Generating Angry Birds-Like Levels With Domino Effects Using Constrained Novelty Search

Febri Abdullah*, Pujana Paliyawan*†, Ruck Thawonmas*, Fitra A. Bachtiar‡
*Intelligent Computer Entertainment Laboratory, Ritsumeikan University, Japan
†Research Organization of Science and Technology, Ritsumeikan University, Japan
‡Faculty of Computer Science, Brawijaya University, Indonesia
ruck@is.ritsumei.ac.jp

***Abstract*—This paper proposes a method to generate interesting Angry Birds-like game levels featuring the Rube Goldberg machine (RGM) mechanism using Constrained Novelty Search (CNS). An RGM level in Angry Birds emphasizes a domino effect, which allows it to be completed by only one bird shooting. By evolving the feasible population and infeasible population in CNS, our results show that the entropy of block-type frequencies is higher than the entropy by our previous generator and that two requirements to achieve playable RGM levels – the 100% stability and the perfect-shot rate – are met. The results indicate that the proposed method can generate levels with more diversity than our previous generator while maintaining their playability.**

***Index Terms*—Angry Birds, Constrained Novelty Search, Procedural Content Generation, Rube Goldberg Machine**

I. Introduction

The goal of procedural content generation (PCG) is to provide in-game contents (e.g., game levels, textures, or audios) that are not only playable but also different from each other. Key factors of PCG are the quality and the diversity of generated contents [1]. Our previous study [2] presented a PCG approach to automatically generate Rube Goldberg machine (RGM) [3] levels, where there are several constraints introduced to ensure that objects are arranged in a way that can create a domino effect among them. Those constraints were tackled by employing pre-designed sets of objects called segments. However, due to relying on such segments, the generator tends to generate levels with low diversity.

This paper proposes a search-based approach using Constrained Novelty Search (CNS) [4] to generate RGM levels. We hypothesize that the proposed method using CNS can generate RGM levels that are playable with higher diversity than levels generated by our previous generator. This hypothesis is examined by an experiment comparing 100 levels from each generator.

II. Related Work

A Rube Goldberg machine (RGM) is a machine built from smaller parts or devices connected in an overly complex way to achieve a simple goal. In our previous studies [2], [5], we implemented the generator on Science Birds. Science Birds is a clone version of the Angry Birds video game for science and research purposes, developed by Ferreira et al. [6]. The game has been the platform for the annual Science Birds Level Generation Competition [7].

In both of our previous studies, a rule-based approach was adopted to generate Angry Birds-like levels featuring RGM mechanisms, where preliminary results suggested that a structure collapsing motion produced by a domino effect in an RGM level could improve the spectator's mood [5]. A structure is a set of stacked game objects that resembles a building and constitutes a part of an entire Angry Birds-like level. However, the method presented therein relies on pre-designed segments, which leads to generation of levels with low diversity.

CNS rewards the diversity of its solutions while maintaining their quality and aims to produce a novel individual that satisfies given constraints by combining the Feasible-Infeasible Two-Population of Genetic Algorithm (FI-2Pop) and novelty search. A previous study conducted by Liapis et al. [8] focused on utilizing novelty search to generate game contents. FI-2Pop evolves feasible and infeasible populations separately; the feasible population evolves individuals to maximize their novelty, and the infeasible one evolves individuals to minimize their distance from the feasibility.

In the aforementioned work by Liapis et al. two approaches of the FI-2Pop were proposed: Feasible-Infeasible Novelty Search (FINS) and Feasible-Infeasible Dual Novelty Search (FI2NS). FINS employs a novel archive in the feasible population. In addition to employing a novel archive in the feasible population, FI2NS employs another novel archive in the infeasible population. Their results suggested that FI2NS outperforms FINS in terms of diversity, while FINS outperforms FI2NS in terms of the speed of reaching a solution. Their study utilized CNS to generate diverse levels in a strategy game. However, the capability of CNS to generate physics-based levels has not been evaluated.

III. Methodology

A. Level Generation

An entire RGM level consists of several structures connected to each other. These structures are generated by CNS and connected accordingly. To achieve a desired domino effect, each structure in an entire RGM level must satisfy several rules as follows:

- The structure is stable.

978-1-7281-4533-4/20/$31.00 ©2020 IEEE

- All available pigs in the structure must be destroyed when the structure is collapsing.
- The structure has at least one object flying outside the structure's boundary when it is collapsing.

In addition, as done in our previous work [2], we employ several variables for structures as follows:

- Structure boundary: the width and height of the structure in which all of its objects must reside.
- Input object: the object that needs to be destroyed to trigger a structure collapse. In this study, it can be a TNT (explosive) object or a b3 object, shown in Fig. 1.
- Output object: the object that flies beyond the structure boundary in a moment after the structure is collapsed.
- Input direction: the direction of an incoming object that will trigger the structure-collapse. There are four types of direction: "left", "right", "down", and "up".
- Output direction: the flying direction of the output object. There are four types of direction similarly to the input direction.
- Output position: the x and y coordinates representing the new position of the output object one time-frame after it flies beyond the structure boundary.

Algorithm 1 Level Generation

```
1: for all direction do
2:     structures.Append(CNS(direction, nBest))
3: end for
4: for i = 1 : lvCount do
5:     levels.Append(Selection(structures, strCount))
6: end for
7: Output(levels)
```

Algorithm 1 shows the process to generate multiple levels defined by $lvCount$. By utilizing CNS (line 2), the algorithm generates $nBest$ structures with the input direction for each direction type (i.e., "left", "right", "down", and "up"). This process is to ensure the availability of necessary structures when connecting them to form an entire RGM level. By the end of the first for-loop (line 3), a total of 4 $nBest$ structures are collected. The "Selection" function (line 5) selects $strCount$ structures according to their probability that is updated at each iteration and connects selected structures to form an entire RGM level.

B. Structure Generation

This section explains the structure generation process using CNS (cf., line 2 of Algorithm 1). In CNS, an individual is represented by a two-dimensional array of integer-type identifiers corresponding to every object in Science Birds, excluding platforms and birds. A platform is a type of object to which gravity is not applicable and cannot be destroyed. Platforms can be used to support any object placed above the ground of a level.

Figure 1 shows the objects with their corresponding identifiers. These identifiers are treated as genes and decoded into a structure. During decoding, genes in a column represent objects with the same type that are stacked starting from the top row regardless of their material. Objects are arranged in each row by a simple mechanism, not shown here due to space limit, in such a way that only the minimum number of them are placed that can support the objects in the above row, if any.

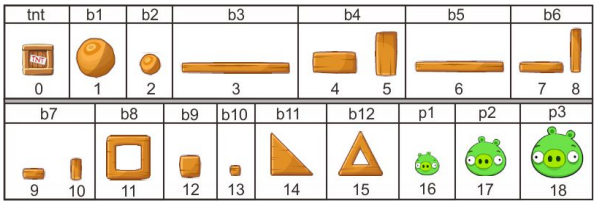


Figure 1: Objects with their corresponding name on the top and identifier on the bottom. Note that same objects but with different rotation have different identifiers.

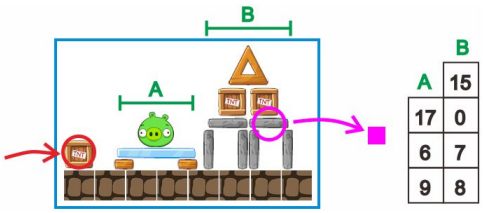


Figure 2: A structure decoded from a set of identifiers on the right side. It has the "left" input direction and the "right" output direction shown by a red and a magenta arrow, respectively. The input and the output objects are shown by a red and a magenta circle, respectively. The structure boundary is shown by a blue rectangle.

After the decoding process is done, an input object is placed according to the input direction. The input object for an incoming object from the "down" direction, coming from below, is a b3 object while it is a TNT for an incoming object from the other directions. Finally, a number of platforms are determined to support all generated objects in a structure. Figure 2 shows a structure decoded from an individual.

We rely on a simulation process to evaluate each individual. Due to the use of simulation, which in some cases could take a large amount of time, FINS is preferred. As mentioned earlier, Liapis et al. [8] suggested that FINS outperforms FI2NS in terms of the solution speed. Some of the aforementioned variables, i.e., the output object, output direction, and output position, are determined at the end of the simulation when there is no moving block detected.

Following a recipe in Liapis et al. [8], a feasible individual is scored using the novelty score (N_f) defined as

$$N_f(a) = \frac{1}{k}\sum_{i=1}^{k} d(a; b_i); \quad (1)$$

where b_i is individual a's ith-nearest-neighbor among individuals in the feasible population and the novel archive, and $d(a; b)$ is a distance function measuring the difference between

individuals a and b. Note that the evolution goal for the feasible population is to maximize the novelty score.

Since the output of our proposed method is a game level, diversity among individuals should be visually observable. Thereby, the visual distance d_v [8] is applied here:

$$d_v(i;j) = \frac{1}{hw} \sum_{y=1}^{h} \sum_{x=1}^{w} D_{x;y}(i;j); \tag{2}$$

where, because the size of every object in Science Birds are significantly different and one gene can lead to multiple objects of the same type, we compute the distance as the accumulated pixel-wise difference between the two structures; h and w are the longest height and width of both structures measured in pixel units, respectively; for the sake of computation efficiency, if pixel $(x;y)$ in structures i and j belong to the same object although they might render different parts of the object, $D_{x;y}(i;j) = 0$; otherwise, 1. In addition, pixels that do not belong to any object are labeled as 'empty' and compared using the same method.

To assess whether individuals are feasible or not, we propose a fitness function to calculate their fitness score (F_i) as follows:

$$F_i(c) = \frac{1}{3}F_1 + \frac{1}{3}F_2 + \frac{1}{3}F_3 \tag{3}$$

Note that individuals with the fitness score of 1 are feasible and the rest are infeasible. At the end of each generation, individuals in both populations are assessed and moved between the two populations according to their fitness score. In addition, the evolution goal for the infeasible population is to maximize the fitness score.

In the above equation, F_1 is the function representing the stability of a structure of interest and is calculated by considering the number of moving blocks, hence being destructed, in the structure during the first 10 seconds of the simulation when there is no bird-shooting occurred. The function is described as follows:

$$F_1 = 1 - \frac{B_s - B_e}{B_s}; \tag{4}$$

where B_s and B_e are the number of blocks at the beginning and the end of the simulation, respectively.

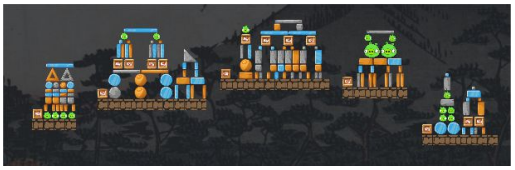
Next, F_2 is the function that calculates the number of destroyed pigs in a structure of interest. It represents whether the structure can be completed and, hence, a level containing it is playable. The function is described as follows:

$$F_2 = \frac{P_s - P_e}{P_s}; \tag{5}$$

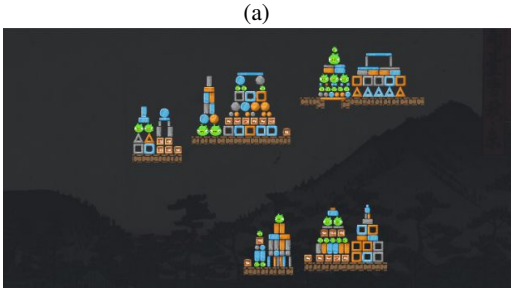
where P_s and P_e are the number of pigs that reside on the structure at the beginning and the end of the simulation, respectively.

F_3 is the function that constrains a structure of interest to have at least one object flying outside its boundary. The function is described as follows:

$$F_3 = \begin{cases} 1; & \#\text{flying\_object} \geq 1 \\ 0; & \text{otherwise} \end{cases} \tag{6}$$



(a)



(b)

Figure 3: Two levels generated by our proposed method.

C. Structure Selection

This section explains how structures are connected each other to form an entire RGM level. This corresponds to the "Selection" function in line 5 of Algorithm 1. Structures are selected from a list of structures generated by CNS.

In Science Birds, structures are supposed to be placed on the right side of the slingshot. This specification requires the player to shoot a bird to the right direction of the slingshot. Because of this specification, we only select the first structure from candidate structures with the input direction of "left". The next structure is then selected according to the current structure's output direction; more specifically, it is selected from candidate structures with the input direction opposite to the current structure's output direction. The probability of structure s being picked from candidate structures, $p(s)$, is defined as follows:

$$p(s) = \frac{1}{n-1}\left(1 - \frac{v_s}{\sum_{i=1}^{n} v_i}\right); \tag{7}$$

where n is the number of candidate structures, v_a is initialized to 1 and incremented each time s is selected to form an entire RGM level. Structure selection continues until the desired number of structures defined by $strCount$ (cf. line 5 of Algorithm 1) is reached.

IV. Experiment and Results

We used *stability*, *perfect-shot rate*, and *block frequency* from existing expressivity metrics [2], [9] to evaluate generated levels. The first two metrics evaluate the playability of generated levels. For RGM levels, playable levels are those that are stable and can be completed by one bird shooting (in other words, all the pigs can be destroyed in one shot). On the other hand, the last metric is used to represent the diversity of generated levels, and, in particular, the entropy of block frequencies is evaluated to represent the usage distribution of blocks in a generated level.