CHAPTER 5
CONCLUSIONS AND FINDINGS

5.1 INTRODUCTION:

In this chapter, the conclusion derived from the observations and results collected in chapter 4 by various forms of experiments on test bed games is discussed. The conclusions are based on the results collected for each of the test bed games namely Go-Moku and Othello.

5.2 CONTRIBUTION:

The contribution of the research work in the game of Go-Moku and Othello is as under.

5.2.1 Go-Moku Conclusion:

There are many game playing programs available for the game of Go-Moku. The purpose of research was to apply genetic algorithm in the game so as to achieve optimization in the play. For any genetic algorithm to be successful it is very much important to develop an efficient fitness function. Most of the work was devoted to development of fitness function and the weights associated with various types of threats as they develop as the game progresses.

All the experiments were carried out for a board size of 15* 15. For a population size of 500, the computer player moves tries to maximize the value of fitness function as the game progresses. At some intermediate moves the fitness value is not good, which shows selection of average move and not the best move. As we increase the size of population, the algorithm behaves in way to maximize the fitness value and better moves are selected. So, we can say that the quality of play increases as the number of chromosomes in a population increases.
The simplicity of fitness function is heavily based on the feature characteristics of the game. The analysis and construction of features is the main driving force to solve the game in terms of creating fitness function. This function when passes through the genetic cycle of selection-crossover-mutation with weight tuning through iterative process of generations it exposes a possibility of improvement and some rearrangement of weights to produce brilliant moves for attack and defense strategies. I also verified that the used weights in fitness function influence strongly the quality of moves by computer player.

The genetic algorithms a have a tendency of giving slow response. The average time required for a move in population size of 500 is 0.76 sec, while that for population size of 800 is 1.92 sec, and that for population size of 1000 is 2.94 sec. From this we can rule out the population size of 1000 with fixed value of other parameters. While the population size of 500 or 800 is reasonable as far as time is concerned.

The most important consideration is the final result analysis in terms of win and loss of the game. 74 % of games won by computer player for population size of 500, while in the case of population size of 800, the computer player won 74% of the games, and in the case of population size of 1000, the computer player won 82% of the games. So, the computer player plays better for a population size of 1000.

Considering all the three parameters, namely fitness function evolution, time required making a move and the final result analysis, I conclude that under the practical considerations, it is advisable to use a population size of 500 with other parameters of genetic algorithms as mentioned in table 3.3, chapter 3. As in that configuration, the genetic algorithm player won 74% of games, which in my opinion is very good. Our game of Go-Moku for a population size of 500 wins 74% of the played games with average time for a move less than a second, while the game developed by Marco Kunze and Sebastian Nowozin (which uses alpha-beta cut off and DB search) wins 72% of the games played, but it takes on average more than 5 seconds for a move, while our program using genetic algorithm takes considerably less time for making a which is a considerable improvement in terms of time efficiency, with slight improvement in winning percentage.
The main aspect of the research was the application of evolutionary approach such as genetic algorithm to the area of game playing and it has been applied successfully for the game of Go-Moku and increases the potentiality of better move by computer player. Previously, many people have worked on Go-Moku program with the application of threat space search, mini-max and alpha-beta cutoff algorithm for game tree search. The research work carried out in this thesis is novel in the sense that it uses genetic algorithm for the computer player successfully. The weights chosen in the formula of fitness function largely influence the quality of moves in the game. This implementation, which takes moderate number of Genetic Algorithm constituents like Number of Genes in Chromosome, Population size, Number of Generations, not only improves the working cycle of better game moves, but also show very promising side of Genetic move optimization. It is a positive sign of moderate effective path to be followed for board games. Board Games in general are very complex due to its inherent attributes of very high search space and complex move-search optimization. Genetic Algorithm is very good simulation tool to cultivate the qualities of medium level human-player.

5.2.2 Othello Conclusion:

For the Othello program, it uses the combination of genetic algorithm and alpha beta cutoff. The genetic operators are being used in each population formation of all generations. The crossover, survivor and mutation percentage rate was kept 90, 10 and 0.01 for all generations respectively. Machine learning has been applied in many computer board games and the application of genetic operators is fit to enhance an Othello game playing program. The program used game playing strategy of Othello with domain knowledge in the form of fitness function evaluation. The purpose of making simple evaluation function is served as the program evolves. The optimization of the genetic algorithm can improvise fitness functions in order to evaluate the board state precisely and improve the computer Othello.

The collected and analyzed results show slow and steady evolution of fitness function weights. This board game problem domain is one of such kind which can potentially be
Conclusions and Findings

solved not using conventional computer program development. But it needs novel branch like soft computing where effective learning can take place using genetic algorithm on one linear fitness function which takes board game features into consideration and selects bit string representing each board square family uniquely.

The weight changes in corner discs indicates that the weight range spectrum attains the value from the initial phase of the game which shows the stability feature of the corner disc as it proves that once the corner is occupied by any player that cannot be taken back by its opponent. For generation 8 to 12 the low values are indicative that the corner discs during these games were captured by the opponent. For generation 13 to 16 the steady growth speaks about positive solidarity of the corner discs and its contribution in these generations.

Weight value graph indicate that firmness support of discs sets which are adjacent to corner discs. It also shows that these discs also supports “firmness” of disc capturing but is not severe enough compared to set of corner discs. The weight values for generations 14 to 24 shows that the discs adjacent to corner disc are contributing positively to the winning side of the Reversi player program which derived evolutionary learning for the disc set.

The GAs enriches the authority of the board game-playing computer program by increasing the potentiality of better move selection. This results in providing a reasonable chance to play the game of Reversi more competently and meritoriously. After this real-world test on a specific board game, it proves that evolutionary learning through genetic algorithm requires optimization as its one of the vital area. It is possible to enhance the efficiency of learning greatly. The optimization of the genetic algorithm can improvise fitness functions in order to calculate the board state accurately and make significant progress to improvise the computer game of Reversi.