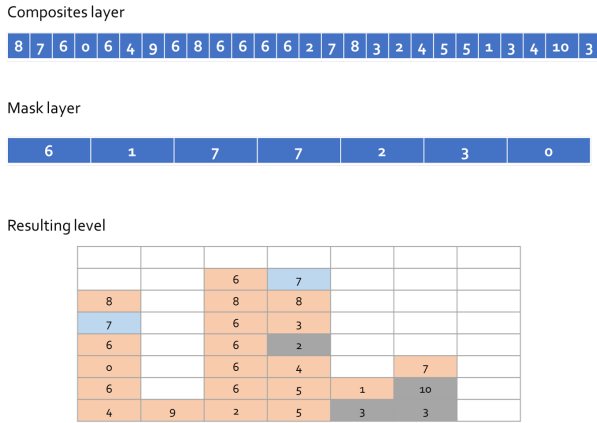


Fig. 12. Combination of composites layer and mask layer.

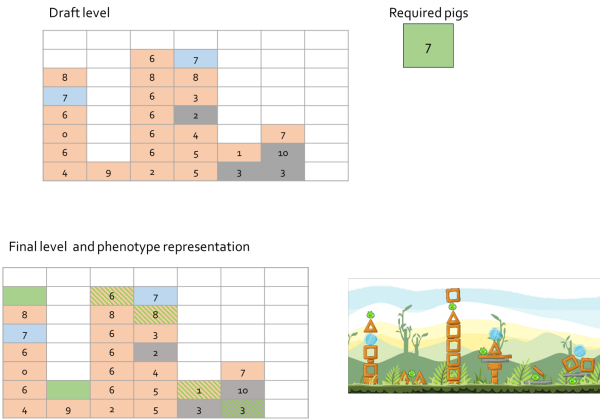


Fig. 13. Combination of figure 12 layers with pig layer.

Using all this configurations the system is capable of creating levels, some of them playable as shown in figure 14 however the levels that can be generated are far from optimized to be played on, this is an issue that must be changed in future instances but the generator itself is able to be used as a base

for a more complex one, and as shown in the future work section this issues can be solved by different means.

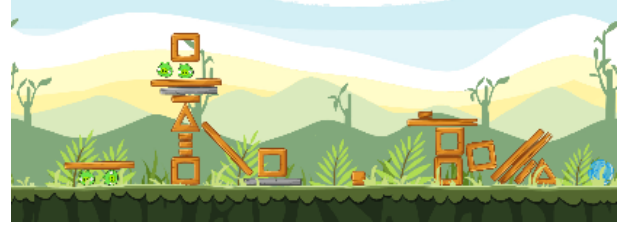


Fig. 14. Example of resulting level with current system.

#### IV. CONCLUSIONS AND FUTURE WORK

The system as been able to generate levels that fulfill the requirements provided by the parameters file, however a way to evaluate that the levels generated are indeed "fun" or "interesting" is still needed, at this point we have two was to evaluate the results:

- User survey, in case a survey is used as an evaluation tool then first the system has to generate the levels and a group of people must participate by playing the levels and answering a set of questions afterwards, the questions themselves must try to capture the feeling of a person towards the game like questioning if a level was too easy or too difficult to solve, if the generated structures look interesting or if the level required them to strategize how to use the provided birds to solve the level, the survey must be applied to a equal quantity of people of three different game skill levels, "non video game players", "casual video game players" and "avid video game players", this way we can evaluate a level or the entire system by knowing if all skill levels did not enjoyed a level because it was too simple or easy.
- By using a naive agent, if a naive agent is used to play the levels we can obtain data of how the agent solves the levels placed before them, and the way to evaluate if the level is or not "interesting" will be by checking how easily the aggent is able to solve a level, the point here is to have a level that cannot be solved by a single movement but to have one that a human player has to think a while to solve tryingto place the difficulty of a level just at the middle point between easy and hard difficulty levels.

Other aspects that the system need to improve is in the processing time required, since the current way to calculate the fitness and create the next generations of a population the system requires the system to execute simulations of the levels on every generation that passes therefore increasing the time it takes to complete an entire cycle of execution, so in order to change this we require to create a way to evaluate a level using only the information of the pieces, locations and pigs locations and using this data to predict the way a level will behave when placed in a real simulation environment therefore drastically reducing execution times.

Finally a different method to evaluate the individuals must be created, this is because the current evaluation method allows us to generate playable levels, this levels are far from ideal in the way that they only use a certain range of items present in the game therefore reducing the potential of a level, we expect to make the current fitness function work the most efficient that it can but another possible solution is found a good combination of formulas to generate the fitness function.

## REFERENCES

- [1] G. N. Yannakakis and J. Togelius, "Artificial Intelligence and Games (First Public Draft)," p. 359, 2017. [Online]. Available: <http://gameaibook.org/book.pdf>
- [2] —, "Experience-Driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, jul 2011. [Online]. Available: <http://ieeexplore.ieee.org/document/5740836/>
- [3] Rovio Entertainment Corporation, "Angry Birds," 2009. [Online]. Available: <https://www.angrybirds.com/>
- [4] M. Stephenson, J. Renz, L. Ferreira, and J. Togelius, "Competitions – IEEE Conference on Computational Intelligence and Games, 14–17 August 2018, Maastricht (The Netherlands)." [Online]. Available: <https://project.dke.maastrichtuniversity.nl/cig2018/competitions/#angrybirds>
- [5] M. J. B. Stephenson, J. Renz, X. Ge, L. N. Ferreira, J. Togelius, and P. Zhang, "The 2017 AIBIRDS Level Generation Competition," *IEEE Transactions on Games*, pp. 1–1, 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8410472/>
- [6] J. Renz, X. Ge, R. Verma, and P. Zhang, "Angry Birds as a Challenge for Artificial Intelligence," *Thirtieth AAAI Conference on Artificial Intelligence*, mar 2016. [Online]. Available: <https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewPaper/12527>
- [7] R. K. STANDISH, "OPEN-ENDED ARTIFICIAL EVOLUTION," *International Journal of Computational Intelligence and Applications*, vol. 03, no. 02, pp. 167–175, jun 2003. [Online]. Available: <http://www.worldscientific.com/doi/abs/10.1142/S1469026803000914>
- [8] T. Taylor, M. Bedau, A. Channon, D. Ackley, W. Banzhaf, G. Beslon, E. Dolson, T. Froese, S. Hickinbotham, T. Ikegami, B. McMullin, N. Packard, S. Rasmussen, N. Virgo, E. Agmon, E. Clark, S. McGregor, C. Ofria, G. Ropella, L. Spector, K. O. Stanley, A. Stanton, C. Timperley, A. Vostinar, and M. Wiser, "Open-Ended Evolution: Perspectives from the OEE Workshop in York," *Artificial Life*, vol. 22, no. 3, pp. 408–423, aug 2016. [Online]. Available: [http://www.mitpressjournals.org/doi/10.1162/ARTL\\_a\\_00210](http://www.mitpressjournals.org/doi/10.1162/ARTL_a_00210)
- [9] T. Taylor, "Evolutionary Innovations and Where to Find Them: Routes to Open-Ended Evolution in Natural and Artificial Systems," *Artificial Life*, vol. 25, no. 2, jun 2018.
- [10] J. Renz, X. Ge, S. Gould, and P. Zhang, "The angry birds ai competition," *AI Magazine*, vol. 36, pp. 85–87, 2015.
- [11] M. Kaidan, C. Y. Chu, T. Harada, and R. Thawonmas, "Procedural generation of angry birds levels that adapt to the player's skills using genetic algorithm," in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE, oct 2015, pp. 535–536. [Online]. Available: <http://ieeexplore.ieee.org/document/7398674/>
- [12] Y. Jiang, T. Harada, and R. Thawonmas, *Procedural generation of angry birds fun levels using pattern-struct and preset-model*. IEEE Conference on Computational Intelligence and Games, 2017. [Online]. Available: <https://aibirds.org/Level-Generation/cig16-competition-rules.pdf>
- [13] J. Togelius, N. Shaker, and M. J. Nelson, "Representations for search-based methods," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2016, pp. 159–179.