24th International Congress on Modelling and Simulation, Sydney, NSW, Australia, 5 to 10 December 2021
mssanz.org.au/modsim2021

An evolutionary approach to balancing and disrupting real-time strategy games

Jacob Snell [a], Martin Masek [a] and Chiou Peng Lam [a]

[a] School of Science, Edith Cowan University, Joondalup, Western Australia
Email: j.snell@ecu.edu.au

Abstract: In most computer games, the level of challenge experienced by a player is dependent on a range of variable factors defined within the game environment. When the end goal of such games is entertainment, the variables are carefully tuned by the designers to achieve a sense of fair, balanced gameplay. For military force design and wargaming applications, where the purpose is to explore elements of a real scenario, the question turns to how the variables can be exploited so as to provide the maximum advantage, and to disrupt the balance in favour of a particular side. In this paper, an automated approach to explore the impact of game variables on game balance is presented and evaluated. Based on an evolutionary algorithm, the approach explores a user-defined set of variables in order to determine optimal combinations of variable values to achieve a defined level of game balance or game disruption. The approach also provides the ability of biasing the search towards a set of user-defined values of the game variables, providing insight into how the most disruption can be achieved with the least amount of deviation from an existing strategy.

Two scenarios were developed in a Real-Time Strategy game environment with a focus on verifying the developed approach. Both scenarios were adversarial, with two opposing teams, Red and Blue, with the goal of each team to eliminate the opposition. The level of balance/disruption for a particular set of Blue Team variables was measured as a function of the difference between a target blue win rate and the actual win rate, with a tuneable bias to favour solutions where the solution deviated the least from a 'fair' solution (where the Blue Team had the same strength as the Red Team).

The first scenario was designed so that both teams were evenly matched in terms of number of units. The scenario was used to explore how the game balance could be disrupted by manipulating variables associated with the Blue Team units. The second scenario was developed so that one team was given significantly more fighting units. This was to test if the approach was able to achieve a particular desired level of balance or disruption despite the starting balance skew.

A series of experiments were performed using the developed scenarios to evolve a set of game variables tied to the Blue Team to achieve a range of balance levels while the red team's variables were kept static. The experiments show that it is possible to use the developed approach to balance or disrupt the variables of a game so that the desired level of game balance is achieved. A separate series of experiments also showed that the evolution process could be biased to find game variables that required the smallest amount of change. This is particularly important for balancing video games where designers often prefer only to make slight changes to the variables of a game even when desiring a large difference in a game's balance.

Keywords: *Real Time Strategy game, game balancing, game disruption, evolutionary computing*

1. INTRODUCTION

Significant effort is required to optimise force design. In Real-Time Strategy (RTS) games, multiple sides compete against each other where each side may have its own unique set of combat unit types, each with unique abilities. The challenge for game designers is to give each side a set of units that are different, whilst maintaining balance and keeping the game fair. Conversely, when a game or simulation is used as a platform for determining force design for a war gaming scenario, it becomes important to explore the characteristics and abilities of combat units that are required to maximise the chance of a win in that scenario. Whether the goal is to balance an RTS game for entertainment, or to disrupt balance in our favour for a wargame scenario, there are a large number of variables involved, how the variables interact, and the full impact of a change also depends on the participants strategy during the game. Evolutionary algorithms may be useful in such a domain, as they have the ability to explore a high dimensional search space and are able to discover optima in search spaces that may have a noisy or discontinuous fitness landscape.

Past research in automated game balancing has tended to focus on using Artificial Intelligence (AI) techniques for establishing a 'fair' game, with some of that work using evolutionary algorithms. Some, for example Olesen et al. (2008), focus on evolving an opponent with the aim of balancing game challenge. Others, such as Wheat et al. (2015), use evolution to determine an appropriate level layout for a particular level of challenge. Others still, for example Weber et al. (2020), evolve the level of access/exposure to in-game resources to control the level of challenge. Typically, work in the area is focused on balancing the game to match the players skill. This is in contrast to our work which seeks to explore how game balance can be disrupted by giving one team an unfair advantage. Previous work has found that such unbalanced games can pose challenges to AI algorithms that learn from self-play, where it becomes difficult for the disadvantaged player to discover a means of progressing towards a viable solution (Moy et al. 2019). Thus, knowing the extent of the imbalance and the means to counter it may be useful.

The work presented in this paper is focused on exploring a game's variables so as to achieve a particular level of balance or disruption. We present a generic evolutionary process that can tune any number of game variables for any game to obtain the desired level of balance or disruption. To demonstrate the approach, experiments are presented using a simplified RTS game, microRTS. In the experiments we measure game balance for a particular variable configuration as a weighted function of the average win rate, calculated from a set of simulated game scenarios using AI players and distance to a fair, balanced solution.

2. AN EVOLUTIONARY APPROACH FOR GAME BALANCING AND DISRUPTION

Our approach is based on an evolutionary algorithm type known as a genetic algorithm (GA), first proposed By Holland (1992). A GA searches a multi-dimensional space of solutions by taking a finite subset of possible solutions, called a population, and using concepts based on the theory evolution to produce subsequent generations of the population. The aim is to change the population over a set of generations to one that contains individuals that are 'better' solutions than those found in the previous generations. Two operators, mutation and crossover, are typically used in evolving individual solutions from one generation to the next. Mutation involves making an individual different from what it currently is and promotes exploration of the search space. Crossover involves taking two solutions and exchanging some of their components, allowing existing 'partially good' solutions to be exploited.

An overview of the developed approach is provided in Figure 1. There are three inputs into the approach: 1) the particular game to balance, 2) the set of game variables that will be explored, and 3) the desired level of balance/disruption. The goal of the evolutionary process is to find the optimal combination of game variables for the game based on the specified measures of disruption.

The process starts by generating an initial population of random individuals, encoding the game variables to be tuned, then evaluating these combinations through the Evaluator (the game in question) and producing a new population based on selection, crossover and mutation operators. This process is repeated, where each subsequent population is evaluated, until the stopping criteria is satisfied. The next section provides further details of the described approach.

2.1. Details of the Approach

In any evolutionary algorithm-based approach, there is a standard set of decisions that need to be made in terms of its configuration. Firstly, how can a particular 'solution' be encoded as an individual 'chromosome' that can be evolved? Secondly, what is an appropriate measure of fitness that can be used to differentiate between different individuals? Thirdly, how will the evolutionary operators, such as selection, mutation, recombination

etc., be designed? We now provide detail of the choices that have been made for our RTS based demonstration problem.

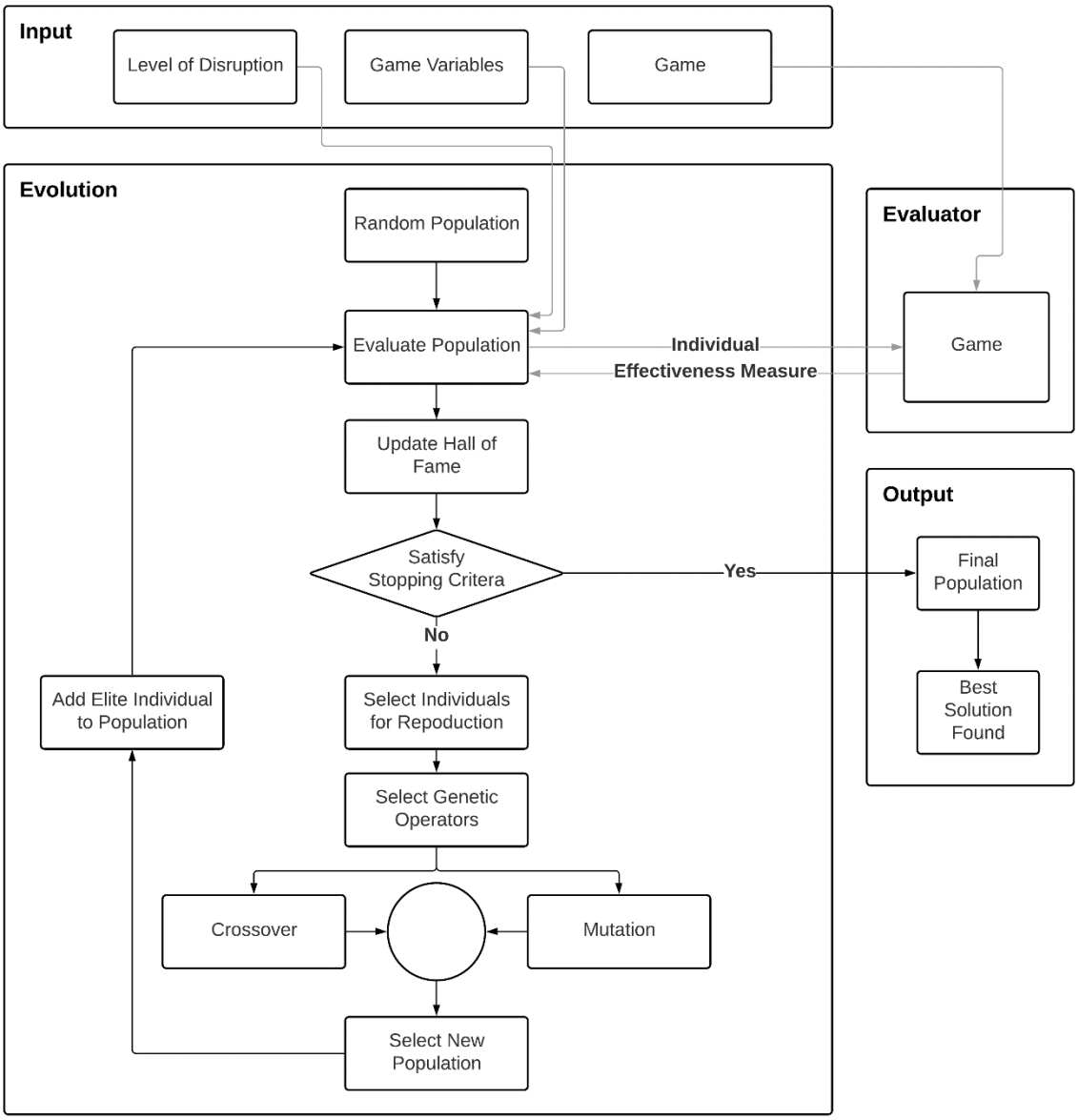


Figure 1. Overview of approach.

Chromosome Encoding

The representation of individuals is always problem-specific, but for a GA is usually composed of a list of numbers of a particular type (e.g., binary, integer, real etc.). Focusing the evolution on game variables naturally lends itself to a representation where each individual chromosome is a set of values, one for each variable.

Table 1. Game variables used, their definition, setting for the static Red Team and range for the evolvable Blue Team (units of measurement are game specific).

Variable	***Description***	***Red Team setting***	***Blue Team range***
Hit Points	Number of hit points a unit starts with.	20	[0,40]
Damage	Maximum damage a unit can inflict per attack.	10	[0,20]
Attack Range	Range at which a unit can attack other units.	2	[0,4]
Move Time	Time it takes for a unit to move one square on the grid.	8	[0,16]
Attack Time	How long it takes for an attack to complete.	6	[0,12]

In the MicroRTS game, variables that represent combat unit abilities are provided as integers. We use five of these variables, described in Table 1, producing an integer-coded chromosome of length of 5 to represent the blue team capability. For every experiment we keep the Red Team capabilities (values of their variables) fixed, and evolve the Blue Team capabilities based on our fitness function, whilst also constraining them within a set of limits. The fixed Red Team variable settings and Blue Team variable range are also shown in Table 1. The limits on these game variables for the Blue team were imposed so as to allow Blue Units to have at maximum (+/-)100% difference to the Red Team Units for each game variable. Together, the set of Red Team and Blue Team variables can be used to instantiate a game with a particular balance, either fair, or in favour of one of the teams.

Fitness Function

The fitness of a solution is used to differentiate between different solutions when it comes time to selecting candidates for the next generation. Thus, if we are seeking to balance/disrupt a game to a certain target extent, the fitness function needs to measure how close a solution is to that target. Here we take a simulation-based evaluation approach, running each solution a large number of times, where a simple reactive AI, with identical logic for both Blue and Red teams is used to control the units. Based on the statistical properties of these runs we calculate a measure of fitness. Simulation is a common approach to evaluating fitness in complex scenarios. An alternative is to analytically calculate balance based on the variable values themselves, but this is problematic because of the complex relationship that can exist between the variables. Measuring balance based on human-player participation in games is also not feasible, due to the large number of individual games that need evaluation and the bias that would be caused by individual players of different abilities.

The fitness of a solution is calculated using Equation (1), as the weighted sum of two components, *WinRateDifference* and *ChangedDifference*. The user defined input into the fitness function is a *TargetWinRate* for the blue team (as a percentage of games played) along with settings for the scalar constants: *WinWeighting* and *ChangedWeighting*, such that the sum of these weightings is equal to 1. The *WinWeighting* controls how much the fitness function should reward a solution for a win-rate that is close to the *TargetWinRate*. The *ChangedWeighting* instead leads to more reward for solutions where the game variables have changed the least.

Referring back to the diagram of the approach in Figure 1, the user defined input corresponds to the 'Level of Disruption' shown in the input layer.

$$Fitness = (\, WinRateDifference * WinWeighting \,) + (\, ChangedDifference * ChangedWeighting \,) \quad (1)$$

The *WinRateDifference* component, given as Equation (2), is a measure of how closely the actual *WinRate* matches the *TargetWinRate*.

$$WinRateDifference = \frac{100 - |TargetWinRate - WinRate|}{100} \quad (2)$$

The *WinRate* is calculated using Equation (3), using the number of wins and draws for the Blue Team over the set of AI-based game simulations. A Blue win is defined as all Red Team units being destroyed, and a draw is when both teams have units remaining after a fixed number of timesteps have elapsed.

$$WinRate = \frac{\left(BlueWins + \left(Draws * \frac{1}{2} \right) \right)}{(TotalGamesPlayed)} * 100 \quad (3)$$

The *ChangedDifference* component, given as Equation (4), quantifies the closeness of the Blue Team variable values to those of the static Red Team. Here, *N* is the number of game variables (in our case 5) and *BlueVariable* and *RedVariable* is the set of variable values for the Blue Team and Red Team respectively.

$$ChangedDifference = \frac{N - \sum_{i=1}^{N} \left| 1 - \frac{BlueVariable_i}{RedVariable_i} \right|}{N} \quad (4)$$

Evolutionary Operators

Selection of individuals to move from one generation to the next was done using a combination of elitism (Baluja et al., 1995) and tournament selection (Miller et al., 1995). Elitism saw the individual with the highest fitness was copied into the next generation. To fill the remainder of the next generation, tournament selection, with a tournament size of 2 was used. In this scheme, two random individuals from the current population are

considered at a time, and the one with the highest fitness is selected for inclusion in the population of the next generation, subject to crossover and mutation.

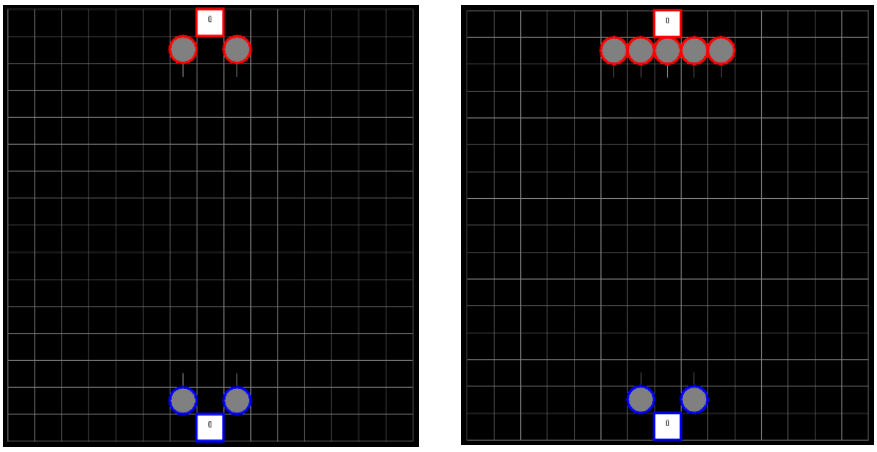
The mutation operator chosen was Gaussian mutation, with a mean of zero and a standard deviation of 0.2. The Gaussian was discretised to allow for integer mutation. Single-point crossover was selected as the genetic crossover operator. With single-point crossover, if it is determined that an individual should undergo crossover, another individual is chosen (also through tournament selection) and a random crossover point is chosen, on a gene boundary. Genetic material then exchanged between the individuals, producing two new individuals which are injected into the population for the next generation. These genetic operators could be applied without special modification due to the game variables being represented as integers in the chromosome.

3. SCENARIOS

The two scenarios that were developed, “even” and “unfair” are shown in Figure 2 (a) and (b), respectively. Both scenarios feature a Red Team base (square) at the top, protected by Red Team combat units (round) and a Blue Team base at the bottom, protected by Blue Team combat units. An irregular grid size of 15x16 was chosen to ensure unit placement could be symmetrical and ensure units meet precisely in the middle, given the same unit movement speed. The Red Team variable values are static, the Blue Team variable values are evolved (See Table 1 for settings).

The "even" scenario, was configured with both teams having an equal number of units symmetrically placed. If both the Blue and Red team units have the same values for their game variables, across many games the Blue team win-rate is close to 50%, and the average health difference is close to zero. Both teams have one base unit, which opposing teams must destroy, including any remaining units. The goal of this scenario is to explore the game variables in an otherwise balanced game.

The "unfair" scenario is configured in favour of the Red Team. Because of this imbalance, if both teams have the same game variable values the Blue team win-rate will be close to 0%. The layout and unit placement match the even scenario except that the Red Team's attacking force is increased from two units to five. This scenario was developed to challenge the automated approach and show its flexibility in a scenario where there is already some imbalance (disruption) in the game.



(a) (b)

Figure 2. The even scenario (a) and the unfair scenario (b).

4. EXPERIMENTS AND RESULTS

To demonstrate our approach, for each scenario we perform a set of experiments, each one with the aim of evolving game variable settings to achieve a specific target win-rate over 10,000 simulations with those variable settings. Target win-rates were set from 0% to 100% in increments of 10%. For all experiments, we set a *WinWeighting* to 0.75 and a *ChangedWeighting* to 0.25 (see Equation (1)) – ie. 75% of the fitness is based on how closely the target win rate is achieved and 25% based on how closely the variable settings for the Blue Team are to the Red Team settings.

For all experiments, the genetic algorithm parameters were identical. We used a population size of 30, and ran each experiment till 100 generations. We repeated each experiment 10 times to examine variability in the results. A mutation-rate of 0.1 (where each gene in an individual has a 10% chance of undergoing the mutation)

and a crossover-rate of 0.9 (each individual has 90% chance of undergoing crossover) was used for experiments, with the mutation and crossover rates determined through experimentation.

Table 2 summarises the experiment results for the even scenario, and similarly with Table 3 for the unfair scenario. Each row represents results from the best individual in the final population for that experiment. For each target win-rate, the percentage change of the Blue Team variables (from being identical to the Red Team) is shown, along with the win rate actually achieved by the solution. For both scenarios, the 10 repeated runs of each experiment produced a win-rate within 2% of within their target win-rate.

Table 2. Amount of change in individual variables to achieve a target win rate in the even scenario

Target Win-Rate	Hit Points	Damage	Attack Range	Move Time	Attack Time	**Win-Rate**
0%		-50%				0.8%
10%	13%	-40%				10.2%
20%	-5%	-20%				19.7%
30%	5%	-20%		-12.5%		30.1%
40%		-10%				38.0%
50%						**50.0%**
60%		10%				61.0%
70%	10%	10%				71.0%
80%	30%					79.8%
90%	15%	10%			-16.67%	90.0%
100%					-50%	99.5%

From these results we see that when the target win-rate is set to zero per cent, the evolutionary approach found that reducing the Blue units' damage by 50% (ie. From 10 to 5) was the smallest change necessary. The two Blue units can only beat the Red units 0.8% of the time in this case. Conversely, when the target win-rate is set to 100%, the 'optimal' change found to achieve this was to reduce the Blue units' attack time by 50% (ie. from 6 to 3), whilst keeping the other variables at the same setting as the Red Team. Reducing the attack time by 50% means that the Blue units are able to attack twice as often as the Red units, with an actual win rate of 99.5% achieved.

Due to the scenario design, when both teams have the same game variable values, the Blue team's win-rate is fifty per cent. The evolutionary process confirmed this as the optimal solution when the target win-rate was set to 50%. Overall, the approach successfully found small changes to the game variables to achieve the even scenario's target win rate.

Table 3. Amount of change in individual variables to achieve a target win rate in the unfair scenario.

Target Win-Rate	Hit Points	Damage	Attack Range	Move Time	Attack Time	**Win-Rate**
0%						0.0%
10%	-5%				-66.7%	11.9%
20%				12.50%	-66.7%	19.4%
30%	20%	30%		-12.5%	-50%	30.6%
40%		-10%		-12.5%	-66.7%	40.2%
50%	-20%	10%		25%	-66.7%	**49.3%**
60%	-50%				-83.3%	59.6%
70%	5%	-40%		12.5%	-83.3%	69.1%
80%		10%		-12.5%	-66.7%	78.7%
90%		30%		-12.5%	-66.7%	90.0%
100%				-12.5%	-100%	100.0%

The unfair scenario was designed to give the Red team an advantage by giving them 3 extra units to make the balancing task more challenging. As a result, the evolutionary process found that leaving the game variables unchanged resulted in a zero per cent win-rate for Blue. It also found that reducing attack time by 100% from 6 to 0 and the move time by 12.5% from 8 to 7 was the smallest change necessary to achieve a 100% win-rate. An attack time of zero means that the Blue units attack every timestep of the MicroRTS simulation when in range. Reducing the move time was necessary so that the Blue units could get into a position to attack before the Red team. This resulted in the Blue team's units being able to shoot while the Red team's units were still moving into position, increasing the Blue teams win-rate.

Counter to intuition, to meet the target win-rates of 50% and 60%; the evolutionary process found that the smallest change necessary included reducing the Blue units' hit points by 20% and 50%, respectively. If the hit points had not been reduced, the reduced attack time resulted in the Blue units achieving a much higher win-rate than the target. This strategy was also observed when the target win-rate was set 70%, with the evolutionary process opting to reduce the damage significantly instead.

5. DISCUSSION AND CONCLUSION

This paper has presented a game-independent automated approach to finding novel game variables combinations to achieve a required game balance, or, in other words, to introduce a quantifiable amount of disruption to the balance. The approach takes as input a game, set of game variables and a target disruption level. Through an incorporated genetic algorithm, the search space defined by the game variables is explored in order to optimise their values with regards to the desired level of disruption. Aspects of this disruption that were explored included win rate, and the desire to obtain the desired disruption level with minimal variable changes from what would constitute a balanced game. Demonstrating the approach using the MicroRTS game, the approach successfully found values for game variables such that the desired level of disruption or balance could be reached. An advantage of using this approach is that it is easily transferable to other games or simulation environments. Our approach is also suitable for complex wargaming scenarios as the evolution can search through a large number of game variables, which would be infeasible to balance manually. Additionally, the evolutionary search is able to significantly disrupt or balance a scenario and is not limited to incremental improvements.

ACKNOWLEDGEMENTS

This research was funded by the Defence Science Centre of Western Australia as part of the national Artificial Intelligence for Decision Making Initiative.

REFERENCES

Baluja, S., & Caruana, R., 1995. Removing the genetics from the standard genetic algorithm. In Machine Learning Proceedings 1995 (pp. 38-46). Morgan Kaufmann.

Holland, J. H., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

Miller, B. L., & Goldberg, D. E., 1995. Genetic algorithms, tournament selection, and the effects of noise. Complex systems, 9(3), 193-212.

Moy, G., & Shekh, S., 2019. The application of alphazero to wargaming. In Australasian Joint Conference on Artificial Intelligence (pp. 3-14). Springer, Cham.

Olesen, J. K., Yannakakis, G. N., & Hallam, J., 2008. Real-time challenge balance in an RTS game using rtNEAT. 2008 IEEE Symposium on Computational Intelligence and Games, CIG 2008, 87–94. https://doi.org/10.1109/CIG.2008.5035625

Weber, M., & Notargiacomo, P., 2020. Dynamic difficulty adjustment in digital games using genetic algorithms. Brazilian Symposium on Games and Digital Entertainment, SBGAMES, 2020-November, 62–70. https://doi.org/10.1109/SBGames51465.2020.00019

Wheat, D., Masek, M., Lam, P., Hingston, P., 2015. Dynamic Difficulty Adjustment in 2D Platformers through Agent-Based Procedural Level Generation. Proceedings of the 2015 IEEE International Conference on Systems, Man and Cybernetics (SMC 2015), 2778-2785, Piscataway, N.J., IEEE.