Applying a Micro Genetic Algorithm to Play a Tower Defense Game

Lucas Charles Croslyn
St. Francis Xavier University
Department of Computer Science
Antigonish, Nova Scotia, Canada
x2019dvh@stfx.ca

David Tyler Gosbee
St. Francis Xavier University
Department of Computer Science
Antigonish, Nova Scotia, Canada
x2019fep@stfx.ca

Fig. 1: Random initialization of towers on the map

Fig. 2: Evolved placement of towers on the map

***Abstract*—The motivation for this paper was to evaluate the performance of a genetic algorithm on the problem space of the tower defense video game "Bloons TD5". "Bloons TD5" features multiple towers and upgrade paths that can be utilized in an attempt to beat as many oncoming waves of enemies, where each successive wave features harder to defeat enemies. Genetic algorithms apply an approximation of evolution to solving problem spaces, and, through the use of successive mutations and breeding, create a population of possible solutions. The goal of this paper was to evaluate the ability to search a tower defense problem space using a small number of generations and a small population size.**

***Index Terms*—Evolutionary Computation; Genetic Algorithm.**

I. Introduction

A genetic algorithm is a way for a computer to approximate the process of evolution in order to search a problem space effectively without prior knowledge of the problem. The algorithm initializes a random set of solutions to a problem within that problem's search space. These initial solutions tend to not be very good solutions, though some are closer to optimal. The algorithm then repeatedly fills out new populations through selecting some of the better ones and performing mutations and crossovers to create new solutions which are possibly closer to the optimum. After the algorithm finishes, it may not end up with the most optimal solution, but it tends to get better answers than doing a random search. [2]

The problem that we decided to run the genetic algorithm on was a tower defense game. There have been some past papers with genetic algorithms and tower defense games. These papers however, deal more with neural network setups [4] or explore how well a genetic algorithm works in contrast with other algorithms [3]. This paper will delve specifically into the genetic algorithm approach. In a tower defense game, the objective is to place towers on a map which will automatically shoot at incoming enemies on a set path. The enemies will come in different waves which increase in difficulty. The player has a set pool of lives which will decrease if any enemies get to the end of the path. When the lives reach zero, that means the player has been defeated. The player also typically will have a bank of money which needs to be high enough to place down a new tower or to improve a previously placed tower.

The specific tower defense game that was chosen is called Bloons TD 5. In this game, there are multiple different types of towers that can be placed down to attack. Each of these tower types has two different upgrade paths with four upgrades in each which will improve that tower. However, only one of these paths can have more than two upgrades in that path. These towers and the upgrades for them are not all equal in strength and some are better than others. They also all have different cost values, typically with the more powerful towers and upgrades costing more.

TABLE I: Settings used in genetic algorithm

	Mut Chance	Cross Chance	Gen Count	Pop Size	Selection Count	Evaluation	Reinitialization
No reinit and wave	80	15	10	10	3	Death Wave	False
No reinit and towers	80	15	10	10	3	Death Wave - Tower Count	False
Reinit and wave	80	15	10	10	3	Death Wave	True
Reinit and towers	80	15	10	10	3	Death Wave - Tower Count	True

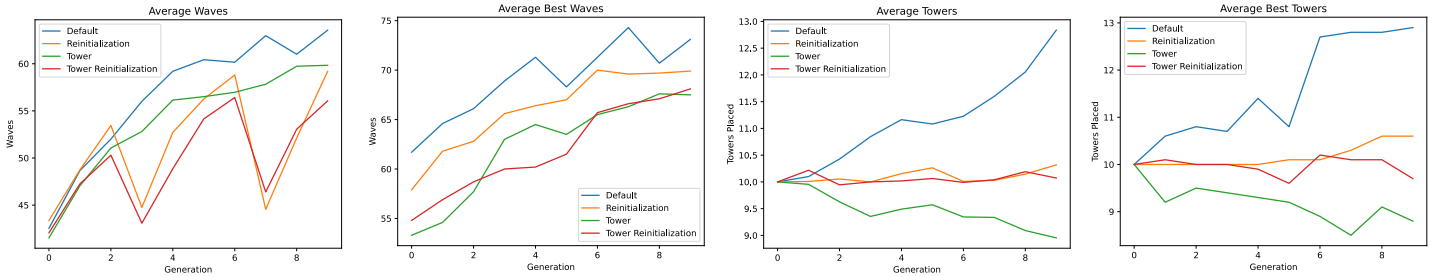

Fig. 3: Graphs of the average and best wave counts and number of towers over the generations for each group

In addition to different tower types, there are also different enemy types. These different types can take varying damage depending on the tower attacking them. There are also two enemy types that only certain towers, sometimes needing certain upgrades, are able to do any damage at all. We decided that in order to have a higher chance of getting towers which could damage these special enemies, whenever the algorithm decided on a new tower, it would first pick from three different categories. These categories were: all tower types, and then two other options for towers which could do damage to the two special types of enemies.

The game mode we used was not the classic traditional game mode. The game mode we chose was called “Deflation Mode”, the player starts off with a set amount of money, and can not gain any more during the run. Additionally, the game starts with wave 30, so enemies start off more difficult. We chose this mode since the algorithm could place all of the towers initially, instead of having to wait to gain more money. This also made it so the runs would finish faster since it starts off on a more difficult wave.

The map chosen was the first map of “Bloons TD5” known as “Monkey Lane” as seen in Figures 1 and 2. It was decided to split the map into a grid system with specific spots where the algorithm could place towers. These positions are sometimes in invalid locations so that the algorithm could still function on any map that was chosen.

II. Implementation

The implemented genetic algorithm in this paper would first initialize potential solutions, which would be arrays of potential towers that would be placed. Each solution would then be evaluated by placing the towers in the order of index and starting the game. Once the fitness for each solution was calculated, parents would be selected with a tournament method. Each parent could then potentially ‘evolve’ with the ability to use a variable number of different forms of mutation, including the ability to add or remove towers in different indexes of the array, change a tower to another, and change the location of a tower. The two parents then had the chance to have a crossover happen between them. The crossover utilized swapped a random tower from each parent to the other parent. The genetic algorithm featured two different evaluation methods, one based purely on the wave reached, and one taking into account both the wave reached and the number of towers utilized. This process of choosing parents was done until the new population was full. Finally, since the fitness computation time is large, there were fewer permutations in the population. Due to this, a micro-genetic algorithm approach was compared against. In a micro-genetic algorithm, every few generations the population will be reinitialized with new permutations except for the best permutation which will be kept [1]. Elitism was also utilized where the best solution would always be carried over between each generation.

III. Statistics

The analysis in this paper was done on a single problem space with 4 different settings. The specific settings can be seen in Table I. These specific settings were chosen based on the fact that there would be a small population and generation size. Having a high mutation rate, with the chance for more than one mutation to happen, will potentially increase the diversity quicker. We also wanted to make sure that since the population could change a lot, we would always keep the best solution, even with the re-initialization.

A. Comparisons

Comparisons were done 10 times, each with 4 different settings for 10 generations. The first settings utilized no population re-initialization, and only evaluated the runs based on the wave achieved. The second settings utilized no population re initialization and evaluated the run based on the wave achieved and the number of towers placed. The third settings utilized population re-initialization and only evaluated the runs based on the wave achieved. The last settings utilized

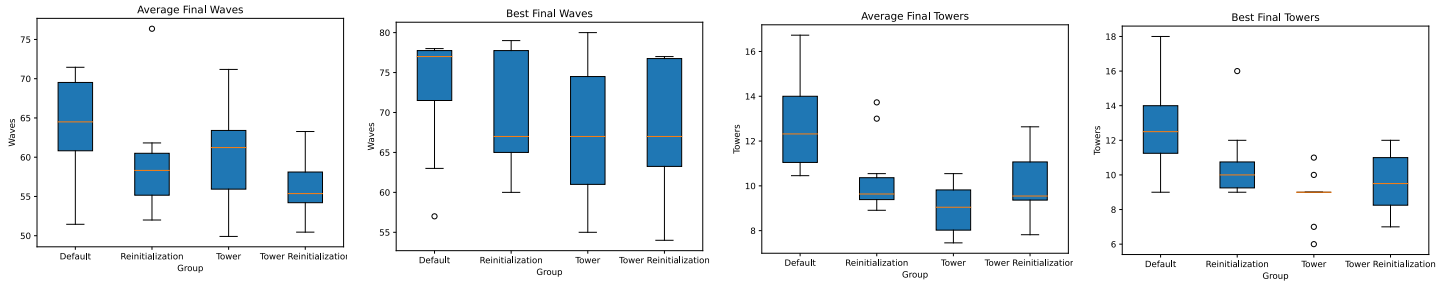

Fig. 4: Boxplots showing the average and best final wave and tower counts for each group of settings

TABLE II: P-Values for settings compared to the default wave fitness settings

	Average Waves	Best Waves	Average Towers	Best Towers
Re-initialization	0.1508	0.6188	0.0072	0.0267
Tower	0.1859	0.2369	0.0003	0.0008

TABLE III: P-Values for settings compared to the tower fitness settings

	Average Waves	Best Waves	Average Towers	Best Towers
Default	0.1859	0.2369	0.0003	0.0008
Tower Re-initialization	0.1405	0.9086	0.0885	0.3274

population re-initialization and evaluated the run based on the wave achieved and the number of towers placed. The values that were compared between these settings were the wave count and the number of towers.

IV. Results

A. Waves Achieved

As determined using the Mann-Whitney U test seen in Tables II and III which shows the comparisons between where the genetic algorithm finished, the differences between the average waves achieved and the average best waves achieved were not significantly different.

B. Towers

As determined using the Mann-Whitney U test again seen in Tables II and III which shows the comparisons between where the genetic algorithm finished, the differences between the average towers placed and the best runs towers placed were significantly different in all instances but tower fitness vs tower fitness with re-initialization. The initial towers placed by all the settings were 10, where mutations allowed for changing the number of towers. The mean towers placed with the default settings was 12.8, and the average best towers placed was 12.9. The mean towers placed for the re-initialization setting was 10.3, and the average best towers placed was 10.6. The mean towers placed for the tower fitness setting was 9, and the average best towers placed was 8.8. The mean towers placed for the tower fitness with re-initialization setting was 10.1, and the average best towers placed was 9.7.

C. Discussion

Based on the results found, we can determine that the tower fitness setting generated solutions that performed comparably to the default settings in terms of waves achieved, while also successfully reducing the number of towers placed by approximately 29.7 percent on average.

The solutions provided by the re-initialization settings did not prove to be helpful in terms of achieving statistically significant higher wave counts, though it did prove to be more efficient in the usage of towers compared to the default settings with a reduction of 21.1 percent for the tower fitness with re-initialization setting and 20.5 percent for the re-initialization setting.

V. Conclusions and Future Work

In conclusion, this paper found that the results of re-initialization on a micro-GA played a role in reducing the towers used compared to the default settings without sacrificing a significant amount of waves achieved. In the multi-objective fitness approach with a micro-GA, no significant difference was found between the default tower fitness setting and the tower fitness setting with re-initialization. Future work that can be done is to allow the algorithm to work on a main game mode. For this to function, towers would have to not all be placed at the start of the game, but throughout it. This would also mean that a single tower cannot be upgraded right when it is placed as well, which was currently being done. Money may have to be kept track of as well to know what can be afforded at any time. However, doing this method may be moving beyond a strict genetic algorithm approach to be effective.

REFERENCES

[1] Carlos A. Coello and Gregorio Toscano Pulido. Multiobjective optimization using a micro-genetic algorithm. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, page 274–282, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[2] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[3] Peng Huo, Simon CK Shiu, Haibo Wang, and Ben Niu. Application and comparison of particle swarm optimization and genetic algorithm in strategy defense game. In *2009 Fifth International Conference on Natural Computation*, volume 5, pages 387–392. IEEE, 2009.

[4] Tse Guan Tan, Yung Nan Yong, Kim On Chin, Jason Teo, and Rayner Alfred. Automated evaluation for ai controllers in tower defense game using genetic algorithm. In *International Multi-Conference on Artificial Intelligence Technology*, pages 135–146. Springer, 2013.