Special Issue of International Journal of Computer Applications (0975 – 8887)
on Optimization and On-chip Communication, No.4. Feb.2012, www.ijcaonline.org

Functional Buildup in Board Game Positional Composition through Evolutionary Genetic Mechanism

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

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

J. S. Shah
Computer Engg. Dept.,
L. D. College of Engineering,
Ahmedabad, India.

ABSTRACT

Board games are very simple games and easy to learn. It has simple rules to move dice or discs. Though they are simple to learn, differences in experiences, skills and strategies make master-level players and naive players. To teach these properties to machine is a daunting task. Researchers attempt to develop evolutionary game player who like humans needs time to start the board game and will improve its performance at each passing game. Evolutionary algorithms simulate this learning procedure and genetic approach helps to find diverse fitter players. With respect to checkers, the evolutionary algorithm was able to discover genetic algorithm that can be used to optimize the move selection in play to near-expert level. Evolutionary approach develops machine player that generates solutions which does not often dominates in the last generation. In real world board game problems, diversity is very useful which can be attained by having a number of machine learning algorithms.This paper highlights, evolutionary genetic algorithm to improve the diversity of a population. From the last generation, representative checkers player fitness values are chosen to carry them to next generation from each species(population member) and combine them in current generation to play the checkers game. There are many high fitness solutions in a search space. Fitness selection techniques can find diverse strategies that survive in genetic search. In this paper, diverse evolutionary checkers players found by such techniques are combined. The evolved move of game player is compared with the fittest player evolved using a simple evolutionary algorithm.

Keywords

Board Game, Evolutionary Algorithm, Game of Checkers Evaluation Function, Genetic Algorithm.

1. INTRODUCTION

Since centuries games serve the main goal of food for brain development across all ancient civilizations. Game playing skill is an important fascination domain for people across all generations and geographic boundaries. It has one unique nature of intellectual challenge and satisfaction which is derived from playing well. Many board games have too many possibilities to explore for a human to understand before making any valid–sharp move at any given stage of the game. Since last five decades due to enormous efforts put in by budding researchers and game professionals in the area of computer games, many

powerful learning methods have evolved to explore and use ‘knowledge’ and ‘search’ methodologies to make game playing decisions on the board, thus the ‘evolved’ human or computer “player” with the best game playing program or algorithm wins in the long run. By experience it is noted that without perfect board rules, knowledge and decision making skill, mistakes are made and which ultimately results in game loss for experts also.[1] Since dawn of AI research evolution, many expert researchers have concluded that the first and foremost very easy path to achieve result driven high performance in game playing is nothing but to imitate the human approach. As humans are comfortably playing majority of board game since centuries in various minutely modified several versions. These versions vary in terms of board configuration, playing rules and game strategies. Though attaining human expertize is weighed down with difficulty of acquiring, interpreting and encoding human knowledge. It is learnt that human-like game playing or “intelligent” move making strategies are not necessarily the best computational strategies in terms of implementing and collecting results.[2][3] To make any intelligent board game playing programs, there is a lot to “learn” and “understand” through algorithms and programs while implementing any problem domain which requires “knowledge” and “decision making process”. In fact, the realization that a search-intensive (“brute-force”) approach is needed is result of where one such leading contributions of applying AI expertise to develop game-playing programs. Such developments can pave new path which has potential of producing high-quality performance using minimal domain specific knowledge. Due to consistent efforts made by AI research groups all over the world, very powerful and result providing search techniques and algorithms have been developed and successfully deployed to variety of problems areas like optimization, planning, and bioinformatics.It is very much evident that the application domain of game learning and move making programs are primarily an optimization problem. Where each program or algorithm module carries out a search, with a consideration that current board position serves as root node(s) for a search tree which provides next moves with due consideration of search depth. Every layer or level in game tree adds one depth to it. The degree of sophistication is well measured by gauging efficiency of search algorithms which in turn evaluates current power of its evaluation function in very large search space. The game playing programs have two major search centric dimensions consideration. [4]

- Decision complexity, the difficulty of making correct move decisions
- Space complexity, the size of the search space.

Game of Checkers is considered to have high decision complexity as it requires extensive skill to make strong move choices and moderate space complexity. [5]

1.1 Game Introduction

Opening board in a Checkers game is shown in Figure 1. Checkers board has 8 columns and 8 rows and each player has 12 pieces (Fig.1).Each player can move forward diagonally one square at a time. If possible, jumping over an opposing player into an empty square is allowed. In this case, opposing player is captured. When a player advances to the last row of the board, it becomes a king who can move forward or backward diagonally. If there is no available player or movable player, the game is over. A draw may be declared upon mutual agreement of the players or in tournament play at the discretion of a third party under certain circumstances. [6][7]

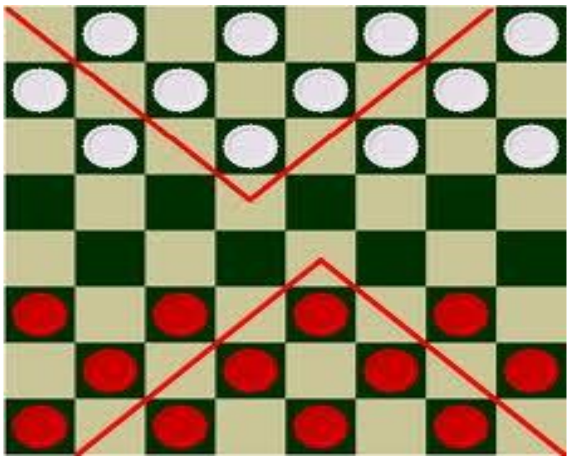
Fig 1: The Initial Board of Game of Checkers

1.2 Game Complexity

The game of checkers has roughly 500 billion- billion possible positions (5 × 1020). So task itself is very daunting to solve the game keeping the space complexity in mind. It determines the finishing result in a game with no error made by either of the player. Many researchers in Game of Checkers are applying state-of-the-art soft computing based techniques to improve the learning process. Game of Checkers represents one of its kind of most computationally challenging game to be solved to date. Evolutionary Learning challenges in Game of Checkers are:

(1) The space to be searched is huge. It is estimated that there are up to 5 × 1020 possible positions that can be searched. So any search algorithm based methods that aim to have exhaustive search for the problem space is infeasible.

(2) The search space bulk is not smooth and straight forward. Game of Checkers evaluation function's parameters which are game feature centric and their board construction are highly inter-dependent. Some typical cases give us worse performance when the values of optimization parameters are increased. On

the other hand the controlled set of evolutionary parameter increases performance through which an improved overall performance could be obtained. [8]

(3) The problem is not completely well understood by researchers. Even though all top performing Checkers playing programs parameters are formulated – derived and hand tuned by their program designers in close consultation with Checkers champions, finding the best possible 'move' value for each parameter is mostly based on operational genetic alternatives.

2. EVOLUTIONARY ALGORITHM

Evolutionary algorithm is very long-standing approach to machine learning program domain. It offers the potential for exploring any possible solution corridor presented by a computational compound problem. It has a very distinct feature of not having any aim to complete an exhaustive search, but it has power to quickly identify and converge on useful proximate solutions. Thus it provides an effective solution dimension for going beyond the structured conventional engineering approach which is very common to many forms of human design. The resolution search process is inherently parallel and can be accelerated significantly by utilizing an evolutionary search algorithmic to find fitter and better solutions through strong moves. Almost at every game tree or solution stage all that is required is to be able to compare two solutions and indicate which is better. [9]

Evolutionary algorithms explore a large number of points simultaneously in a search space at every game stage. This phenomenon eludes the chances of poor local optima quickly, which results in a quicker and fruitful search. It also makes fruitful search faster and better which helps in finding the better move in accordance with space and time complexity parameters.In evolutionary algorithm, the proposed solution does not aim to find global optimum. Evolutionary program's main objective lies in tuning an evaluation function to adjust its parameters so that the overall game performance of the program is enhanced. For a large set of high complexity board games' problems, a unique global optimum does not exist. So automatic tuning of the evaluation function appears like the much sought for optimization assignment. Such solution philosophy is very well suited for Evolutionary algorithms like GA. The many game feature centric parameters associated with the board game become basis for the evaluation function.(fig 2) They are basically nothing but the mirroring of the playing strategies correlated features associated with the game. They can be encoded as a bit-string. [10]

This initial set of bit-string (initial population) can be randomly initialized in form of "chromosomes". Each one representing one evaluation functions for the generations or a set of generations. This very first population is evolved through many generations until a highly tuned "fit" evaluation function gets emerged. This evolutionary process has one major obstacle in the form of the fitness function that deters the application of GA For a given a set of evaluation function derived parameters, encoded as a chromosome, the foremost objective is to calculate the fitness value. For many initial research years, firsthand solution was to let the individuals in each generation get allowed to play against each other a series of program or algorithmic operations. Then subsequently, take the fitness score of each individual in each of the generations. The main deficiency of this approach is the unacceptably large amount of time needed to evolve each generation. Thus it imposes severe limitations on the length of the iterations played after each generation, and also on the size of the population involved. [11][12]

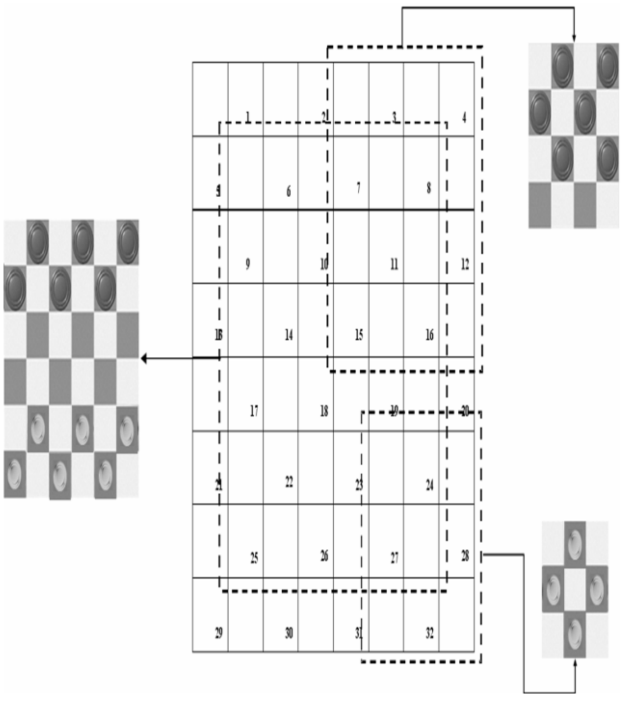

Fig.2 Checkers Game Centric Board Features

3. EVOLUTIONARY CHECKERS

Game of Checkers relies on features that were very well chosen using human expertise and weighted by evaluation function tuning. It has its own dependence on volume of game moves' database for all possible exposable moves explored through extensive game board possibilities. It is believed, once the best move exploration is reached, Computer Checkers will never make a mistake because the final outcome can be achieved from these states have already been evaluated. Game of Checkers aim to improve its play through better game rule understanding and perfect board positions information. It helps high-speed evolutionary computation to look ahead as many ply (search depth) as possible. [13]

The evolved evolutionary program exhibits a very stimulating flexibility that can be achieved with Checkers playing program which is heavily dependent on all of their "intelligence" being pre-programmed. One major adverse advantage is that evolutionary algorithm is also capable of adapting its game play to meet more demanding challenges from better-quality Fig. 3 opponents. It can also conceive new and untraditional procedures. [14]

By coding a given Checkers game playing problem algorithm into an Evolutionary computation framework, these clever algorithms are able to "evolve" solutions to real world problems. The game of checkers has space complexity of roughly 500 billion-billion possible positions (5×1020).The task itself is very daunting to solve the game which determines the finishing result in a game with no error made by either of the player. Since many years as long as thirty years, almost relentlessly, dozens of high end computing systems have been working on solving Game of Checkers by applying contemporary soft computing based techniques to improve the learning process. [15]

4. GAME IMPLEMENTATION THEMES

These basically describe the idea and the implementation in evolutionary checkers. The game passes through many stages. The goal of this approach is to improve the performance of the game in each of the stages. Evolutionary process helps in building better Checkers program which helps in playing and maximizing winning return in every move of each stage. Genetic algorithm is used for the speciation of population and it is easily implemented for a specified number of generations. A min-max algorithm with alpha beta pruning is used to implement to find the associated fitness values. [16]

4.1 Game Tree

To find the next move of a player, a game tree is constructed with a limited depth. Each node in a game tree represents the configuration of the board at some stage of the game. The quality of the terminal nodes (leaf nodes) in terms of fitness weight is measured with the evaluator like min max algorithm. The evaluated values using the evaluation function of the terminal nodes propagate upward using min/max operations. The max operation chooses the max value of all children nodes and the min operation chooses the min value among all the children at their level. The current configuration of the board is represented as a root node and the arc represents a move. At an odd number level, the max operation is used and vice versa. [17]

4.2 Evaluation Function Configuration

Generally, an evaluation function is the linear sum of the values of relevant features selected by experts. The input of the evaluation function is the bit value derived from the current configuration of the board and the output of the function is a value of quality. Evaluation function can be designed by two approaches. Either manually that requires game expertise along with tedious process of trial-and-error tuning the coefficient weights of the board square values and their functional co-efficient. These co efficient are derived from the positional significant of the board squares. These values are key values in determining some features of the board evaluation function, that can be modeled using various machine learning techniques such as automata, neural networks, and genetic algorithm. The board games are very important problem domain for learning the evaluation function such as determining the architecture of the board game model and transformation of the board configuration into numerical form.[18][19]

4.3 Evolutionary Genetic Algorithm

A collaborative theory of evolutionary genetic algorithm can perform better than the single and individual best school of thought. In the Game playing research community nowadays, on-line matches and tournaments between a professional player and a number of amateur players are interesting events.[20] Each move of the inexpert players can have voting option for selection. It is very natural for a professional player to defeat an inexpert player in a one-to-one match. However, the combination of opinions of multiple players can be as powerful as single professional player. It picks the move that has the greatest number of votes. If there is no clear winner, one of the moves that have the greatest votes is selected randomly. [21][22]

In initial play of the program, a tournament-style league is conducted to select the best player in generations. The moves are selected on the fitness values and games are played accordingly generations by generations. The values tend to get better as better as 'good' moves have the tendency to give better moves- winning moves. Good individuals are kept as for next generations and select-crossover-mutate genetic cycle functions in its totality and gives genetically evolutionary values. [23][24]

5. GAME EXECUTION

The estimated quality of the board is calculated using the co efficient weights of the board squares to evaluate the leaf nodes of the tree Games. This attracts considerable interest from AI researchers. The field of evolutionary algorithms is no exception to this rule. Over the years many games have been tackled with the evolutionary approach. A GA with genetic string representing board game features and having evaluation weights as their co-efficient in evaluation function is been used for the game of Checkers playing program resulted in a competent player program that employed sophisticated mobility play. [25][26]

5.1 Chromosome Structure

The genetic representation is initialized as a sequence of 32 genes, one per occupied board square (Fig. 3). Each gene is a real value in the range [0, 1]. The gene is decoded into a move by multiplying the real value by the total number of moves based calculated evaluation value. This is based on "goodness" of board square which gets calculated on basis of game features' significance, calculated min-max algorithm value and current status values available. This is used as an index for a single move from the list of available moves. Although this prototype with future moves dependent on the current move choice (the number of available moves at each node in the game-tree will vary significantly), it serves to examine whether the co-evolutionary approach can produce useful game-play. [27][28]

After each move is made, the first gene is cropped from all chromosomes, resulting in a chromosome structure that shortens as the game progresses.

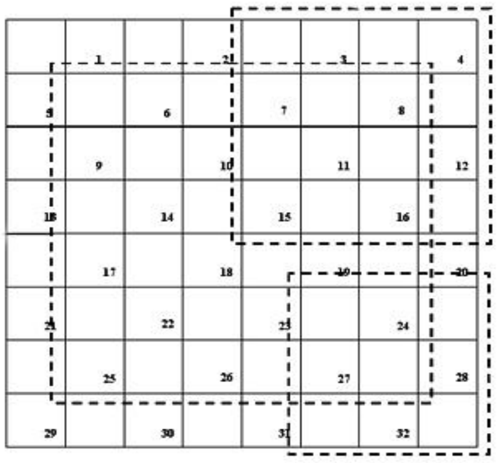

Fig 3: Board Squares of Game of Checkers

6. RESULTS

The results are collected for 50 game generations of Games for Checkers game as shown in figure 4. Evolutionary weight behavior is visible even with this relatively small population size and chromosome size also. The board square fitness values maximum (series 2) and minimum (series 1) for each game generation shows the evolutionary rise as the game generation progresses. The fitness value gains for maximum values are high and increasing. Whereas, minimum fitness for board squares are either steadily low or decreasing. This can be interpreted as evolving behavior of Genetic approach on Checkers learning.

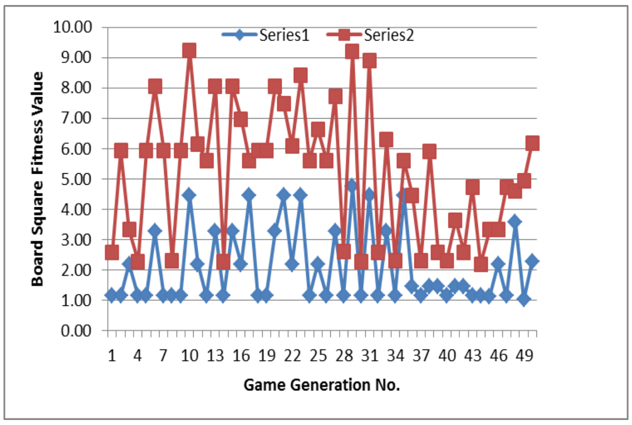

Fig 4: Checkers Collected Fitness for 50 Game Generations

7. CONCLUSIONS

The game of Checkers is conceived and implemented using evolutionary themes and genetic string manner. Based on the collected and analyzed results, the paper concludes that Genetically Implemented Evolutionary algorithm for a board game like Checkers enhances the power of the board game-playing computer program by grasping the potentiality of better board square move selection. This results in providing a reasonable chance to play the game of Checkers more proficiently and worthy.

This board game domain implementation shows a novel path of evolutionary learning through genetic algorithm for other group of problem domains also where optimization is one important domain to arrive at a better solution. Evolutionary algorithm enhances the effectiveness of learning momentously. The genetic optimization can be improvised further by devising better fitness function drafting techniques in order to calculate the board state fitness more precisely and make noteworthy progress to improvise the computer board games.

8. REFERENCES

[1] D. B. Fogel, "Evolutionary entertainment with intelligent agents," IEEE Comput., vol. 36, no. 6, pp. 106–108, Jun. 2003.

[2] P. Godefroid and S. Khurshid, "Exploring very large state spaces using genetic algorithms," Int. J. Softw. Tools Technol. Transf., vol. 6, no. 2, pp. 117–127, 2004.

[3] E. Alba, F. Chicano, M. Ferreira, and J. A. Gmez-Pulido,"Finding deadlocks in large concurrent Java programs using genetic algorithms," in Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08). ACM, 2008, pp. 1735–1742.

[4] E. Alba and F. Chicano, "Ant colony optimization for model checking," in Proceedings of the 11th International Conference

[5] on Computer Aided Systems Theory (EUROCAST 2007), vol. 4739. Springer, 2007, pp. 523–530.

[6] "Genetic programming and model checking: Synthesizing new mutual exclusion algorithms," in Proceedings of the

[7] 6th International Symposium on Automated Technology for Verification and Analysis (ATVA ’08), vol. 5311. Springer, 2008, pp. 33–47.

[8] C. Johnson, “Genetic programming with fitness based on model checking,” in Proceedings of the 10th European Conference on Genetic Programming (EuroGP 2007), vol. 4445. Springer, 2007, pp. 114–124.

[9] Hauptman and M. Sipper. Evolution of an efficient search algorithm for the Mate-in-N problem in chess. In Proceedings of the 2007 European Conference on Genetic Programming, pages 78–89. Springer, Valencia, Spain, 2007.

[10] Barone, L., While, L.: Adaptive learning for poker. In: Proceedings of the Genetic and Evolutionary Computation Conference. (2000) 566{573

[11] Fogel, D., Hays, T., Hahn, S., Quon, J.: A self-learning evolutionary chess program. Proceedings of the IEEE 92 (2004) 1947{1954

[12] P. Aksenov. Genetic algorithms for optimising chess position scoring. Master’s Thesis, University of Joensuu, Finland, 2004.

[13] Y. Bjornsson and T.A. Marsland. Multi-cut alpha-beta-pruning in game-tree search. Theoretical Computer Science, 252(1-2):177–196, 2001.

[14] 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.

[15] R. Gross, K. Albrecht, W. Kantschik, and W.Banzhaf. Evolving chess playing programs. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 740–747. Morgan Kaufmann Publishers, New York, 2002.

[16] Barone, L., While, L.: An adaptive learning model for simpli¯ed poker using evolutionary algorithms. In: Proceedings of the Congress of Evolutionary Computation (GECCO-1999). (1999) 153{160

[17] Chellapilla K. and Fogel D. B.:Evolving an Expert Checkers PlayingProgram without Using Human Expertise.IEEE Trans. Evolutionary Computation,Volume 5, Number 4, 2001, pp. 422-428.

[18] Hauptman and M. Sipper. Using genetic programming to evolve chess endgame players. In Proceedings of the 2005 European Conference on Genetic Programming, pages 120–131. Springer, Lausanne, Switzerland, 2005.

[19] Chellapilla K. and Fogel D. B.:Anaconda Defeats Hoyle 6-0A CaseStudy Competing an Evolved CheckersProgram against Commercially Available Software. Proc. of CEC, 2000, pp. 857-863.

[20] Galuszka A. and Swierniak A.: Game Theoretic Approach to Multi-Robot Planning. WSEAS Transactions onComputers, Issue 3, Volume 3, July 2004, pp. 537-542.

[21] D.N. Allsopp, et al. Coalition agents experiment: multiagent cooperation in international coalitions, IEEE Intell. Syst. 17 (2002) 26–35.

[22] G. Kendall and C. Smith, “The evolution of blackjack strategies,” in Proc. Congr. Evol. Comput., vol. 4, 2003, pp. 2474–2481.

[23] M. Harman, “The current state and future of search based software engineering,” in Proceedings of International Conference on Software Engineering / Future of Software Engineering 2007 (ICSE/FOSE ’07). IEEE Computer Society,2007, pp. 342–357.

[24] Y. Jin, Knowledge Incorporation in Evolutionary Computation. New York: Springer-Verlag, 2004.

[25] K.-J. Kim and S.-B. Cho, “Evolving speciated checkers players with crowding algorithm,” in Proc. Congr. Evol. Comput., vol. 1, 2002, pp.407–412.

[26] Handbook of Evolutionary Computation, Oxford Univ. Press, London, U.K., 1997. C6.1 S. W. Mahfoud Niching methods.

[27] Dharm Singh, Thaker Chirag S and Shah Sanjay M. “Multimedia Game Based Fitness Function Optimization in Evolutionary Search Process” in IJCA Special Issue on IP Multimedia Communication in October 2011 ISBN:978-93-80864-99-3.

[28] “Evolving an expert checkers playing program without using human expertise,” IEEE Trans. Evol. Comput., vol. 5, no. 4, pp. 422–428, Aug. 2001.

[29] D. B. Fogel, “Evolving a checkers player without relying on human experience,” ACM Intell., vol. 11, no. 2, pp. 20–27, 2000.