Evolutionary Approaches to Evolve AI Scripts for a RTS Game

Lucas Ferreira
Institute of Mathematics and Computer Science
University of São Paulo
Sao Carlos, Brazil
lucasnfe@icmc.usp.br

Rodrigo Pereira
Institute of Mathematics and Computer Science
University of São Paulo
Sao Carlos, Brazil
rodrigofp@grad.icmc.usp.br

Claudio Toledo
Institute of Mathematics and Computer Science
University of São Paulo
Sao Carlos, Brazil
claudio@icmc.usp.br

Leonardo Pereira
Institute of Mathematics and Computer Science
University of São Paulo
Sao Carlos, Brazil
leonardop@usp.br

ABSTRACT

This paper evaluates three different Evolutionary Algorithms (EAs) for generating non-player characters (NPCs) via Artificial Intelligence scripts for a Real Time Strategy (RTS) game. The first approach executes only a Genetic Algorithm (GA), while the second and third ones combine GA with a Dynamic Scripting approach. The Bos Wars game is used for testing these EAs, which are able to create and to evolve strategies of this RTS game coded as scripts in LUA language. In order to do this, the EA communicates with Bos Wars' engine by sending scripts, playing matches and capturing statistical data to evaluate its individuals.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *games*.

General Terms

Algorithms, Management, Measurement, Performance, Experimentation, Verification.

Keywords

Evolutionary algorithms, Artificial intelligence, games.

1. INTRODUCTION

The present paper applies Evolutionary Algorithm (EA) to generate and evolve NPCs for a Real Time Strategy (RTS) game called Bos Wars. Real time adaptation of the NPCs behavior can increase the level of entertainment [1] and the natural adaptability of the EAs can allow finding different and unpredictable strategies [2].

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2598486

EAs can be used by game developers as a tool for generating game IAs, forming a base several AI scripts with different difficult levels. It also can be used to dynamically evolve a script against human players, forcing them to improve their game abilities and making the game a better experience. The authors in [3] argue that playing against adapting AI scripts, produced by the application of EAs, can improve human players skills more than the games which are played against other humans.

The goal of this paper is to evaluate three EA approaches as methods to generate scripts that control the NPCs of the RTS game Bos Wars. The first EA proposed expands the work presented in [4], which introduced the first individual's representation, as well as genetic operators to dynamically construct and evolve script for this game.

The present paper introduces another representation of individuals as well as an uniform crossover and tailor-made mutation operators to deal with the proposed representation. The second EA is an adaptation of the method proposed in [1], where an EA is combined with Dynamic Scripting approach. The third one is an EA that evolves an initial population generated from the same knowledge base build for the second EA. The results found against Bos Wars' standard scripts are reported, as well as the performance of scripts evolved by the proposed EAs against each other.

2. EVOLUTIONARY ALGORITHM

Each individual represents a script or a game strategy, where each gene is a possible action. The genes are classified as army genes or building genes and they encode information about creating armies or constructing buildings, respectively.

In the *army* gene it is defined which unit must be created as well as its quantity and force. The parameter *force* is an integer value from 0 to 9 used by the decoding process to identify the type of army. The *building* gene has the type of structures that will be built, i.e., magma pumps, vaults, power plants and aircraft

factories, among others. Thus, the decoding process is responsible to transform an individual into a game script.

Each Bos Wars script is compounded by two set of instructions (actions): one set is executed just once and another which is executed repeatedly. The proposed representation handles these two sets, so each one is encoded separately and can have different sizes. There are two chromosomes for each individual. The first chromosome has genes with information about actions that will be in the beginning of the script, outside of the loop. The second chromosome has the actions that will be decoded inside the loop of the game script.

The information encoded by the proposed individual can be completely decoded into a game script. For instance, the gene G1 of a chromosome can encode the action "Build a vehicle factory", while another gene G2 can represent the action "Create 3 helicopters unities. Thus, all the genes from the two chromosomes of an individual are decoded into a game script in LUA language. The decoded script is executed by a NPC during a match.

2.1 Fitness Function

The fitness of an individual is calculated taking into account if a script won or lost the match. A higher value is assigned to a winner script that spent less time to win than other winner scripts. On the other hand, a loser script which played longer has greater fitness than scripts that quickly lost. If the time limit is reached, there is no winner in the match. Thus, the individual being evaluated is assumed to be a loser and the enemy is named the winner. The idea is to consider as a defeat for the EA a tie match when playing against other scripts.

To improve the initial scripts to be used by the EA, the method evolves a population of scripts previously selected by an off-line learning phase, or preprocessing phase, which is responsible to provide a better initial population to the on-line evolutionary process.

2.2 Genetic Operators

This paper used an uniform crossover for each chromosome of the individual. This means that only genes between the parents' chromosomes of actions inside the loop are exchanged to create the child's chromosome of actions inside the loop, and the same rule applies to the actions outside the loop.

In the uniform crossover, each gene of the produced individual has 50% of chance to be inherited from one of the parents. If the chromosome from one parent is larger than the same chromosome from the other parent, the procedure follows the larger chromosome until its end, with 50% of chance of each gene to be copied to the child.

Still on the EA routine, the child generated by the crossover has a 50% chance to have each of its chromosome mutated. In total, four types of mutation operators were proposed: Swap, Change, Removal and Addition.

3. COMPUTATIONAL RESULTS

The proposed methods are evaluated first taking into account their ability to overcome the standard AI of the game. To achieve this goal, each approach played the role of an NPC against three different native scripts from Bos Wars: 'Default', 'Blitz' and 'Tank Rush'. These scripts were chosen because they represent the harder AI to beat. The 'Default' script presents a balance between offensive and defensive actions. On the other hand, 'Blitz' and 'Tank Rush' are two offensive strategies, where the first uses both infantry and tank units to perform attacks, whereas the second applies mainly tanks to perform such task. The computational experiments was executed 10 times in order to evaluate the average performance of the proposed methods.

The GA was set with population size of 7 individuals and intermediate population with 4 individuals. The crossover was always applied (100% of crossover rate) and mutation rate was 50%. The stop criterion in the preprocessing evolution is the number of generation, which is limited to 8 generations. The stop criterion to define the Knowledge Base is to find 7 winner individuals against the enemy script evaluated or until 100 matches have been performed. In the tests against the Bos Wars native scripts, the evolution phase (online) was done until a winner individual has been found, or until a maximum of 200 matches has been played. In the tests where each algorithm confronted scripts generated from another algorithm, a total of 200 matches was executed. Figure 14 presents the amount of executions where the methods weren't able to beat the Bos Wars native scripts.

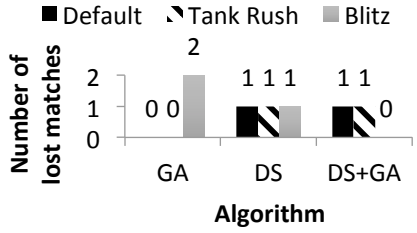

Figure 14: Executions without victory

The GA always generated a script able to win the Default and the Tank Rush scripts, and it produced a better script in 8 out of 10 executions against Blitz. DS and DS+AG were not able to generate a better script in only one execution against Default and Tank Rush. DS + AG beat Blitz script in all 10 executions performed.

These results indicate that all approaches are able to generate scripts able to win those available in Bos Wars. Thus, the methods can produce different script strategies, at least competitive against the native ones, which can be used as a start point for a game designer define a better and more interesting AI for the game.

4. REFERENCES

[1] PONSEN, M., SPRONCK, P., MUÑOZ-AVILA, H. AHA, D., 2007, *Knowledge Acquisition for Adaptive Game AI*. Science of Computer Programming, v.4, n.1, p. 59-75.

[2] LUCAS, S.M. AND KENDALL, G. 2006, *Evolutionary Computation and Games*. IEEE Computational Intelligence Magazine., February, p.10-18.

[3] SMITH, G., AVERY, P., HOUMANFAR, R., LOUIS, S., 2010.*Using Co-evolved RTS Opponents to Teach Spatial Tactics*. IEEE Conference on Computational Intelligence and Games (CIG'10) p. 146-153.

[4] TRAISH, J. AND TULIP, J.. 2012. Towards Adaptive Online RTS AI with NEAT. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2012.* Granada, Spain, September 11 - 14, 2012). IEEE, USA, 430-437.