Evolving Tic-Tac-Toe Playing Algorithms Using Co-Evolution, Interactive Fitness and Genetic Programming

Helia Mohammadi, Nigel P. A. Browne, Anastasios N. Venetsanopoulos, and Marcus V. dos Santos

***Abstract*—This paper presents a novel use of Genetic Programming, Co-Evolution and Interactive Fitness to evolve algorithms for the game of Tic-Tac-Toe. The selected tree-structured algorithms are evaluated based on a fitness-less double-game strategy and then compete against a human player. This paper will outline the evolution process which leads to producing the best Tic-Tac-Toe playing algorithm. The evolved algorithms have proven effective for playing against human opponents.**

***Index Terms*—Co-evolution, game algorithms, genetic programming, interactive fitness, tic-tac-toe, tournament selection**

I. Introduction

In this paper we deployed Genetic Programming, Co-Evolution, and Interactive Fitness to create a human-competitive algorithm for playing Tic-Tac-Toe game.

A game of Tic-Tac-Toe, also known as "Noughts and crosses", involves two players, one playing as an "X" and the other playing as an "O". The game board consists of a 3x3 grid, where a player may put their symbol into an empty position in the grid. Only one player's symbol may occupy each position. The objective of the game is for a player to place three of their symbols in a row, either vertically, horizontally, or diagonally. The game is considered a draw, if neither player is able to successfully get three symbols in a row.

The proposed method for generating a Tic-Tac-Toe playing algorithm is to use Genetic Programming (GP), Co-Evolution and periodic Interactive Fitness. GP is one of many methodologies for computer simulation of evolution. It is inspired by biological evolution, mainly Darwin's theory of evolution [1], to evolve tree-structured computer programs in order to generate more efficient solutions and programs.

Co-Evolution, also known as fitness-less evaluation, replaces the standard fitness measurement in GP by playing one evolved algorithm against another. The winner of these games is deemed to have a better fitness value and is selected for further evolution. It is important to note that this is qualitative.

We are proposing the use of periodic Interactive Fitness evaluation, by which the individuals are played against a human player, to guide evolution. This methodology provides the creation of more capable individuals through the generations. The evolved algorithms not only play against the other evolved algorithms but also are evaluated by a human player after each run. Therefore the best individuals are kept and the worse algorithms are eliminated by the human player.

The goal of this paper is to successfully use GP, Co-Evolution and Interactive Fitness to develop a human-competitive algorithm for playing Tic-Tac-Toe and to evaluate the impact of Interactive Fitness evaluation in contrast to a simple fitness-less Co-Evolution. The success of this project will be measured by the algorithm's ability to play the Tic-Tac-Toe game against a human competitor.

Genetic Programming is a methodology based on evolutionary algorithms. Stephen F. Smith was the first person who reported results on this methodology in 1980. However, John Koza, the main proponent of GP, popularized GP and applied it to several complex optimization and search problems. Additionally, he was the first to separate GP from Genetic Algorithms (GA) [2].

There are three main operators used in GP: Reproduction, Cross-over and Mutation. Reproduction is basically the replication of the same tree or individual to the next generation without performing any changes to it. This operation is mainly utilized when the fitness value of the individual is comparatively high. Cross-over is applied using two individuals (parents) and creates two children (offspring) by exchanging the whole sub-tree of the selected node of the parents, at the specified crossover points [3], [2]. Mutation affects an individual by replacing one of its nodes with a new element or sub-tree.

Co-Evolutionary algorithms are mainly utilized in artificial life, machine learning, optimization and game learning. Daniel Hills, who Co-Evolved sorting networks, and Karl Sims, who used Co-Evolution to evolve virtual creatures, were the pioneers of Co-Evolution methods. In this paper, we focus on evolving a human-competitive game-playing algorithm for the Tic-Tac-Toe game.

II. Background

There have been few works done to optimize the existing algorithms in order to generate a fast and efficient

Manuscript received January 5, 2013; revised March 8, 2013. This work was supported in part by the Ryerson University (RGS) and Government of Ontario (OGS).

H. Mohammadi is with the Electrical and Computer Engineering Department, University of Toronto, Ontario, Canada (e-mail: helia.mohammadi@utoronto.ca).

Nigel P. A. Browne was with the Computer Science Dept., Ryerson University, Toronto, Ontario, Canada (e-mail: nbrowne@acm.org).

Anastasios N. Venetsanopoulos is with the Department of Electrical and Computer Engineering, University of Toronto and also Department of Electrical Engineering, Ryerson University, Toronto, Ontario, Canada (e-mail: anv@comm.utoronto.ca).

Marcus V. dos Santos is with the Computer Science Dept., Ryerson University, Toronto, Ontario, Canada (e-mail: m3santos@ryerson.ca).

DOI: 10.7763/IJCTE.2013.V5.799

Tic-Tac-Toe playing algorithm. D.B. Fogel produced interesting results in an earlier application of Neural Networks to competitive games [4]. His model was based on multi-layer feed-forward Neural Networks. In both [5], [6], they examined a look-ahead planning model based on a recurrent Neural Network and applied it to the game of Tic-Tac-Toe.

A review of recent literature showed that little research has been done using Co-Evolution as the sole method to improve a Tic-Tac-Toe algorithm. [7] introduced a Co-Evolutionary system in which they utilized Shared Sampling and Competitive Fitness Sharing. [8] performed an interesting implementation using single population Co-Evolution. The authors of [8] have introduced Fitness-less Co-Evolution in order to generate Tic-Tac-Toe playing algorithm. However, they did not utilize Mutation operation along with Reproduction and Cross-over. [9] reduced the possible states of a Tic-Tac-Toe game to 827 states, by eliminating the seven similar states produced by rotating or flipping the game board. This idea is an inspiration to reduce the terminal set to three terminals (corner, center and edge), instead of nine positions of the playing game board. [10], [9] have utilized GA in order to produce Tic-Tac-Toe game playing algorithm.

This paper aims to apply Co-Evolution and Genetic Programming to the game of Tic-Tac-Toe, since it is a competition-based game containing two players; thus, it is a suitable game to perform competitive Co-Evolution on. It is a simple game with Minimax strategy and few interesting works have been done on this game, starting with Michie's MENACE [11], through Susan Epstein's Hoyle system [12]. Achieving good results from applying Co-Evolution on a simple game of Tic-Tac-Toe is a leading way to solving the many existing unsolved problems.

III. Methodology

In this paper we used Genetic Programming, Co-Evolution and Interactive Fitness in order to evolve a human competitive Tic-Tac-Toe game playing algorithm.

We selected Genetic Programming (GP) as the mechanism of evolution over Genetic Algorithms (GA) because of the ability for GP to evolve more expressive programs. Genetic programming is a tree-based evolutionary algorithm that consists of a population individuals competing against each other in order to improve their ability to perform an assigned task. The trees that represent the individual programs are executed in a depth first manner, where the leaf nodes are terminal values and the branch nodes are functions. The terminals and functions are selected from pools or sets referred to as the Terminal Set and the Function Set. Inspired by the implementation of [5], our terminal represent game board positions and the function sets are game playing queries and tasks.

The Terminal Set uses constant values based on row and column notation to refer to each possible board position. For example: {pos00, pos01, ..., pos22}. As Fig. 1 demonstrates, these are the positions on the game board. As mentioned earlier, the game board consists of nine positions in which the players can place their symbols.

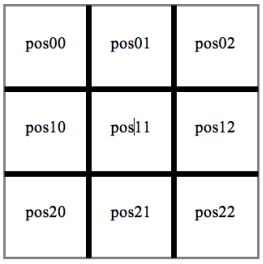

Fig. 1. Tic-tac-toe game board positions

The Function Set contains the following game operations: AND, OR, IF, MINE, YOURS, OPEN, and PLAY-AT. These functions return either a position on the game board or the value NIL, which is introduced in [8] and discussed shortly. The "IF" function is the only operator in the function set that requires three arguments. This function will return the second argument, if the first argument (the condition of the "IF") is non-NIL, and the third argument, otherwise. If none of the arguments are NIL, "AND" returns the second argument, otherwise it returns NIL. "OR" returns the first non-NIL argument. If both of the arguments are NIL, it will return NIL. MINE, YOURS, and OPEN return the position that is passed to them if it belongs to the player, the opponent or is empty, respectively. They return NIL otherwise. PLAY-AT places the symbol of the player in the position that is passed to it and makes the player wait for the opponent to play. It returns the position otherwise.

There is a special return value NIL utilized by the functions in this algorithm. The NIL value is used to represent the absence of a value and is generated as a return value from a function. The NIL value is not considered a terminal value, because it is never used as the value of a tree node.

We evolved tree-structure individuals with an initial tree depth size of 20. Each node has a parent link and three possible child links. All the child links that do not exist are null or empty. The non-leaf nodes are the functions from the function set and the leaf nodes are the 9 positions of the game board. We populate the child links from left to right. That is, for the functions with only of child (such as MINE, YOURS, PLAY-AT, and Open), the terminal is assigned to the left child. For the functions with two child links (such as OR and AND), the left child and middle child are assigned. Finally, for our only function with three child links (IF function), all the left child, middle child and right child are assigned from left to right.

The trees are traversed in a depth-first order and the traverse function is a recursive function that is forced to return a value when meeting a PLAY-AT function.

According to [8] one population Co-Evolution is computationally more feasible than a two population Co-Evolution, therefore, we utilized Co-Evolution with a single population.

Considering the fact that there is a greater probability of winning for the player who has started the game, we employed a double-game strategy. Both players have the opportunity to be the first player. A player is assigned 20 points if it wins both games and 5 if wins one game and loses the other. We have observed players with no PLAY-AT function in their algorithms; therefore, we have decided to

assign extra points regarding the number of PLAY-AT function in the tree-structure of the individual. If the individual has the turn and makes a move, they gain a point.

We employed Random Sampling method to randomly select a group of opponent algorithms to play against the individuals of the population. The individual plays a double-game with the first individual of the group and the winner plays against the next individual of the sampled group. This group is selected in a manner that does not contain the individual of the population that starts playing against the group. Afterwards, the next individual in the population is selected to play against another randomly selected group of individuals. This procedure iterates over every individual of the population and the best individuals of each sampled group are stored in the parent pool.

After selecting the best individuals and storing them in the parent pool, we apply the GP operators on the individuals. The likely hood that a GP operator will be applied to a given individual is specified as tuning parameters prior to the run. We utilized three standard GP operators: Crossover, Mutation and Reproduction. We prevented the Mutation operator from modifying the root node of the individuals, because it was observed to be excessively disruptive.

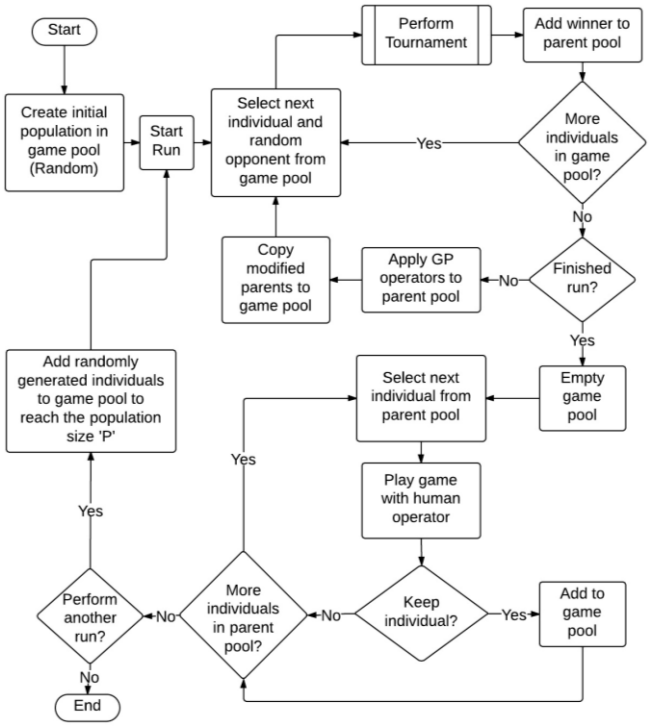

Fig. 2. Overall architecture

To evaluate the fitness of the individuals, we have employed a simple tournament technique in addition to Interactive Fitness Evaluation. The tournament selection technique selects a sample of individuals from the population and plays them against other members of the random sample, which then evaluates the fitness of the individuals. The fitness of the individual is the sum of the number of wins and the bonus points that was explained earlier. This method has been used during the double-games between the individuals and the randomly selected group. The winner of each tournament is used as a parent for the next generation of individuals.

After the completion of a run (or series of generations), the Interactive Fitness Evaluation (IFE) phase begins, during which a human player plays against the evolved individuals and decides which N individuals provide the best game-play and should be used as seeds for the next run. Since N individuals are used as seeds for the next evolutionary run and we need to maintain a constant population size of P, (P - N) individuals are randomly generated and added to the population for the next run. Since no Interactive Fitness occurred before the first run, the starting population of run number 0 is entirely composed of random individuals. Fig. 2 illustrates the overall architecture of our system.

IV. Results

The expected result of this experiment is that a Genetic Programming system utilizing Co-Evolution and Interactive Fitness as its fitness mechanism, will successfully develop a human competitive Tic-Tac-Toe playing algorithm for a 3x3 Tic-Tac-Toe board.

The experiments consisted of a series of runs, utilizing different combinations of parameters and evaluation methods. The population size was tested from 64 to 512 individuals, but 128 seemed to provide the best results in the shortest time.

The probability of crossover was tested from 0 to 1, in 0.1 increments. It was observed that too little crossover hurt the evolutionary process, but so did too high a crossover level. The final value that provided the best balance was 0.8. Since the implementation selected an individual either for crossover or reproduction, the reproduction value was always (1 - crossover).

Finally, a mutation rate from 0 to 1 was also tested. Too low of a mutation value caused the population to become stagnant and not evolve. However, when the mutation was very high, it was extremely disruptive, causing the population to not evolve effectively. Ultimately, a mutation rate of 0.5 seemed to work best.

The parameters of the final experiment were:

- Population size = 128
- Depth size = 20
- Crossover probability = 0.8
- Mutation probability = 0.5
- Replication probability = 0.2
- Coevolution and Interactive Fitness

After evolving an effective Tic-Tac-Toe playing game, we have performed several games in order to determine whether the obtained algorithm is a human-competitive algorithm or not.

We have performed several experiments to determine if the original hypothesis is valid. By evolving optimal players using fewer evaluations than what has been done until now, a great amount of success can be achieved.

We have performed the methodology explained earlier and gained promising results.

V. Discussion and Future Work

By randomly selecting a group of individuals from a single

population, we greatly reduced the run time. Individuals of the population were randomly generated in the first generation and evolved quickly through the generations. Fig. 3 demonstrates a case where the individuals of the population competed against one another and co-evolved without employing the Interactive Fitness evaluation. It can be observed that individuals have fewer wins and there are a few fit individuals in the population.

As Fig. 4 demonstrates, applying Interactive Fitness evaluation to the fitness evaluation procedure of the individuals in the population, we were able to highly enhance the evolution process. The results demonstrate that the individuals have become more enhanced and are able to defeat more algorithms. This Fig also shows that the individuals have successfully co-evolved and the difference between the number of wins of the individuals have decreased.

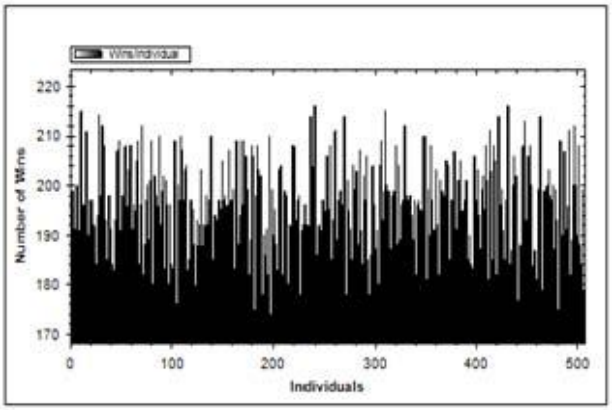

Fig. 3. Co-evolution of 512 individuals without employing interactive fitness evaluation

The best 20 individuals of the run and 20 random individuals were played against a human player. This was observed to drastically improve the playability of the individuals. However, it was surprising that the fitness values of the population did not appear to be drastically different. It would appear that by guiding the evolution using Interactive Fitness, the over fitness was not affected as much as the playability of the individuals. This is significant, because while it is possible to create highly fit individuals, the metric of playability can only be ascertained by a human competitor.

Another significant and unexpected result of the experiments was the evolution of the location and use of the play-at function. Without Interactive Fitness the individuals of the population were observed to commonly evolve multiple play-at nodes. This was the expected evolutionary path, since more play-at nodes would ensure different strategies and higher fitness. However, when Interactive Fitness was introduced, it was observed over multiple experiments and runs that the trees would quickly evolve to a structure with only a single play-at node, which was always in the root of the tree. This seems to work more like a human opponent would plan and execute a strategy.

In the future, we are planning to perform a comparison between different sampling and fitness evaluation methodologies discussed in [9] in order to achieve a more enhanced game playing algorithm.

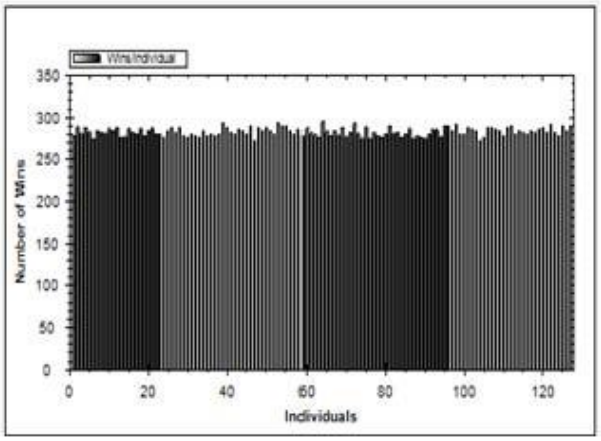

Fig. 4. Co-evolution of 128 individuals after employing interactive fitness evaluation

As mentioned before, Hochmuth has explained the fact that there are seven similar playing states, which can be mapped into only one game state by rotating or flipping the game plane. This would be an inspiration for a novel solution to reduce the number of the similar states by changing our proposed terminal set. By changing "pos00, pos02, pos20 and pos22" to "corner", "pos01, pos10, pos12 and pos21" to "edge", and "pos11" to "center", our terminal set would contains only three elements and we are able to eliminate these seven identical permutations for each game state hence reduce massive replicated game states.

VI. Conclusions

This paper was a demonstration of a novel way of using Co-Evolution and Genetic Programming in order to evolve a human-competitive Tic-Tac-Toe playing algorithm. We have performed Random Sampling to select a number of individuals to play as the opponents of the individual that was to be evaluated. As mentioned earlier, Co-Evolution is fitness-less evolution of individuals which are being improved by competing against one another. In order to evaluate the fitness of the individuals, we have performed Simple Tournament method which is based on the number of wins an individual can achieve through playing against the randomly selected group.

Furthermore, we have used Interactive Fitness evaluation by which we were able to enhance the obtained individuals in order to achieve a human-competitive algorithm.

References

[1] C. Darwin, *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*, London: John Murray, 1859.

[2] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, Mass.: The MIT Press, 1992.

[3] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming, an Introduction*, San Francisco, CA: Morgan Kaufmann Publishing., 1998.

[4] D. Fogel, "Using evolutionary programming to create neural network that are capable of playing tic-tac-toe," in *Proceedings of ICNN93*, pp. 875-880, 1993.

[5] P. Angeline and J. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 264-270, 1993.

[6] Y. Sato and T. Furuya, "Coevolution in recurrent neural networks using genetic algorithms," *Syst Comput Jpn*, vol. 27, no. 5, pp. 64-73, 1996.
[7] C. Rosin and R. Belew, "Methods for competitive co-evolution: Finding opponents worth beating," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 373-380, 1995.
[8] W. Ja śkowski, B. Wieloch, and K. Krawiec, "Fitnessless coevolution," *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 2008.
[9] G. Hochmuth, "On the genetic evolution of a perfect tic-tac-toe strategy," *Genetic Algorithms and Genetic Programming at Stanford*, pp. 75-82, 2003.
[10] A. Bhatt, P. Varshney, and K. Deb, "In search of no-loss strategies for the game of tic-tac-toe using a customized genetic algorithm," in *Proc. 10th Annual Genetic and Evolutionary Computation Conference, GECCO 2008*, pp. 889-896, Atlanta, GA, 2008.
[11] D. Michie, "Trial and error," *Science Survey*, Part 2, pp. 129-145, 1961.
[12] S. L. Epstein, "Learning plans for competitive domains," in *Proceedings of the Seventh International Conference on Machine Learning*, pp. 190-197, 1990.

Helia Mohammadi is a Ph.D. student of the Edward S. Rogers Department of Electrical and Computer Engineering at the University of Toronto. She received her M.Sc. degree from Ryerson University, department of Computer Science in 2010. She has two Bachelor's of Science degrees, one in the field of Computer Engineering and the other in the field of Business Management. She has received a number of scholarships and awards, including: Natural Sciences & Engineering Research Council of Canada (NSERC), Ontario Graduate Scholarship (OGS), Ryerson Graduate Scholarship (RGS), Edward S. Rogers Sr. Graduate Scholarship, and Graduate Teaching Assistant Award of Excellence. Additionally her Master's thesis (a Precarn funded project) was nominated for the Governor General's Gold medal from the department of Computer Science at Ryerson University and was awarded the "Best Graduate Thesis Award in Recognition of Excellence in Research and Nomination for Governor General's Gold Medal".

Nigel Browne is currently a senior programmer/analyst with the Woodbridge Group based in Mississauga, Canada, but with work experience throughout Mexico, the USA and Canada. He received his Master of Science in computer science from Ryerson University in 2009 and his Bachelor of Science in computer science from Ryerson University in 2005. His thesis "Adaptive Representations for Improving Evolvability, Parameter Control, and Parallelization of Gene Expression Programming", was published in Applied Computational Intelligence and Soft Computing, Volume 2010 (Hindawi). Mr. Browne is a member of the Association for Computing Machinery and was a receipient of Ryerson Graduate Scholarship.

Anastasios N. Venetsanopoulos is a professor of electrical and computer engineering at Ryerson University, founding VP-Research and Innovation at Ryerson University (2006-2010), Professor Emeritus of the Edward S. Rogers Department of Electrical and Computer Engineering at the University of Toronto, and the 12th dean of the Faculty of Applied Science and Engineering at the University of Toronto (2001-2006). He has published over 800 papers on digital signal and image processing and digital communications and has served as Chair on numerous boards, councils and technical conference committees including IEEE committees. In 1994 Dr. Venetsanopoulos was awarded an Honorary Doctorate from the National University of Technology in Athens, Greece. In 1996, he was awarded the "Excellence in Innovation" Award from the Information Technology Research Centre of Ontario and the Royal Bank of Canada for his work in image processing. Venetsanopoulos was also awarded the "Millennium Medal of IEEE", and the MacNaughton Medal". In March 2006, he was a joint recipient of the IEEE Transactions on Neural networks Outstanding Paper Award. He is a Fellow of the Engineering Institute of Canada, the IEEE and the Canadian Academy of Engineering. In 2008, A. N. Venetsanopoulos was awarded the "Most Cited Paper Award" by the Journal of Visual Communication and Image Representation for their work in artificial neural networks. In 2010, Dr. Venetsanopoulos was elected as Fellow of the Royal Society of Canada.

Marcus Dos Santos is currently an associate professor, undergraduate program director of the department of computer science, Ryerson University, Canada, where he supervises research students both at the undergraduate and graduate level. Marcus was born and raised in Minas Gerais, Brazil. He studied at the Federal University of Uberlândia, where he received Bachelor's and Master's degrees on computer engineering, and at the University of São Paulo, where he received a Ph.D. in Computer Engineering. His research interests reside in advancing the understanding and application of evolutionary computing systems. In 2007, he was a visiting professor in the Department of Electrical Engineering at Chung-Ang University, Seoul, Korea.