Machine Learning – Othello Project

Tom Barry

The assignment.

We have been provided with a genetic programming framework written in Java and an intelligent Othello player(“EDGAR”) as well a random player. The initial framework has 13 primitives including the simple operations of addition subtraction multiplication and division in addition to numeric data about board positions. This provides the basic tools to evolve board evaluation function to be used to create Othello players. One of the goals of the assignment is to “Compare, contrast and discover methods to approach Othello with GP”.

Motivation for my experiment.

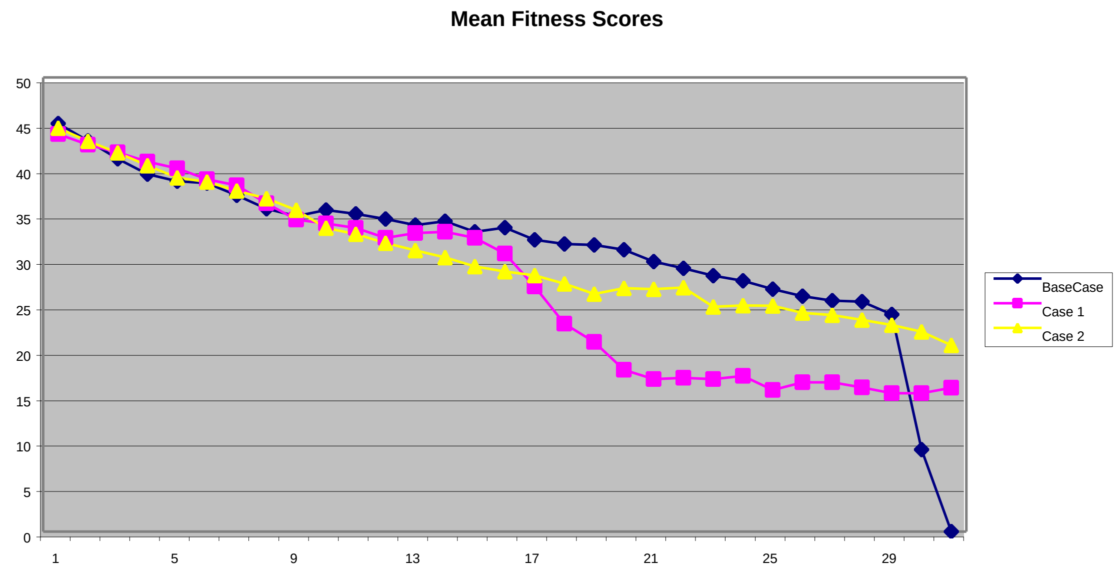
I was not familiar with Othello prior to this exercise. So I initially invested several hours being soundly defeated by EDGAR, the AI player provided with the assignment. A recurring theme in these games was my lack of alternatives towards the end of the game. This reminded me of a chess concept of “zugzwang” or forced moved. Although zugzwang can occur in the middle or end game it is most often associated with king and pawn endgames. The main thrust is that while a player’s position is acceptable as it is any move he makes significantly diminishes his position. The other observation I had and that was reinforced in the Evans/Schiffman paper is that the endgame of Othello is the more important than the opening. Evans/Schiffman used a player which was random for several moves and then began to use EDGAR. They found that EDGAR was often strong enough to compensate for the weak start.

The experiment.

In order to explore the issues above I setup generated populations using 3 sets of primitives.

- **Base Case**. These are the primitives provided with the assignment. They include the operators "+,-,*,/" for addition, subtraction, multiplication, and division, respectively. They also include the terminals: "white, black, white_edges, black_edges, white_corners, black_corners, white_near_corners, black_near_corners, and the integer 10".
- **Case 1**. In addition to the Base Case primitives two terminals, “black_available-moves ,white_available_moves”, were added. These terminals indicate the number of legal moves available to each color. It was hoped that this would permit the evaluation function to take zugzwang into account
- **Case 2.** In addition to the primitives in Case 1 the “move_number” terminal was added. This was calculated as the number of pieces on the board –4. This was created to provide a tool to measure the passage of time.

A population size of 400 was used and run for 30 generations. The probability of breeding was 50%. This rate was selected so that half of the existing population would be carried into next generation. The populations were trained against EDGAR. The lower the fitness score the better. The fitness score is equal to the number of the opponents pieces remaining at the end of the game. A fitness measure below 32 implies the new player won the game. A graph of the results is below.



It is worth commenting that the dramatic improvement in Base Case mean fitness was also accompanied a dramatic decline in diversity. In other words a few successful players have become dominant.

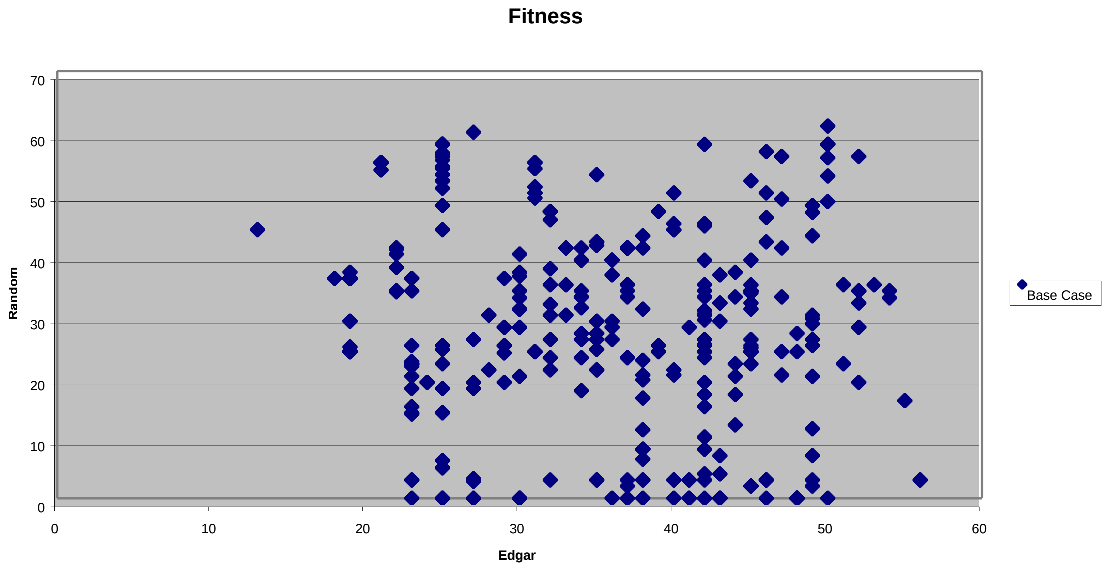
In order to see if the results were generalizable seven of the best unique players from population 0,5,15and 30 or 28 from each case. Unique, for this purpose, was defined as having one of the variables fitness, length or depth not match between two players.They each then played 25 games against a random player. The results were surprisingly bimodal the players won or lost more than 20 of the 25 games. The fitness results from these tests are on an equivalent basis with the EDGAR results.

	Fitness		Number of Players winning more than 20 Random Games
	EDGAR	Random	
Base Case	16.1	27.8	19
Case 1	18.3	32.3	13
Case 2	20.4	33.7	9

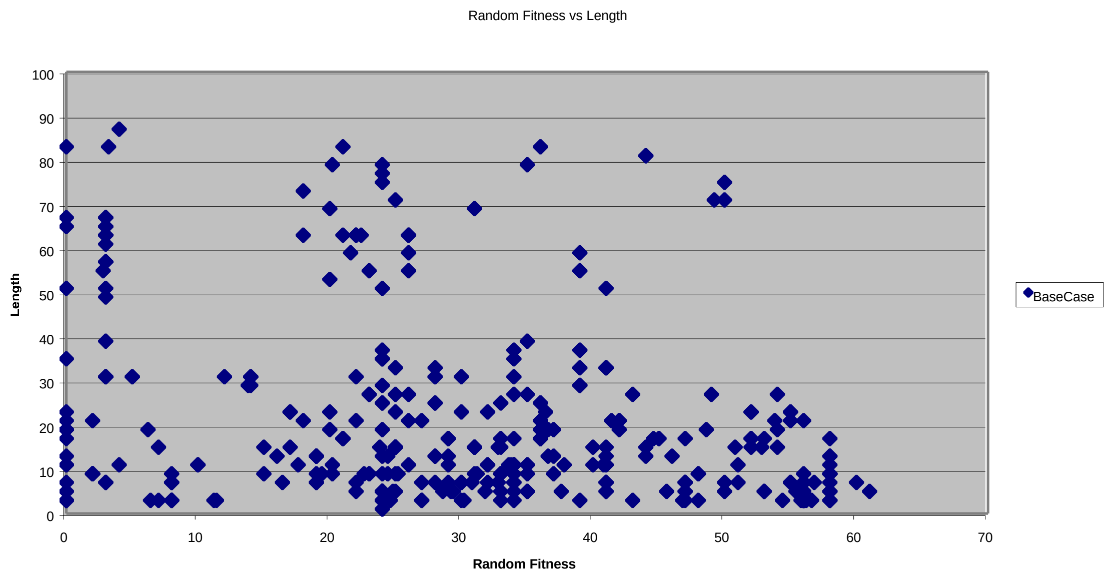
There are several interesting observations.

- Beating EDGAR did not assure victory against a random player. Many EDGAR savants were created.
- Better performance against EDGAR did indicate a higher probability of defeating the random player. In the Base Case 19 of the 28 players defeated the random player only 9 of the Case 2 players did as well.
- The additional primitives seemed to diminish performance against the random player.

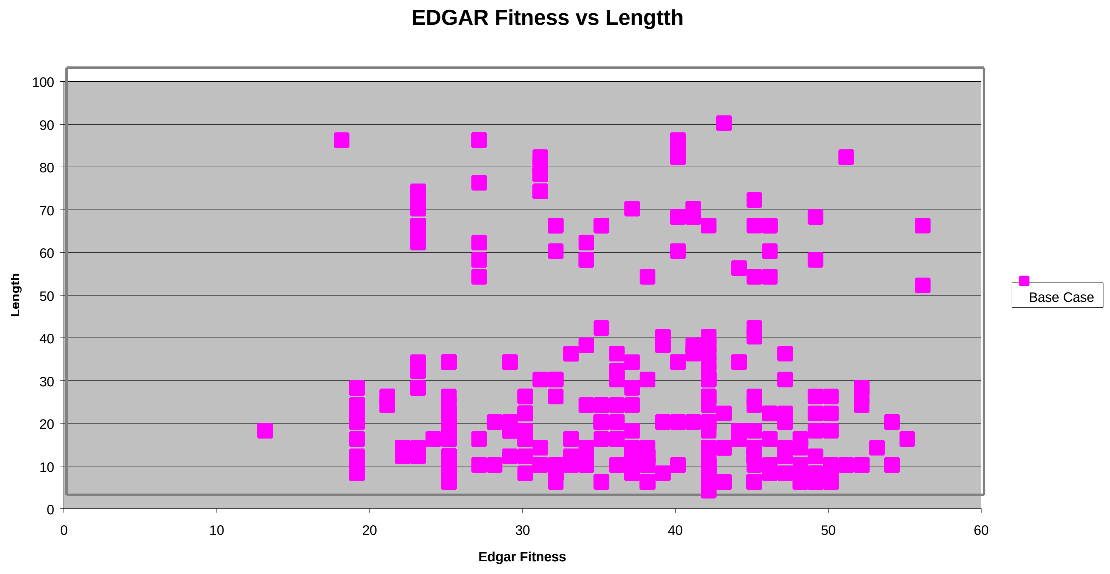
In order to achieve some insights into the broader population I examined the 10th generation of each of the 3 cases. Each of the 400 members of that generation played 5 games against the random player. Results were similar across the three cases so I have selected the Base Case for these graphs.



The fitness measure for EDGAR is on the x axis . Since this was the fitness measure used for training you can see a reasonable amount of concentration. No effort was made to eliminate duplicates. As you can see from the vertical lines players with the same capability against EDGAR varied performance against the random players. Those players in the area bounded by 32 on both axis are the ones who beat both players.



The above graph compares the fitness against the random player(x-axis) and the length of the player. Although there is not a substantial bias it does appear that a longer player is more likely to defeat the random player. This is indicated by the higher density in the upper left quadrant as compared to the upper right. But there does not appear to be a preference for short string over long.



As you would expect EDGAR is a more difficult competitor and so there is a certain scarcity at the left hand portion of the graph. It does not appear that there is any relationship between the length of the string and the likelihood of success against EDGAR.

Conclusion

It was somewhat disappointing that the additional primitives added no apparent value. It was, however, striking that players which could defeat EDGAR would fair so badly against a random player. The cautionary lesson to be learned is that if you wish to achieve generalization you must make sure that your training technique is varied. But there is good news as well. If faced with a complicated but very specific problem not requiring generalization genetic algorithms may be a very effective approach even without extended diverse training