Emerging Soft Computing Methodology to Enrich Evaluation Function Weights Efficiency

Chirag S. Thaker
Research Scholar
Faculty of Engineering
SGVU, Jaipur, India

Dharm Singh
College of Technology & Engineering, MPUAT
Udaipur, India

S.M. Shah
Research Scholar
Faculty of Engineering
SGVU, Jaipur, India

ABSTRACT

The soft computing approach for gaming is different from the traditional one that exploits knowledge of the opening, middle, and endgame stages. It is aims to evolve efficiently some simple heuristics that can be created easily from the basic knowledge of the game. Integrating sphere knowledge into soft computation can enhance the performance of evolved algorithmic methodologies and quicken the learning of solution finding. In this paper, one of the major constituents of soft computing- genetic algorithm approach is employed to develop a game playing program for Reversi (Game of Othello).

Evaluation function based genetic game playing strategies are been used to implement than a single simple heuristic based one. Genetic parameters implemented using Reversi game based fitness function using min –max search algorithm is strategic combination focus of the paper. Experimental results show that the proposed method is promising for generating better strategies.Developing players programs for board games has been part of novel soft computing research arms for decades. Board games have precise, easily formalized rules that make them perfect modeling in a programming environment. In this paper focus is on full knowledge (perfect information), deterministic, zero-sum board games by inculcating genetic algorithm as better move making search optimization.

Keywords

Soft Computing, Reversing, Fitness Function, Genetic Algorithm, Genetic Weight.

1. INTRODUCTION

The idea of constructing computer programs modeled on the intelligent decision based on move making is motivational purpose for systems which exhibit acumen, learning aptitude and self-adaptation. The human brain has many highly desired features that are hard to imitate in conventional computer systems. Incremental systematic efforts are made to acquire increasingly sophisticated proficiencies over a span of definite time. These soft computing based systems have shown the tendency of constantly altering and improving. Simultaneously it always retains its truthfulness as a learning system. Such learning algorithms and systems are adaptive in nature and show elasticity to changes in its learning atmosphere, so that new practices and stimuli are incorporated into soft computing based

system without altering existing competences. It shows to leverage computing power to withstand loss of intermediate results and the ability to self-evolve and self-reorganize in such a way that it retains developing functionality. [1]

Since the beginning of the computer era, people specially computer experts and game researchers is keen to build a smart game program capable of defeating human experts. They have chartered many different approaches for different board games including neural networks for backgammon, special-purpose hardware called Deep Blue for chess, and the application of expert knowledge with relatively small computational power for checkers and Othello. Some of these approaches are branches of soft computing.

Most of these techniques exploit expert knowledge as main facet of learning as much as possible, such as the proper learning algorithm for training the evaluation function, game feature centric relevance factors for the evaluation, the weights evolution of the evaluation parameters, board game opening knowledge and an endgame database. Acquiring such knowledge requires multidimensional help and advice of game experts, computational power for processing the knowledge extracted, and a process of trial and error to find the best overall approach. Various soft computing branches like fuzzy logic, neural network and evolutionary algorithms help many programmers and players, to acquire expert knowledge which can be digitalized and be made accessible through various network based technologies like Internet, grid networks or in latest cloud networks. [2][3]

The human brain is highly versatile in its ability to learn diverse tasks and to develop abstract symbolic models which enable the living system to operate effectively in complex environments. The game playing programs tries to imitate them in its own limited functioning scopes. Such competences can be well explored in an important domains like board games of zero-sum, deterministic, full-knowledge, alternate move and two player. A game of research for this paper is Game of Reversi as shown in fig.1.

They are played on an NxN *board* for some given N. Here work is done on Reversi, also known as Othello, is a popular game with a rich exploration history. Though a board game played on an 8x8 board, it differs widely from other board games as it is a piece-placing game rather than a piece-moving game. In Reversi the number of pieces on the board increases as the game progresses on, rather than decreasing as it does in board games like Chess and Checkers. The number of moves in Reversi is limited by the board’s size, making it a limited move game. [4].

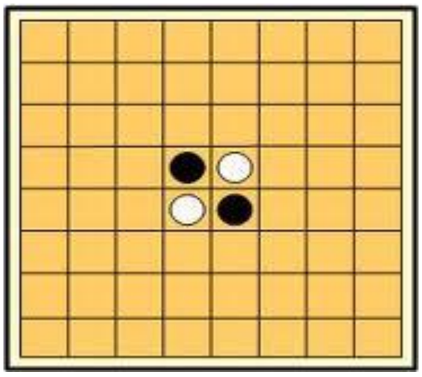

Fig.1 Game of Reversi Board

2. GENETIC ALGORITHMS ESSENTIALS

The Genetic Algorithms (GAs) are inspired in the Darwin's principles of evolution. Genetic algorithms are probabilistic algorithms that offer a parallel and adaptive search mechanism based on the principle of natural selection, survival of the fittest and reproduction. GAs uses a direct analogy of natural life behaviour. They work with a population of "individuals" called population embers, each representing a possible solution to a given problem. Each individual chromosome is assigned a "fitness score" according to how good a solution to the problem it is. The fittest individuals are given opportunities to "reproduce" and transmit their "good" features to the next generation. The least fit members of the population are less likely to get selected for reproduction process and so "die out" and most of the times are not carried to the next generation. GAs are able to "evolve" solutions to real world problems, if they have been suitably encoded. One of the innumerable applications of the GAs is to solve problems with a large search space and with characteristics that should be combined to look for the best solution. The utilization of GAs is really important and most suited to deal with the problem found in this work. [5][6]

Genetic algorithm approach as a branch of evolutionary computational systems were developed in the 1960s and 1970s as optimization tools to solve engineering problems, with early applications including the optimization of real-valued parameters for airfoils. The idea behind these early systems was to evolve a small population of candidate solutions to a given problem by applying operators inspired by biological evolution, particularly mutation and selection. Genetic algorithms (GAs) were first developed by John Holland in the 1960s as an abstraction of biological evolution, rather than as a tool to solve a specific problem. Holland's book Adaptation in Natural and Artificial Systems presented a sound theoretical foundation for the study of GAs as a method for moving from one population of chromosomes (candidate solutions) to another, using genetic operations such as selection, crossover (recombination), mutation, and inversion. Since then, GAs have been effectively applied to solve a wide selection of modern-day problems in the scientific and engineering communities. [7][8]

An important step in the application of GAs is the identification of a "fitness function", which is used to measure how close each chromosome comes to solving the problem at hand. The fitness function is also used to select those chromosomes that will

participate in the creation of offspring. Characteristics of the fitness function play a significant role in the behavior and success of the GA in finding a solution. Considerable research efforts have focused on the issue of epistasis, a characteristic of a fitness function in which the fitness of a chromosome depends on the interaction between gene values at different locations on the chromosome. Generally speaking, the more stagnant or static a fitness function, the more likely that the GA will prematurely converge to a local optimum, delaying its convergence to a solution. [9]

It is important and noteworthy to know that the GAs do not guarantee to find the best of the possible solutions for a problem, but they are generally good at finding acceptable solution in an acceptable time. This is a requirement of many problem domains involving very high search space. Before a GA can be run, a suitable coding (or representation) for the problem must be devised, it is required a fitness function, which assigns a merit to each encoded solution and it is important to define the selection and reproduction rules which are genetic operators. Each possible solution for a problem is represented by a set of parameters or genes. The genes are joined together to form a string of values or a chromosome. The most common representation is the binary string form as it is simple and easy to be manipulated by the genetic operators. [10][11]

The most traditional genetic operators are the crossover and the mutation. In the first case, two individual's chromosomes of the population are selected based on some selection criteria and their chromosome strings are cut at a randomly chosen position. Resultant two tail segments are then swapped over to generate two new full length chromosomes. The mutation operator is generally applied to each descendent individually after crossover. It randomly alters some genes with a small probability. Mutation is traditionally seen as a "background" operator; however, examples in nature show that asexual reproduction can evolve sophisticated creatures without crossover. [12]

To solve the board game described, it was used a binary representation with 10 bits where each bit (or each chromosome gene, in the language of GAs) represents one of the possible board square position family members which can be selected in the radial direction. When the value of the bit is 1, it means that the bit to which it refers is occupied by player's disc and when the value of the bit is 0, the disc is not selected or occupied.The fitness function provides the decidability. The greater the decidability values for a given distribution of points, the greater the fitness of the individual and the higher the probability of being chosen for reproduction.An individual is selected from the current population and then, two genes are randomly chosen. Therefore, those genes are swapped to produce a valid descendant. In order to ensure that the next population will have better individuals, that is, the populations members will converge to a better result, the offspring are joined with their "parents" and then, those fittest entities are selected to form the next generation. This procedure avoids the loss of an individual with high fitness. [13][14]

3. DETERMINISTICS BOARD GAMES

The main focus of traditional procedures for developing tactics for board games such as checkers and chess divide the game into various game stages like opening, middle, and endgame stages. For each stage, a different heuristic can be applied. For example, it is very problematic to define the most appropriate choice in the opening stage of a game, so the use of an opening book from games played by experts is beneficial. These opening games are to build very strong piece or disc formation strategies which can help to build good board capturing or board mobility feature in mid game. In the middle stage, a game tree with a limited depth is constructed and a game feature based heuristic evaluation function is applied to estimate the relevance of each move. Here efficient and optimized search algorithm and good evaluators are badly needed to build strong end game base. Finally, in the end game, the number of pieces (or possible moves) becomes reasonably small and deterministic calculation of the final moves is possible. [15]

3.1 Board Game Solution Hypothesis

In Reversi, the typical linear evaluation function and min-max search algorithm uses a computer program to search a game tree to find an optimal move at each play, but there are challenges in overcoming an expert's experience in the opening, middle, and endgame stages. Sometimes a computer Reversi program fails to defeat a human player because it makes a mistake that is not common among human expert players. Sometimes the error is discovered by the computer program after searching beyond the predefined depth of the game tree (a so-called "horizon effect"). To defeat the best human players, Reversi uses an opening game, middle game and end game strategies. Also, game of Reversi relies on expert knowledge that is captured in an evaluation function, which is used to find out next move by exploring and evaluating current possible moves in the stage. Reversi's success is based largely on traditional game theory mechanics (game tree and alpha-beta search) and expert knowledge (opening book, middle game, components in evaluation function and endgame). [16]

Recently, soft computing based evolutionary induction of game strategies have gained popularity because of the success reported in various board game programs. In games such as Othello, Go, Chess and Backgammon, the evolutionary soft computing approach has been applied to discover better game playing strategies. For games, it might take a long evolution time to create a world-level champion program without a predefined knowledge base. [17]

Incorporating a priori knowledge, such as expert knowledge, evaluation function and human move preferences and most importantly, domain knowledge discovered during evolutionary search, into evolutionary algorithms (EAs) has gained increasing interest in recent years. In this paper, soft computing approach is incorporated to propose a method for systematically inserting expert knowledge into evolutionary board game framework at the opening, middle and endgame stages.

3.2 Conventional Game Playing Phases

A game can usually be divided into three general phases: the opening, the middle game, and the endgame. Entering thousands of positions in published books into the program is a way of

creating an opening book. A problem with this approach is that the program will follow published play, which is usually familiar to the human players. Without using an opening book, some programs find many interesting opening moves that stymie a human quickly. However, they can be potential producer of fatal mistakes and may enter a losing board configuration quickly (within a span of 2 or 3 moves) because a deeper search would have been necessary to avoid the probable mistake. Human players certainly possess an advantage over computers in the opening stage because it is difficult to quantify the relevance of the board configuration at an early stage. To be more competitive from very early playing phase, an opening book can be very helpful but a huge opening book can make the program inflexible and without innovation. One of the important parts of game programming is to design the evaluation function which builds very good move making and disc positioning by capturing good positions for games like Reversi for the middle stage of the game. [18][19]

The evaluation function is often a linear combination of game features and board square weights. These features are based on human knowledge, such as the number of important disc positions, the number of free positions to make move(potential mobility), the piece differential between two players, the stable discs which can't be flipped by opponent (stable discs) and other pattern-based features. Determining these components and assigning weights to them requires expert knowledge and a long trial-and-error tuning. Attempts have been made to tune the weights of the evaluation function through various automated processes by using linear equations; the processes are fuzzy game theory, neural nets, and EAs. These can compete with hand-tuning weights in terms of time and efficiency. In Reversi, the results of the game can be calculated in real-time if the number of empty spaces is less than 26. Recently, the construction of a ten-piece evaluation function based on Reversi symmetry has been completed. [20]

4. REVERSI FUNCTION AND GAME EVALUATION

Reversi also known as Othello is a well-known and challenging game for human players. Reversi triggered the emergence of mobility strategies based evaluation functions to characterize different stages (opening game, mid-game, and end-play) in the game of Reversi and the corresponding static evaluation function for each stage was evolved using a genetic algorithm.The evaluation function proposes that output of the quality of each possible move at the current board configuration. The function along with min-max search in the game of Reversi at each move making level saw the updated board and gives the rank of each move and only a subset of these moves was explored in limited ply depth. [21]

The function identified several important principles of game play and used them as the basis for genetic evolving of populations.In this paper we are examining whether developmental programs can be evolved through genetic evolution to play Reversi. In Soft computing research building computer programs that play games has been considered a worthwhile objective. The idea of using a game tree of a certain depth and using a board evaluation function that allocates a numerical score according to how good a board position is for a genetic player. The method for determining best moves from these available moves is performed by min-max search algorithm using alpha beta pruning. The main task is to define and genetically refine a board evaluation function. [22]

After two computer players have played a game, the loser is replaced with a deterministic alternative of the winner by changing the weights for other set of discs or the determining features that were used, or by replacing features that had very low weight co efficient compared to other features. More recently, board evaluations functions for various games have been obtained through various evolutionary techniques have been used to adjust the weights in Games of Reversi, Go, Chess, and Checkers.Three are three major criticisms of these approaches.The first disapproval is in the use of a board evaluation function and the min-max algorithm. Such combined methods appear to bear little resemblance to the methods that human players use to play games well. Typically, human beings consider relatively few potential board positions and evaluate the favorability of these boards in a highly intuitive and heuristic manner. They usually learn during a game, indeed, this is how, generally humans learn to be good at any game. If learning is one major criteria of research then these approaches stands very little or no value. Because the research algorithms will not return a numerical value for the favorability of a board position or use min-max but merely indicate which piece to move and where. The second criticism is with those methods or algorithms that evolve weights to achieve a high standard of play through better move selection. For this there is no counterpart biological plausibility. First in case, natural evolution produces organisms which are gradually better suited to their environment and this is an enormously slow process. Second, fitness function weight evolving efforts will unavoidably become infeasible when size becomes very large.Third point is organisms learning happens in their lifetime and evolution is not involved into them. Evolution inducted in next pool of generation. An evolutionary approach is interested in finding the learning positions and wants to use evolutionary process to create the learning rules that ultimately construct a learning system. In this way the size of the genotype will be unrelated to the size and connectivity of the population member networks.As development of self-learning computational system design is required it is a better option to constructs a learning system with a method of automatic program evolution called Genetic algorithmic approach to discover better move making system. The population members are made up of bit strings which act as chromosomes that encode programs that represent various aspects of board squares. When the encoded genetic string programs plays from generation to generations (player) genotype are executed as they cause a computational bit strings to evolve that can play game of Reversi. The key idea here is that genetic string do gets evolved which has weights of potential board squares and their related fitness function which when executed build and continuously shape and change the string at run time.

5. REVERSI BOARD EXECUTION

Recent applications of Genetic Algorithms (GAs) in the search for better fitness weights found in min-max search objects have suggested that GA convergence time can be improved by adapting to fitness function components on the fly. In particular, it is known that if such fitness functions F have the following format

$$\mathbf{F = \Sigma\ (Wi * Fi)}$$

Wi where is a set of fitness functions weights associated with properties

- Fi that collectively define the search problem in terms of feature parameters of the Board Game (Reversi)

Board game performance can be improved by enhancing GA evolution by defining a related fitness function. [23]

The application theory of Genetic Algorithms (GAs) to "binary" search intensive problems has been well developed. It also provides a solid theoretical foundation for future research. This general class of GA search community highlights problems which are fertile areas for those researchers wishing to demonstrate the correctness of general theories. In this situation, the chromosomes (potential solutions) are expressed as fixed length binary strings. Reversi has a binary string of 10 bits which are based on symmetry feature of the board game. This ten discs set gets replicated throughout the Reversi board so their relative weight for each of the disc set is same across board. It is also known that any fitness function defined on binary strings can be evaluated by assigning fitness values to them and subsequently gets evolved over a span of defined set of generations. Thus productive research efforts have been exploited in this representation. Unfortunately, the actual representation for an arbitrary binary fitness function and their evolution is computationally expensive. Consequently, the alternative strategy of many researchers is to demonstrate the correctness of hypotheses by showing that the results hold. [24]

In particular, results show that multiple modifications of the coefficients improve convergence time over GA runs in which no coefficient changes are made; furthermore, results in show that just one special modification to the coefficients, early in the GA run, brings greater improvement in convergence times than those runs in which multiple coefficient modifications were applied. In this paper, we examine the effect of similar coefficient changes when applied to GA search problems in which the fitness function is defined on chromosomes that have the form of binary strings of fixed length. The program implementation algorithm works as follows:

1. Start with a randomly generated population of n chromosomes, all of fixed length. (10 bit string chromosome)

2. Calculate the fitness F(*c*) of each chromosome *c* in the population.

If a solution has been found, i.e. F(c) = T for some c, and predefined target T (Target T is Win in Reversi Game or no dics is empty condition), then stop. Otherwise,

4. Repeat the following steps until n offspring have been created:

a. Select some pair of chromosomes as parents for the creation of offspring. (Selection)

b. Recombine the genetic material from the two parents to form two offspring (Recombination)

5. Mutate p randomly selected genes on each of q randomly selected offspring (Mutation).

6. Replace all the chromosomes in the current population with the new offspring.

7. Go to step 2 for next Generation

In the experiments conducted in this paper, the 'fitness proportionate' selection method has chosen to apply for pairing parents. In addition, out of two traditional recombination operators: the 'single point crossover' and the 'uniform crossover', uniform crossover is used as crossover method keeping the disc family set representing disc position for each type in mind. [25]

Genetic Approach is to search for a chromosome that satisfies game features and their importance properties F1, F2, …, Fk. The fitness function could be constructed with k components as

$$F(x) = \Sigma\kappa\, Fi(x)$$

Where each Fi would represent a measure of fitness of a candidate chromosome regarding property weight Wi. Suppose further that a chromosome c is a solution if and only if Fi(c) = 1 for all i < k.

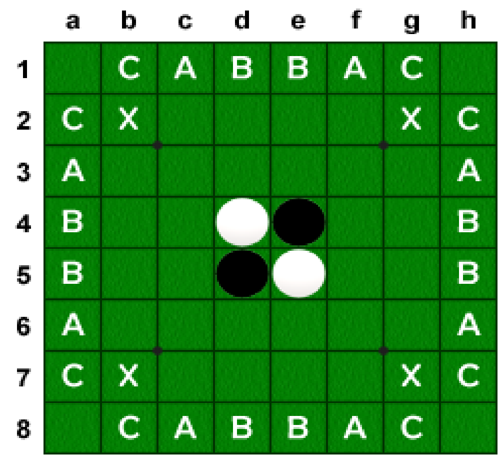

Fig.2 Different Game of Reversi Board Positions

A typical implementation of the GA would apply the fitness function F to each chromosome without regard to the function's composite characteristics. However, in a board game which all discs are same in terms of importance it is the board square position shown in Fig. 2 as A,B,C and X which would be expected that some property Fi is less selective (more easily satisfied) than others.

Consequently, chromosomes bit positions satisfying this property are given influence in the evolution of the population in early GA reiterations.

The paper presents more effective approach which has associated function is constructed with component 'coefficients', αi , as follows:

$$F'(x) = \Sigma\kappa\, \alpha i Fi(x) \quad (5)$$

- where αi > 0, Σκ αi = 1,
- where c is a solution if and only if Fi(c) = 1
- for all i < k (and consequently, F'(c) = 1).

Throughout iterations of the GA, the values of the coefficients αI can be manipulated in such a way as to control the influence of some of the less selective properties and their corresponding fitness function components.

6. RESULTS

The Reversi game program played twenty five games and each game was played for 50 game generations. Figure 3 shows the fitness weight values collected for 25 games.

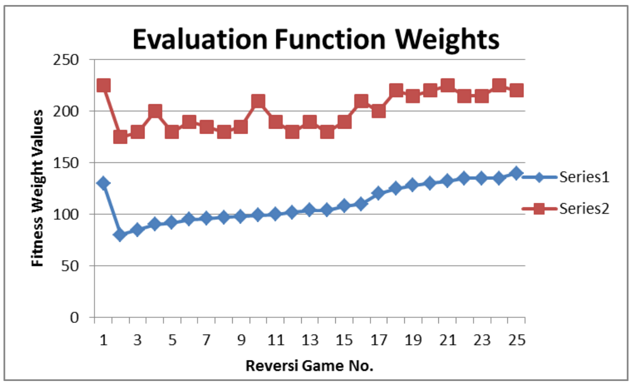

Fig.3 Reversi Game's Peak & Average Fitness Value Chart

Plotting of average fitness values is shown as series 1 and peak fitness value in each full span of game is shown as series 2 in the figure. Both these categories of values show rise or evolution of weight values. Series 1 shows consistent rise in weight values after initial learning drop in first two generations. Whereas peak fitness values have zigzag pattern of evolution but it overall shows positive increment after game no. 18 and attains new peaks as game progresses.

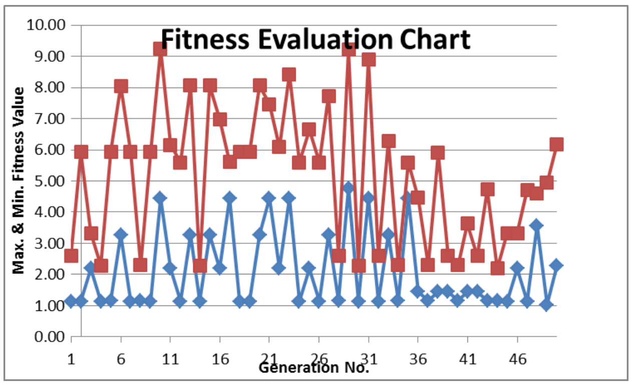

Fig.4 One Game Generation's Min. & Max. Fitness Values

Figure 4 shows Min. and Max. fitness values collected for one sample Reversi game which has 50 generations. For all generations the Maximum and Minimum fitness values shows very turbulent fitness values as they are varying in a larger range of values. This proves that fitness evolution genetic approach passes through tough phase of each generation in every individual Reversi game.

7. CONCLUSIONS

The game of Reversi is perceived and executed using soft computing approach as genetic algorithm as its significant branch to understand learning. The above mentioned 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 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 take board game features into consideration and selects bit string representing each board square family uniquely. The collected results very possibly prove and re affirm the researchers' belief that soft computing not only enhances the effectiveness of learning momentously but it also proves that genetic approach can improvise the board game learning progress further.

8. REFERENCES

[1] S. Chong, D. Ku, H. Lim, M. Tan, and J. White. Evolved neural networks learning othello strategies. In Evolutionary Computation, 2003. CEC ’03. The 2003 Congress on, volume 3, pages 2222 – 2229 Vol.3, December 2003.

[2] Daugman J. G., “How iris recognition works”. Proceedings of 2002 International Conference on Image Processing, Vol. 1,2002.

[3] S. Schiffel and M. Thielscher. A multiagent semantics for the game description language. In Proc. of the Int.’l Conf. on Agents and Artificial Intelligence, Porto 2009. Springer LNCS.

[4] T. Srinivasan, P.J.S. Srikanth, K. Praveen and L. Harish Subramaniam, “AI Game Playing Approach for Fast Processor Allocation in Hypercube Systems using Veitch diagram (AIPA)”, IADIS International Conference on Applied Computing 2005, vol. 1, Feb. 2005, pp. 65-72.

[5] M. Hlynka and J. Schaeffer. Automatic generation of search engines. In Advances in Computer Games, pages 23–38, 2006.

[6] Rosenbloom, P. (1982). A world championship level Othello program. Artificial Intelligence, 19:279-320.

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

[8] Holland, J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975

[9] Hong, J.-H. and Cho, S.-B. (2004). Evolution of emergent behaviors for shooting game characters in robocode. In Evolutionary Computation, 2004. CEC2004. Congress on Evolutionary Computation, volume 1, pages 634–638, Piscataway, NJ. IEEE.

[10] S. Luke. Code growth is not caused by introns. In D. Whitley, editor, Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, pages 228–235, Las Vegas, Nevada, USA, July 2000.

[11] T. P. Runarsson and S. M. Lucas. Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board Go. IEEE Transactions on Evolutionary Computation, 9(6):628–640, 2005.

[12] J. Schaeffer, N. Burch, Y. Bjornsson, A. Kishimoto, M. Muller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. Science, 317(5844):1518–1522, 2007.

[13] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – a survey. IEEE Trans. Evolutionary Computation, 9(3):303–317, 2005.

[14] Y. Jin and B. Sendhoff. Tradeoff between performance and robustness: An evolutionary multiobjective approach. In Proc. Evolutionary Multi-Criterion Optimization, LNCS 2632, pages 237–251, 2003.

[15] Kargupta, H., and B. H. Park. Gene expression and fast construction of distributed evolutionary representation, Evolutionary Computation, pp 43-69, 2001.

[16] Linton, R.C. Policies for Managing Composite, Epistatic Fitness Functions in Genetic Algorithms,” Proceedings of the 40th Annual ACM Southeast Conference, pp 23-30, ACM Press, 2002.

[17] McClure, M., and R. C. Linton, Proper Timing of Fitness Function Adaptation in Genetic Algorithms, Proceedings of the 41st Annual Southeast ACM Conference, pp 251-256, ACM Press, 2003.

[18] Mitchell, M. An Introduction to Genetic Algorithms, Cambridge, MA:MIT Press, 1996.

[19] M. Muller, “Computer Go,” Artificial Intelligence, vol. 134, pp. 145–179, 2002.

[20] Naudts, B., Suys D., and Verschoren A. Epistasis as a basic concept in formal landscape analysis. Proceedings of the 7th International Conference on Genetic Algorithms, 1997.

[21] Lee, K. -F., and Mahajan, S. (1990). The development of a world class Othello program. Artificial Intelligence, 43:21-36.

[22] Matt Gilgenbach. Fun game AI design for beginners. In Steve Rabin, editor, AI Game Programming Wisdom 3, 2006.

[23] Singh Dharm, Thaker Chirag S and Shah Sanjay M. Fitness Value Optimization for Disc Set in Board Game Through Evolutionary Learning in IJCA 2011:3728/encc/017 IJCA Special Issue on “Evolution in Networks and Computer Commumnication” ISBN 978-93-80864-98-7.

[24] O. David-Tabibi, A. Felner, and N.S. Netanyahu. Blockage detection in pawn endings. Computers and Games CG 2004, eds. H.J. van den Herik, Y. Bjornsson, and N.S. Netanyahu, pages 187–201. Springer-Verlag, 2006.

[25] J¨org Denzinger, Kevin Loose, Darryl Gates, and John Buchanan. Dealing with parameterized actions in behavior testing of commercial computer games. In Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG), pages 37–43, 2005.