Evolving Game Playing Strategies for Othello

Clive Frankland
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Pietermaritzburg, South Africa
clive.frankland@promix.co.za

Nelishia Pillay
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Pietermaritzburg, South Africa
pillayn32@ukzn.ac.za

***Abstract*— There has been a fair amount of research into the use of genetic programming for the induction of game playing strategies for board games such as chess, checkers, backgammon and Othello. A majority of this research has focused on developing evaluation functions for use with standard game playing algorithms such as the alpha-beta algorithm or Monte Carlo tree search. The research presented in this paper proposes a different approach based on heuristics. Genetic programming is used to evolve game playing strategies composed of heuristics. Each evolved strategy represents a player. While in previous work the game playing strategies are generally created offline, in this research learning and generation of the strategies takes place online, in real time. An initial population of players created using the ramped half-and-half method is iteratively refined using reproduction, mutation and crossover. Tournament selection is used to choose parents. The board game Othello, also known as Reversi, is used to illustrate and evaluate this novel approach. The evolved players were evaluated against human players, Othello WZebra, AI Factory Reversi and Math is fun Reversi. This study has revealed the potential of the proposed novel approach for evolving game playing strategies for board games. It has also identified areas for improvement and based on this future work will investigate mechanisms for incorporating mobility into the evolved players.**

Keywords—genetic programming; game-playing; board games, Othello

I. INTRODUCTION

Game playing has been one of the founding areas of artificial intelligence. Initially search was essentially used for inducing game playing strategies. As the field developed researchers started looking to machine learning techniques to improve the performance of computer players. One such technique is genetic programming [1] which has proven to be fairly effective for generating game playing strategies. Genetic programming is an evolutionary algorithm which searches a program space rather than a solution space which is typical of genetic algorithms [2]. There has been a fair amount of research into using genetic programming for evolving game playing strategies for board games. This research has essentially involved using genetic programming to evolve board evaluation functions which are used together with a game search algorithm, such as the alpha-beta algorithm or Monte Carlo tree search, to decide which move should be made next.

This approach has been successfully used to induce game playing strategies for lose-checkers [3, 4], endgames for chess [5, 6], backgammon [7], Othello [4, 8, 9], Dodgem [8, 9, 10] and Hex [10]. In [11] the idea of using GP to evolve evaluation functions is extended to evolve a second evaluation function for determining which child nodes should be expanded. The research presented in this paper proposes a different approach from previous work based on heuristics. For all two player zero sum board games certain positions on the board are more strategic than others. The proposed approach views the board as a set of areas, with a heuristic representing each area. Each game playing strategy, representing a player, is composed of heuristics and determines the next move to be made. In this study genetic programming is used to evolve both the heuristic values and game playing strategies, i.e. the player. The strategies are evolved in real time during play unlike previous work in which the strategies are induced offline. The proposed approach is illustrated using the board game Othello and evaluated against human players and three well known online Othello players.

The following section provides an overview of Othello. Section III presents the genetic programming approach and section IV describes the experimental setup for implementation and evaluation. The performance of the proposed GP approach is discussed in section V. Section VI provides a summary of the findings and outlines future extensions of this work.

II. OTHELLO

Othello, also known as Reversi, is a two player zero sum game. The version of Othello used in this study is an 8 x 8 non-checkered board with 64 discs. All 64 discs are identical with one white side and one black side. To start the game each player takes 32 discs and chooses one colour to use throughout the game (black or white). At the start of the game one player places two black discs and the other places two white discs at the center of the board. Each player takes turns placing discs on the board with their colour facing up. If a newly placed disc bounds the opponent's discs in a straight line, the bounded discs are to be flipped, adding to the player's disc count. The game ends when all of the positions of the board are occupied or no possible moves are left. The player with the higher disc count is then declared the winner.

978-1-4799-7492-4/15/$31.00 ©2015 IEEE

Research into generating game playing strategies for Othello was initiated by the work of Rosenbloom in 1982 which essentially used the alpha-beta algorithm [12]. Later research has investigated machine learning techniques. Leouski [13] uses a neural network to evaluate Othello boards using temporal learning. Later work focused on using evolutionary algorithms to evolve neural networks to assess Othello board configurations [14, 15]. As discussed in the previous section genetic programming has essentially been used to evolve evaluation functions to assess Othello board configurations which are used with game search algorithms [4, 8, 9]. An extension of this work also evolves an evaluation function to determine which nodes to expand based on the board configuration evaluations instead of using a standard game search algorithm [11]. The following section proposes a novel heuristic based approach for evolving game playing strategies for two player zero sum games using genetic programming.

III. GENETIC PROGRAMMING APPROACH

A generational genetic programming algorithm presented in [1] is implemented on each run. The general algorithm is depicted in Fig. 1. Each run begins with creating an initial population. This initial population is then iteratively refined via the processes of evaluation, selection and regeneration.

```
Create an initial population
Repeat
  Evaluate the population
  Select parents
  Apply genetic operators to create next generation
Until termination criteria are met
```

Fig. 1. Overview of the GP approach

The application of the general algorithm in Fig. 1 employed in this study to evolve game playing strategies is illustrated in Fig. 2. A separate run is performed for each move in the game with the runs independent of each other. The dynamics of an Othello board may change so dramatically during game play, especially when considering that an entire section of the board can change colour with a single move, that it was considered pointless to keep a single population throughout an entire game. To this end, after each move made by the computer player, the entire population is terminated. Hence a new run is performed to determine each move with generally no information passed on from one run to the next. Each run starts with initial population generation followed by *n* generations. On each generation the reproduction, mutation and crossover operators are applied to create the population. An alpha player is maintained from one generation to the next. At the beginning of a run the alpha player is randomly generated. Alternatively, the alpha from the last generation of the previous run is copied across if a run has already been performed. The processes of initial population generation, evaluation and selection and regeneration are discussed in the sections that follow.

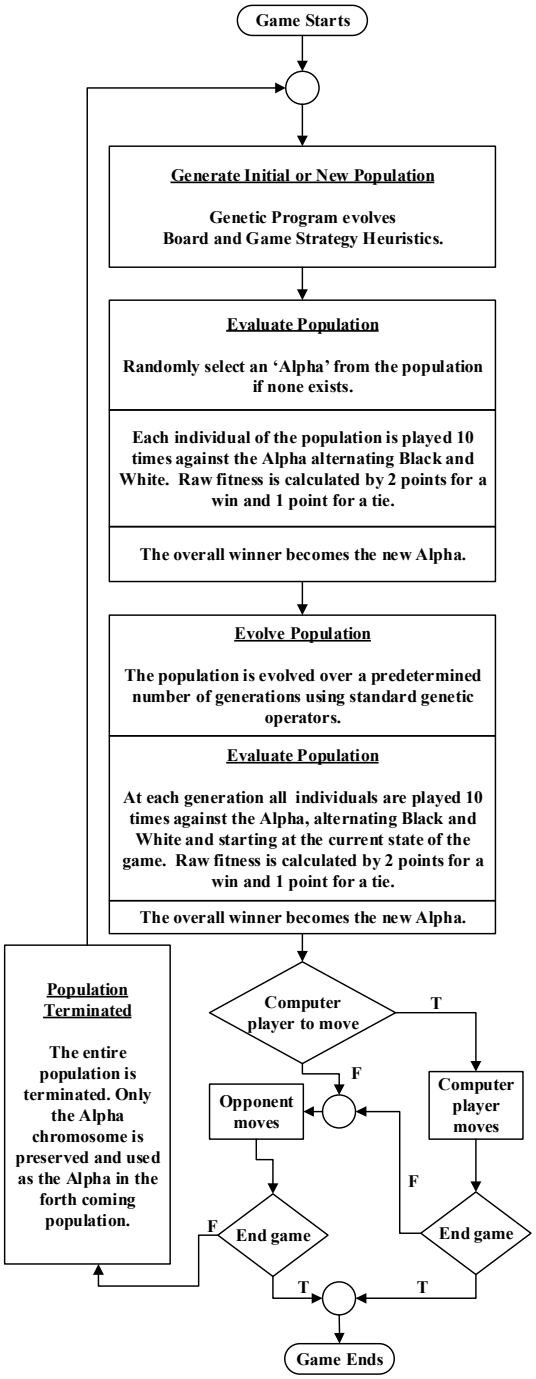

Fig. 2. Overview of the GP approach

A. Initial Population Generation

Each element of the population represents a player and defines a game playing strategy in terms of heuristics. Each heuristic represents an area of the board. The area of the board that each heuristic corresponds to is dependent on the game. For Othello the board has been divided into the nine areas depicted in Fig. 3. The Othello board is made up of 8 x 8 identical squares which can be viewed as 4 x 4 identical quadrants. The strategic areas are defined within the 4x4 quadrant.

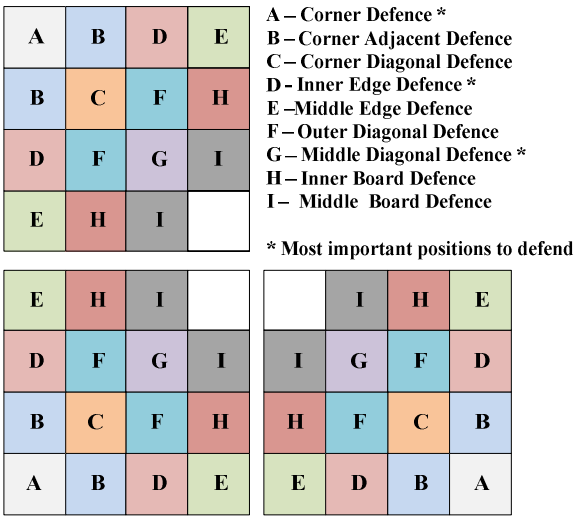

Fig. 3. Three symmetrical quadrants of the Othello board showing the strategic areas.

The terminal set contains an element corresponding to the heuristic for each area. The terminals tBH*i* represent each of the areas as follows: tBH1=A, tBH2=B, tBH3=D, tBH4=E, tBH5=H, tBH6=I, tBH7=G, tBH8=C, tBH9=F. During initial population generation a value is randomly generated for each heuristic. A range of 1 to 32 was used to test the GP. This range of values was selected simply from observation, as a greater or lesser range did not influence the computer player's playing ability. Each game playing strategy is composed of heuristics. Not all the heuristics are necessarily contained in a strategy. The strategy is applied by considering all legal moves and choosing the move that results in the position with the highest heuristic value. In addition to the nine terminals corresponding to the board heuristics, the terminal set includes the following three Boolean terminals corresponding to strategy heuristics:

- tCnrH - The purpose of the heuristic that this terminal represents is to give corner positions priority. If this terminal is set to true and one of the legal moves results in a corner position, the board heuristic value for this position is doubled. If the value of this terminal is false it has no effect.
- tEdgH - This terminal performs the same function as tCnrH for the eight edge positions (not including the corners). If the value of this terminal is false it has no effect.
- tCntH - This terminal prioritizes board positions that will give the player a higher number of friendly discs (i.e. discs of the player's colour). If this terminal is true and there are a number of heuristics with the same value, the position resulting in a higher number of friendly discs for the player will be chosen. If this terminal is false a position from those with the same heuristic value will be randomly chosen.

These heuristics simply give the artificial intelligence an advantage when having to choose between equally weighted board positions and do not instruct the artificial intelligence in Othello strategy. During initial population generation these terminals are randomly assigned a value of true or false. The function set contains two elements, namely, fArg2 and fArg3, which takes two and three arguments respectively. An example of an element of the population is illustrated in Fig. 4.

An example of the use of these heuristics is illustrated using Fig. 4 and Fig. 5. Fig. 5 illustrates the board heuristic values allocated by the player/game playing strategy depicted in Fig. 4 for the first quadrant of the board. As tCnrH has a value of true the values of the corner positions will be doubled, i.e. 56, 22 and 22. Similarly, the heuristic values of the edge heuristics will also be doubled as tEdgH is also true. The heuristic values are used to choose the next move. The move resulting in the board position with the highest heuristic value is chosen. If there is more than one position with the highest heuristic value, the position is randomly chosen from the positions that share the highest value. The initial population is created using the ramped half-and-half method. If a terminal representing the heuristic of a board position does not occur in a strategy, the board position is assigned a value of zero. If a terminal appears more than once in a strategy with different values, the first or last value is randomly chosen.

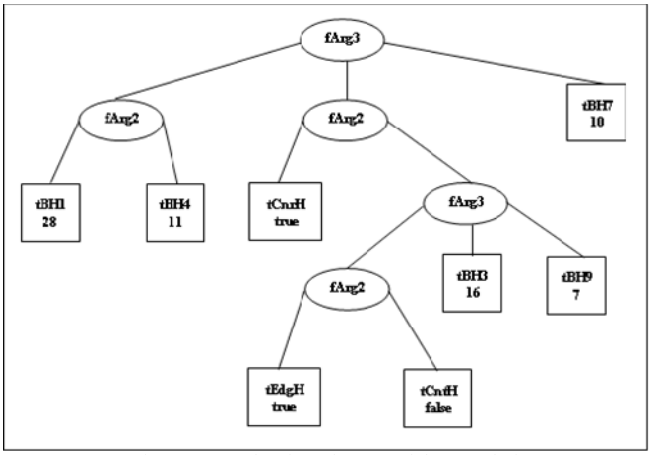

Fig. 4. Example of an element of the population

28	0	16	11
0	0	7	0
16	7	10	0
11	0	0	

Fig. 5. Board heuristic values for the strategy in Fig. 4

B. Evaluation and Selection

The population is evaluated by each element playing against each other. This begins with the first element of the population playing against the alpha player. The number of games played is a parameter value and is preset. Each win is rewarded with 2 points and each tie with 1 point. The raw fitness of the individual is calculated to be the sum of the points. The winner becomes the alpha player and goes on to play the next element of the population. This continues until

the last element of the population is reached. Tournament selection [1] is used to choose parents to create offspring of successive generations. This method randomly chooses *t* elements from the population. The fittest element is returned as the winner of the tournament and a parent. Selection is with replacement, so an element of the population can serve as a parent more than once.

C. Regeneration

The reproduction, mutation and crossover operators are used for regeneration. The reproduction operator copies the parent chosen using tournament selection into the next generation. Mutation randomly selects a mutation point and replaces the subtree rooted at the point with a new subtree. Hence, mutation is the only operator that can change the heuristic value of a terminal. The crossover operator randomly selects a crossover point in each of the parents and swaps the subtrees rooted at the crossover points to create two offspring. Offspring that exceed the preset size limit are pruned by replacing function nodes with the rightmost terminal in the subtree of which the function node is the root.

The alpha player from a generation can be used in the successive generation if the flag, a parameter value, to use it is set to true. In the case of crossover, the alpha player and a parent selected from the population are chosen to produce two offspring. In the case of mutation, it is randomly decided whether to mutate the alpha player or not.

IV. Experimental setup

This section describes the methodology used to evaluate the GP approach presented in this paper. The proposed approach was tested against:

- Twenty human players with expertise ranging from beginners (14 players) to intermediate players (6 players). A minimum of three games were played against each human player. A total of 34 games were played against the beginners and 16 against the intermediate players.
- Othello WZebra[1]. Five games at each level of difficulty were played. In this case the level of difficulty is determined by the number of look aheads. Five games were played for each look ahead, namely, 1, 3, 5 and 7, giving a total of 20 games.
- Reversi from the AI Factory[2]. Five games at each level of difficulty, namely, 3, 5, 7, 8, 9 and 10 were played. A total of 30 games were played.
- Reversi from "Math is fun"[3]. Five games were played at the easy, medium and hard levels giving a total of 15 games.

The following parameter values were used for the GP algorithm:

- A population size of 100 was used. A greater population size increased the time it took for the GP to find a solution and did not improve performance.
- The ramped half-and-half method with a depth limit of 6 was used to create the initial population. This was done to establish a wide range of heuristic values at the start.
- Twenty generations were performed for each run. This allowed for sufficient convergence without creating high runtimes for real time play. Tests on runs with more generations did not produce fitter individuals.
- A tournament size of 4 was used, with higher and lower values not producing any improvements.
- Mutation was given a typically low probability of 10% as this genetic operator is a global search operator resulting in high genetic variety during the evolutionary process. Crossover on the other hand is a local search operator, which limits variety thus preserving genes and was set to 80%, higher than the standard value of 60% suggested by Koza [1]. This is important as one of the main objectives of the experiment was to evolve a population that was highly fit and combines several individuals for a final result. By sharing more genetic code through increased crossover, it was anticipated that the population as a whole would stay more tightly bunched, resulting in better individuals at the end of each run. The probability used for reproduction was 10%. A maximum offspring limit of 64 nodes was used.
- The number of games played by each element of the population during evaluation was set at 10. This proved to be sufficient for purposes of evaluation.
- The parameter "Preserve Alpha Player" is included to specify whether the alpha player should be carried through to the next generation. If this parameter is true the alpha player is selected as a parent for a crossover operator and it is randomly decided whether the alpha player should be muted to create an offspring or not. For all the experiments in this paper this value was set to true.
- The parameter "Supreme Alpha AI" was included to specify whether the alpha player from a previous game should be used as the alpha player at the beginning of a new game or a new alpha be created by randomly selecting an element of the first population. This parameter is set to true for the experiments in this paper indicating that the alpha player from one game must be carried through to the next.

The game Othello was written in Java 7 (build 1.7.0) running on the Windows 7 operating system and developed on a Core i5 Gigabyte laptop with on-board NVIDIA GeForce 1GB graphics card and 4GB DDR3 RAM.

V. Results and discussion

This section reports on the performance of the proposed GP approach in inducing game playing strategies. A discussion of the performance of the approach is firstly

[1] http://www.radagast.se/othello/
[2] http://www.aifactory.co.uk/
[3] http://www.mathsisfun.com/

presented followed by an analysis of the game playing strategies typically evolved by the GP approach.

A. Performance of the Genetic Programming Approach

The proposed approach was able to successfully evolve game playing strategies. One of the differences between this study and previous work was that strategies are evolved in real time during game play instead of offline. As is evident from Fig. 6 the GP approach was able to evolve a strategically strong game playing strategy in under a minute.

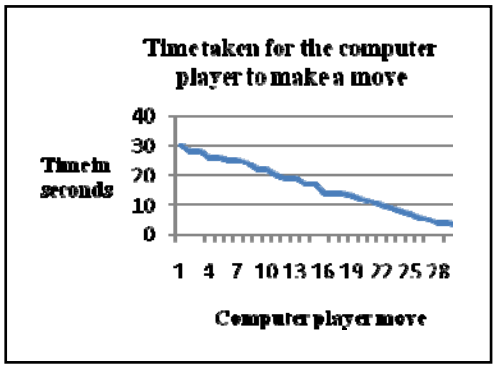

Fig. 6. Time taken for a move

Table 1 lists the results of the games played between the GP approach and the human and online players. The GP approach performed well against the human players winning 81% of the games at beginner level and 69% at intermediate level. The evolved players also played well against WZebra with 1 to 2 moves look ahead, winning 95% of the games. However, as the number of moves that WZebra looks ahead increases, the GP approach becomes less successful. This can be anticipated as the game strategies evolved do not take into consideration any measure of mobility and cannot anticipate future moves that may be made by the opponent.

TABLE I. RESULTS OF GAMES PLAYED BY THE GP APPROACH

Opponent	Level of difficulty	Percentage Computer Player	
		Wins	Ties
Human	Beginner	81%	0%
Human	Intermediate	69%	13%
WZebra	1 - 2 moves look ahead	95%	5%
WZebra	3 - 5 moves look ahead	50%	5%
WZebra	> 5 moves look ahead	0%	0%
AI Factory	Level 1 - 7	95%	5%
AI Factory	Level 8	50%	2%
AI Factory	Level 9 - 10	0%	0%
Math is fun	Easy	95%	5%
Math is fun	Medium	70%	2%
Math is fun	Hard	0%	0%

The GP approach also performed well against the AI factory at level 1 to 8 and the easy and medium level of Math is Fun. The GP approach does not perform as well for levels 9 and 10 of the AI factory and the hard level for Math is fun. It is hypothesized that this can again be attributed to the game playing strategies not including any mobility heuristics. Future work will examine the play at these levels in more detail to test this hypothesis.

B. Performance of the Genetic Programming Approach

The games played by the GP approach were analyzed to identify the strategy employed by the GP players. Observing numerous games played against the GP approach it was noted that in the start and middle-game scenarios, the GP player considers corner and edge positions low priority and assigns them low weight values, whereas positions in the mid-region of the board are considered high priority positions and are assigned high weight values. In the end-game, the GP changes strategy, attempting to capture corner and inner-edge positions as soon as possible by assigning them a high priority weight value. The example below is included to illustrate this general strategy employed by the GP evolved players.

The human is playing black (Black) and the computer is playing white (White). The game ends in a convincing victory for the computer. At the start of the game (Fig. 7) the corners and edges are considered least important and are assigned low weight values.

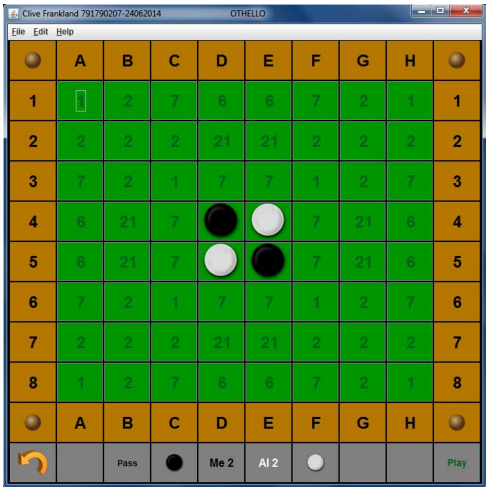

Fig. 7. Start game board heuristics

The surrounding central positions (7) will be selected as the first move. It is important to note that the inner board positions (21) are already considered high priority positions to be captured by White. After Black has made the first move the board status will be re-evaluated and new weight values assigned. The three strategy heuristics are assigned a value of 'true' early on in the game and remain unchanged throughout the rest of play. Towards the end of the start-game (Fig. 8) the corner positions become more important and higher weight values are assigned to them (5). The three positions adjacent and diagonal to the corner positions are considered 'bad' positions, not to be captured by the computer player, and are assigned very low weight values (1 and 3). As the play

converges towards the bottom left of the board the inner edge (7) and middle diagonal positions (8) are considered high priority positions and are assigned high weight values. This is crucial for White as capturing any of those three positions will eventually force black to capture positions directly next to the corner square allowing White to capture the corner and thus gaining a distinct advantage over Black.

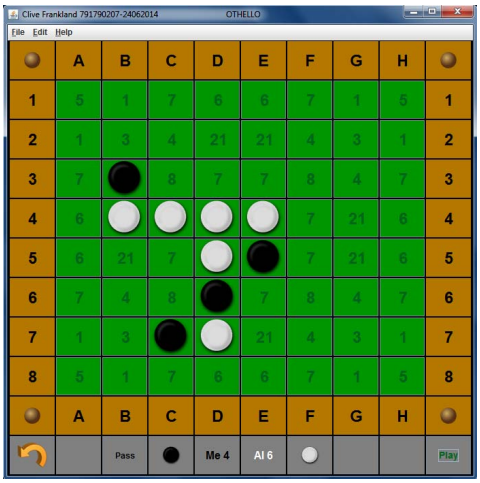

Fig. 8

White captures C8 (Fig. 9) fragmenting Black's defence. The inner edge (22) and outer diagonal (14) are now critical to White's defence and are assigned high priority weights.

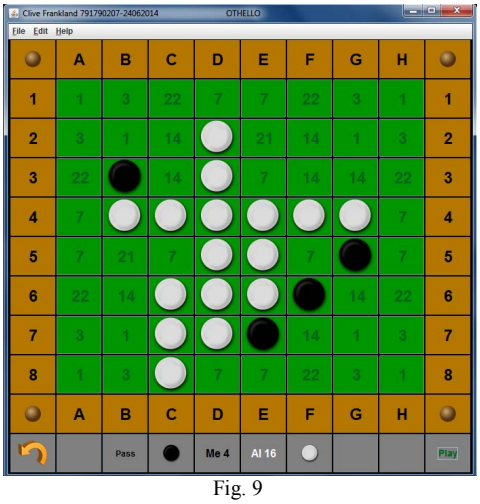

Fig. 9

Towards the end of the middle-game White has gained the upper hand (Fig. 10). Its corner and edge strategies allowed it to capture the bottom and top left corners (29) and effectively defend the positions (1) surrounding the corner squares. High priority is now given to the inner edge squares and they are assigned weight values of 29, equal to those of the corner positions. At the end-game White had secured the top left quad of the board as well as securing all four corners, leaving Black no option but to play the weaker board positions (Fig. 11 and Fig. 12). The computer player, White defeats Black, the human player by 39 to 25 (Fig. 13). White was able to strategically defend the corners and play edge tactics to the disadvantage of Black. White was able to fragment Black's defence through clustering and good middle-game tactics (Fig. 11 and Fig. 12). This illustration shows that GP is able to evolve sound Othello strategies.

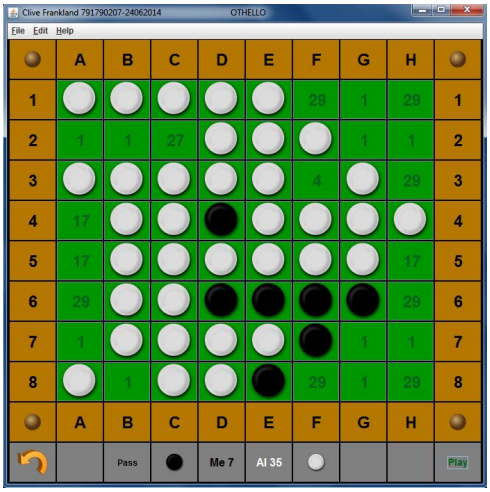

Fig. 10

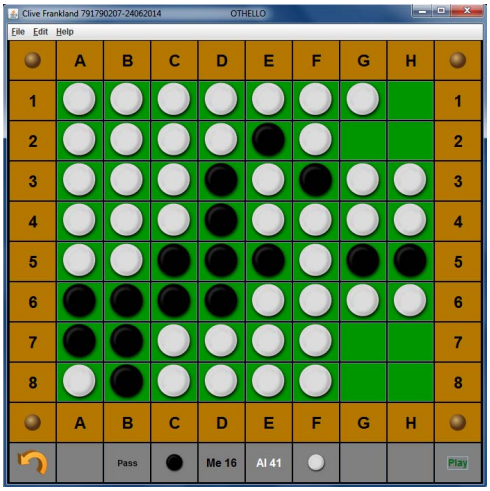

Fig. 11

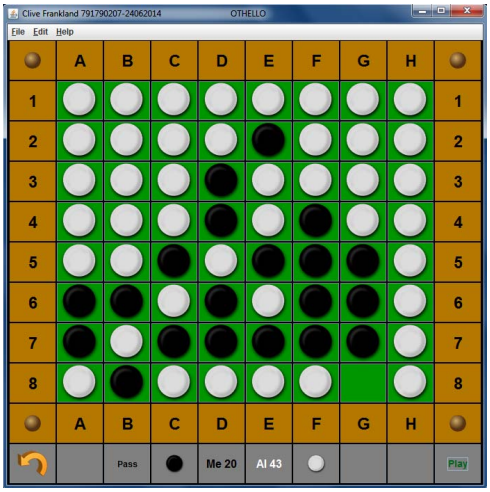

Fig. 12

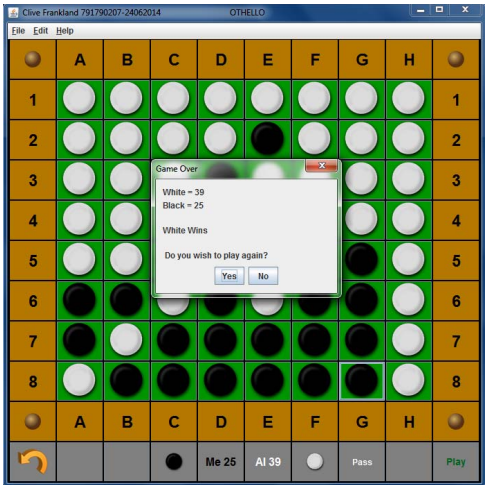

Fig. 13

VI. Conclusion and Future Work

The research presented in this paper proposes a novel approach for evolving game playing strategies for two player zero sum games using genetic programming. This work differs from previous research in that firstly the strategies are induced online during real time play. Secondly the approach is heuristic based and combines heuristics representing different strategic areas on the board. The GP approach was found to play well against players at the beginner to intermediate level but not as well at the hard or advanced level. This can be attributed to mobility not being incorporated into the game playing strategies. As a result the evolved players could not win against players that would look ahead 3 or more levels or in more challenging games at higher levels. Future work will study this further and examine how a measure of mobility can be included in the evolved game playing strategies. The study has also revealed the potential of the novel approach proposed and future work will also look at applying the approach to other board games such as checkers and chess.

References

[1] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[2] W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone, Genetic Programming: An Introduction, Morgan Kaufmann, 1997.

[3] A. Benbassat and M. Sipper, "Evolving lose-checkers players using genetic programming", in proceedings of the 2010 IEEE Symposium on Computational Intelligence and Games (CIG), 18-21 August 2010, Dublin, pp. 30-37.

[4] A. Benbassat and M. Sipper, "Evolving board-game players with genetic programming", in proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO '11), 2011, pp. 739-742.

[5] A. Hauptman and M. Sipper, "GP-EndChess: using genetic programming to evolve chess endgame players", in proceedings of the 8th European Conference on Genetic Programming (EuroGP '05), 2005, pp. 120-131.

[6] A. Hauptman and M. Sipper, "Emergence of complex strategies in the evolution of chess endgame players", Advances in Complex Systems, Vol. 10, 2007, pp. 35-59.

[7] Y. Azaria and Sipper, M., "GP-Gammon: using genetic programming to evolve backgammon players", Genetic Programming, Lecture Notes in Computer Science, Vol. 3447, pp. 132-142.

[8] A. Benbassat, Elyasaf, A. and M. Sipper, "More or less? two approaches to evolving game-playing strategies", Genetic Programming Theory and Practice X, 2013, pp. 171-185.

[9] A. Hauptman and M. Sipper, "EvoMCTS: a scalable approach for general game learning", IEEE Transactions on Computational Intelligence and AI in Games, Vol. 6, No. 4, December 2014, pp. 382-394.

[10] A. Benbassat and M. Sipper, "EvoMCTS:enhancing MCTS-based players through genetic programming", in proceedings of the 2013 IEEE Symposium on Computational Intelligence and Games (CIG),11-13 August 2013, Niagara Falls, pp. 1-8.

[11] A. Benbassat and M. Sipper, "Evolving both search and strategy for reversi players using genetic programming", 2012 IEEE Conference on Computational Intelligence and Games (CIG), 11-14 September 2012, Granada, pp. 47-54.

[12] P. Rosenbloom,"World-championship-level Othello program", Artificial Intelligence, Vol. 19, No. 3, 1982, pp. 279 – 320.

[13] A. Leouski, Learning of Position Value in the Game of Othello, Masters Dissertation, Department of Computer Science, University of Massachusetts, 1995.

[14] D. E. Moriarty and R. Miikkulainen, "Discovering complex Othello strategies through evolutionary neural networks", Connection Science, Vol. 7, pp. 195-209.

[15] S. Y. Chong, M.K. Tan and J.D. White, "Observing the evolution of neural networks learning to play the game of Othello", IEEE Transactions on Evolutionary Computation, Vol. 9, No. 13, 2005, pp. 240-251.