ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

The Effectiveness of Genetic Algorithms In General Game Playing

Khan Faizan Ahmed Asad, Hararwala Huzaifa Quaid, Kshirsagar Saumitra Jitendra, Kulkarni Nikhil Vivek.

B.E. Students, Dept. of Computer Engineering, Sinhgad College of Engineering, Pune, India

ABSTRACT:Artificial intelligence is used in gaming industry to enhance game-play of the system.The proposed system is aimed at developing an agent to play classic console games like Mario bros, Pac man, etc. It was designed to start from scratch with very little domain specific knowledge about the game. It learnsto play the games, by hit-and-trials and evolutionary techniques. The agent uses genetic programming to select the best from a pool of generated neural nets, which finished game levels with better scores. The implemented agent successfully learned to play two types of video games without knowing their rules beforehand.

KEYWORDS: Artificial Intelligence (AI), General Game Players (GGP), Neural Networks (NN), genetic programming, Bizhawk emulator, evolutionary algorithms.

I. INTRODUCTION

Artificial intelligence has been used in the gaming industry for designing the behaviour of non-playable characters and enemy characters in games. But designing a self-playing intelligent agent that could learn on its own to play a game posed a few challenges.

Some basic challenges, like how to make software understand the rules of game, how the software understands and interacts with the game world and most importantly how the software actually makes decisions to play the game, triggered the idea of evolving such an agent.

An important motivating factor was to develop an optimal problem solving capacity of intelligent agents in the game world, and use them effectively in real life problems, like geographical navigation, maximizing profit in short runs, terrain mapping, etc..

II. RELATED WORK

There have been multiple attempts at this challenge using different methods.

1. Agents for games like chess, backgammon, poker etc. have been successfully developed. One such noteworthy implementation is the Deep Blue by IBM. Most of such implementations concerning board-games like chess use methods of informed search methods (like mini-max algorithm or more advanced search methods like Monte Carlo Tree Search). This class of agents generally requires the entirety of game rules to be specified before playing.

2. The above implementations are not capable of playing any general game, because they are generally written to tackle specific gameplay and techniques used in one game do not transfer as they are to other games.
   The GGP implementations, on the contrary, are bots which take input as rules of games by common interface GDL (Game description Language) and play any game using the simple tree-search methods or advanced learning algorithms. The games used in the GGP competition are usually variants of existing board games, or otherwise turn-based discrete games. Game Description Language (GDL) is

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

used to describe the rules of the class of games playable within the GGP framework, i.e. Finite, discrete, deterministic multi-player games of complete information. GDL describes games in a variant of Datalog. Game states are defined in terms of facts and algorithms for computing legal moves, subsequent game states, termination conditions and terminal scores for players are represented as logical rules. Notable implementations include - Cluneplayer, developed by James Clune. It uses DFS. Flux player was developed by S. Schiffel and M. Thielscher and proved superior to Cluneplayer. Similarly to Cluneplayer, it depends on depth-first game tree search algorithm with few enhancements and fuzzy logic based evaluation functions.

3. Dr. Tom Murphy's agent, to play Mario game. From a sample gameplay recording, it formulates objective function. The objective function is then used to guide search over possible inputs, using an emulator. The purpose of the agent is to develop an elegant and simple objective function through which a novel gameplay can be generated.

4. Google DeepMinds agent, which can play Atari (a classic gaming console) games using only pixel data from the screen. This implementation involves extracting features from raw input i.e. Screen frames. It uses convolutional neural networks to learn successful control policies from raw video data in complex Reinforcement Learning environments. The network is trained with a variant of the Q-learning algorithm.

III. PROPOSED ALGORITHM

The use of genetic programming is proposed. A general evolutionary algorithm works as follows –

1. A population of genetic encodings of neural networks is evolved in order to find a network that solves the given task.
2. Each encoding in the population (a genotype) is chosen in turn and decoded into the corresponding neural network (a phenotype).
3. This network is then employed in the task, and its performance over time measured, obtaining a fitness value for the corresponding genotype.
4. After all members of the population have been evaluated in this manner, genetic operators are used to create the next generation of the population.
5. Those encodings with the highest fitness are mutated and crossed over with each other, and the resulting offspring replaces the genotypes with the lowest fitness in the population.
6. The process therefore constitutes an intelligent parallel search towards better genotypes, and continues until a network with a sufficiently high fitness is found.
7. For running the games, Bizhawk open-source emulator is used which runs on a Windows machine. Bizhawk has support for LUA Scripting, that allows to observe game state and send control signals to it while the game is running.

The system is functionally divided into 3 major components. These are as follows –

1) Input Extraction.

Here we read the RAM memory of the game and extract game details like score. The environment of game is represented as a simple integer array with different values for different types of objects. For example, 0 for empty space, 1 for wall, -1 for enemies and so on. This input array is sent to the other modules and the neural networks.

2) Genetic Evaluation.

Here, the genetic algorithm, as described earlier, is used to evaluate the neural network. The final aim being the network being able to finish the current level of game.
Initially a pool of around 250 to 300 networks are initialized in the form of genome, an alternate representation of neural networks on which genetic algorithms can operate. Then after one round of mutation

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

and cross-reproduction operation on the pool, each genome is converted to its corresponding neural net and evaluated. The resultant score is noted as fitness of that genome.
After entire pool undergoes this process, the genetic algorithms operate on it again.
This keeps on going until the current level is successfully completed.

3) Neural Network Generation and Evaluation.
As mentioned in the previous section, genetic algorithms operate on genomic representations of neural networks. This component has modules to convert genotypes to corresponding neural networks, as well as to run it against the game level.

IV. PSEUDO CODE

The main program's flow of control is as follows-

1) Define buttons as per the game.
2) Define size of input to the agent.
3) Define total population of species in genetic programming and other related parameters.
4) Define timeout constant.
5) Define maximum number of nodes allowed in neural network.
6) Initialize pool of species if pool is nil.
   6.1) Perform initialization.
      6.1.1) Reset progress parameter.
      6.2.2) Set timeout value.
      6.2.3) Save state of current pool.
      6.2.4) Generate NN from current genome of pool and evaluate the genome fitness.
      6.2.5) Set joy pad controls as per output of network.
7) Load/Reload GUI with values of progress, maximum progress and current genome, species value.
8) Display network if corresponding button pressed.
9) Generate neural network from current genome of pool and evaluate the genome fitness.
10) Get current position/state of player.
11) If position/state improved from previous value, update the max progress attained value as well as reset timer.
12) Reduce timer countdown.
13) If time out –
    13.1)Evaluate new fitness of species.
    13.2)If new fitness larger than previous max fitness of pool, set max fitness as fitness of this genome.
    13.3)Go to next genome.
    13.4)Reinitialize.
14) Update GUI
15) Increment currentFrame parameter of pool.
16) Call emulator for next frame of game.
17) Repeat from step 7.

V. RESULTS

The current implementations were successful on levels of two games, Super Mario World, which is a 2-D Running and Jumping game for avoiding obstacles to reach the goal, and Mario Kart, a Racing game. The rules of any of these games were not provided beforehand.
For Super Mario World, the training time was around 5 hours and level was finished in 15 generations. Similarly, for Mario Kart, the player won in 13 generations. It was also observed that adding opponents reduced the learning time in Mario Kart.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

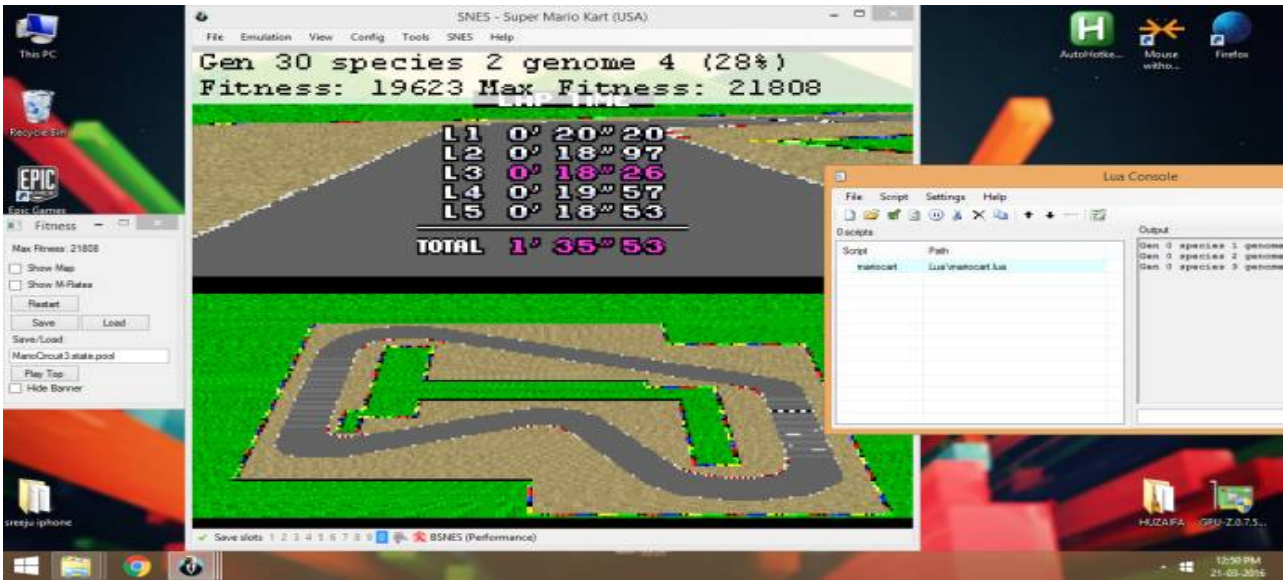

Figure 1: A snapshot of the agent playing Mario Kart.

In the Figure 1, we see the GUI of agent including the buttons to load/store state of training, play/pause the training and the overlay of information of current generation and species over game screen. The agent is run against Super Mario Kart in above screenshot, where it can be seen it has completed the track.

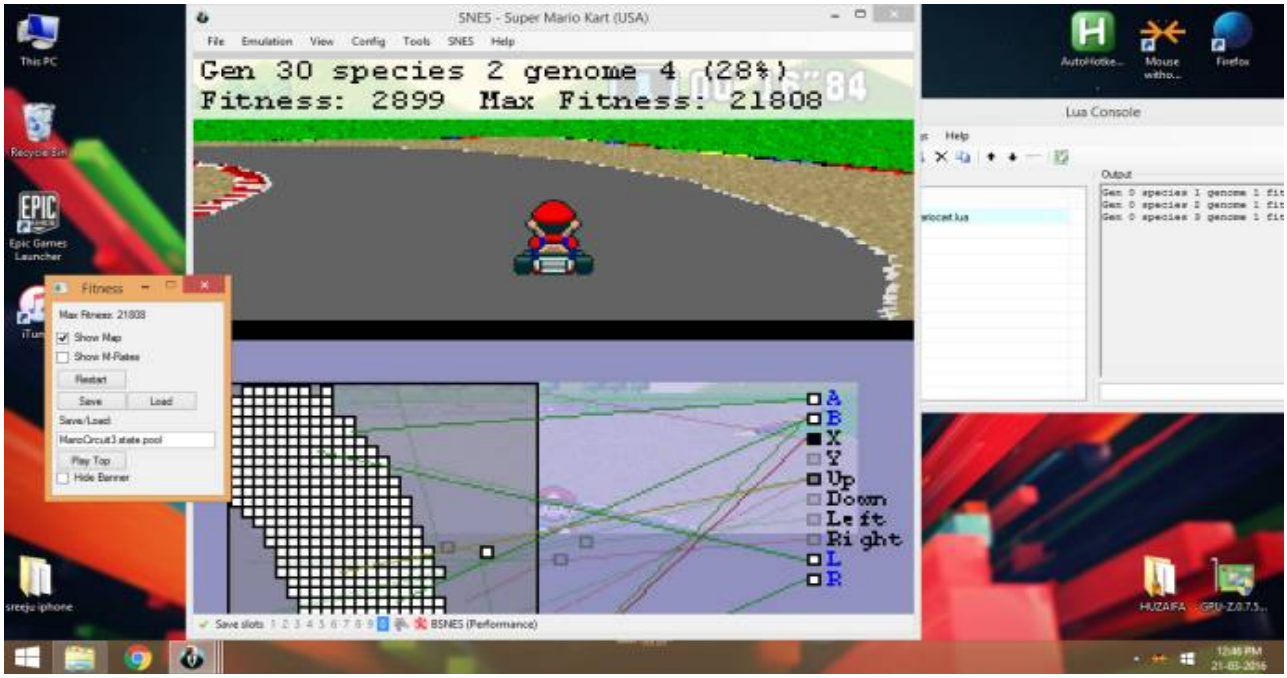

Figure 2: A snapshot of the agent playing Mario Kart, along with the feature extractor working, mapping states with actions.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

In figure 2, the screenshot is shown of the agent in action during the game of Super Mario Kart. The image below game screen is the representation of the input grid to neural network as well as the neural network, with its hidden and output layers, in action. This network representation can be shown or hidden by checking or unchecking the 'Show Map' button of GUI.

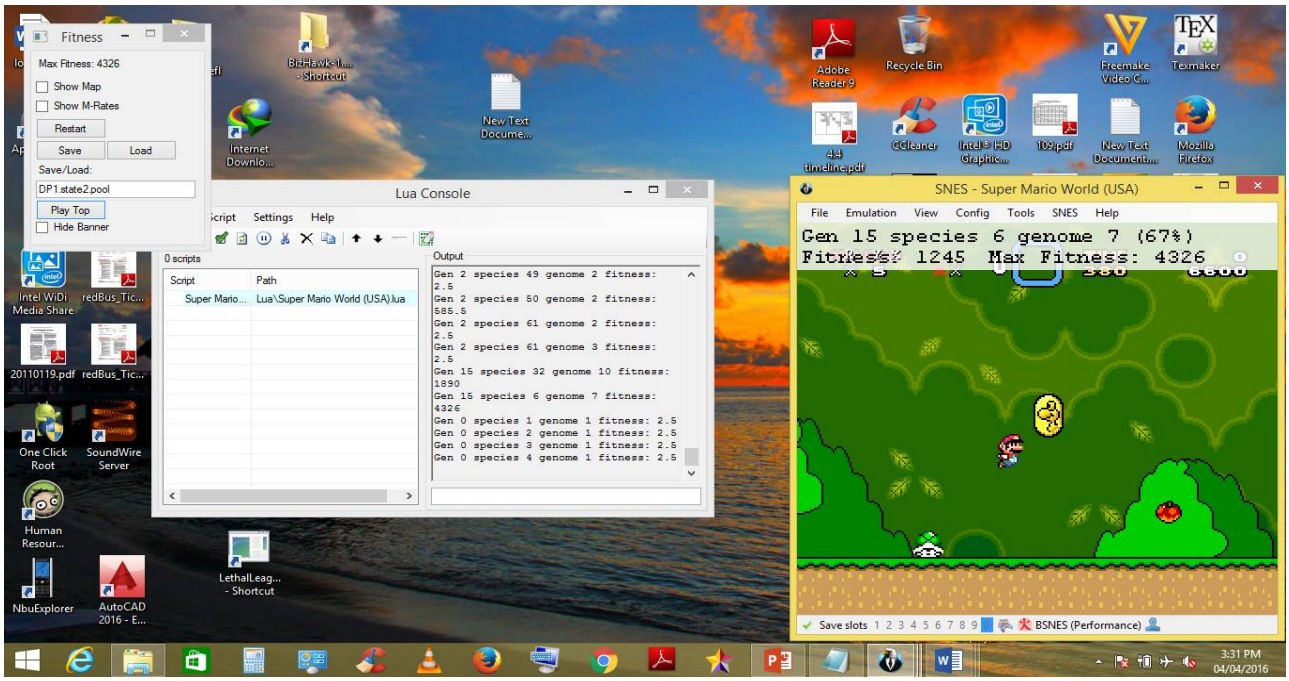

Figure 3: A snapshot of the agent playing Mario Super World, along with its feature extractor working, mapping states with actions and the Bizhawk emulator working in the background.

The above Figure 3 shows screenshot taken while the agent plays Super Mario World game. The generation and species of current network is visible. Also Mario is seen jumping and avoiding enemies and covering distance of around 1200. This shows the gradual progress of Mario.

VI. CONCLUSION AND FUTURE WORK

Thusthe implementation of a general video game player is possible and efficient using genetic algorithms to train neural networks. Two games belonging to different categories were successfully played and won by the agent. Genetic algorithms thus prove useful for intelligent agents to learn to make correct decisions in the game world. The statistics also indicate that in some cases, the agent outplays an average human performance by a considerable margin in the time needed to finish the level. The future of these trainable software agents is in the real world problem solving opportunities. E.g. A self-driving car.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

REFERENCES

1. Dr. Tom Murphy, "The First Level of Super Mario Bros. is Easy with Lexicographic Orderings and Time Travel ... after that it gets a little tricky.", Sigbovik 2013.
2. Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou Daan Wierstra Martin Riedmiller, Playing Atari with Deep Reinforcement Learning, arXiv:1312.5602v1 [cs.LG] 19 Dec 2013
3. Michael Thielscher, General Game Playing in AI Research and Education, The University of New South Wales, Proceeding KI'11 Proceedings of the 34th Annual German conference on Advances in artificial intelligence Pages 26-37.
4. Michulke, D., Thielscher, M., Neural Networks for State Evaluation in General Game Playing, Proceedings of the European Conference on Machine Learning (EMCL) Pages 95-110 (2009)
5. Kenneth O. Stanley, Risto Miikkulainen, "Evolving Neural Networks through Augmenting Topologies, Journal of Evolutionary Computation, Volume 10 Issue 2, Summer 2002 Pages 99-127
6. John Levine, Clare Bates Congdon, Marc Ebner, Graham Kendall, Simon M. Lucas, Risto Miikkulainen, Tom Schaul, and Tommy Thompson, General Video Game Playing, Artificial and Computational Intelligence in Games. Dagstuhl Follow-Ups Volume 6, 2013

BIOGRAPHY

The team of authors of this paper are final year (B.E.) students of the Computer Engineering Department of Sinhgad College of Engineering, Pune -411041, Maharashtra, India. The project idea, its exploration was appreciated at the "MIT Technogenesis 2016", a national level project competition conducted by MAEER's MIT group of colleges, Pune. This paper is written to present their work done during the course of their undergraduate degree curriculum under Savitribai Phule Pune University (formerly known as University of Pune).