Optimizing Fitness Function for the Game of Go-Moku

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

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

Shah Sanjay M
Research Scholar, Faculty of Engg. SGVU, Jaipur

ABSTRACT
Game playing has been the area of research in Artificial intelligence. Particularly, board game playing programs are often described as being a combination of search and knowledge. Board Games, due to its very nature, provide dynamic environments that make them ideal area of computational intelligence theories, architectures, and algorithms. In board games, it has always been the challenging task to build a quality evaluation function. The goodness or badness of the evaluation function is determined by its accuracy, relevance, cost and outcome. All of these parameters must be addressed and the weighed results are added to an evaluation function experimentally. Evolutionary algorithms such as Genetic algorithm are applied to the game playing because of the very large state space of the problem. While following the natural evolution, the fitness of an individual is defined with respect to its competitors and collaborators, as well as to the environment. Evolutionary algorithms follow the same path to evolve game playing programs. Among all computer board games, Go-moku, which is a variant of a Game of GO. This paper mainly highlights how genetic algorithm can be applied to game of Go-Moku.

Keywords
Deterministic Games, Board Games, Go-Moku, Chromosome, Fitness function.

1. INTRODUCTION
Playing games require sophisticated intelligence in a well-defined problem where success is easily measured. Most of the board games provide very large state space to search. Games have therefore proven to be important domains for studying problem solving techniques. Most of the research in game playing has attracted researchers on creating deeper level searches through the possible game scenarios. Through games, efficiency of AI working can be measured in terms of capability to acquire intelligence. The old techniques of artificial techniques were developed, tested and improvised for such games. [1][2]

While implementing any problem, which requires knowledge and decision making process, there is lot to learn. In fact, one of the leading contributions of applying AI expertise to develop game-playing programs was the realization that a search-intensive ("brute-force") approach. This 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 have been developed and successfully deployed to variety of problems.The application domain of game learning and move making programs are primarily an optimization problem. The degree of sophistication lies in efficiency of search algorithms which in turn evaluates current power of its evaluation function in very large search space. In this paper, how a genetic approach can be applied to the Go-Moku game is presented.The availability of cheap and powerful hardware has made the programming of complex problem easy and affordable. Availability of these hardware and software tools made it possible to simulate complex physical learning environments, resulting in an exploration of artificially improved soft cognitive moves by computer programs in all sorts of board games. Game playing programs have become a facet of many people's routine lives. [3]

The high state complexity of almost all traditional board games leads it to AI research area of state space search for making a next move. These games provide challenges in the form of guiding the evolution with the use of human knowledge and achieving successful and intelligent game playing behaviour [4][5]

2. HISTORY AND RULES OF GO-MOKU
Japan country has given many board games to the world. The game of GO is a well known board game from Japan. The game of Go-Moku is a variant of the game of GO. It is also known as a connect-5 game. It is a two player game. It falls in the category of zero-sum deterministic finite board game with perfect information. In two player games, the result can be win by one of the players or a draw.The game is having very simple rule but a highly complex game. It is played on a square board of 19 X 19 size. The rows and column are labelled.

The players alternate their moves. The player with black piece starts the game. The move can be made in any free position on the board. The game is over when one of the player has got five pieces in one line either horizontally, vertically, or diagonally-major or minor. To reduce the advantage to the first player, in a variant of the game, the next move of black is restricted in 5 * 5 square area of first move. [18].

State-space complexity of any board game represents the number of possible board states in the game. For example, in the game of Go-Moku, there are 361 board locations where each location can take one of three values, giving approximately 3361

total state space. The players need to understand some important structures to maximize win and to minimize loss.

Table 1: shows important structures of the game

Open four	Four	Three	Split three
4 same colour pieces in a row, with both sides open	4 same colour pieces in a row, with only one side open	3 same colour pieces in a row, chance to convert it into four in next move	3 same colour pieces in a row, with an empty position. Chance to convert it into four in next move
In the next move, definite win for player having open four.	In the next move, there is a chance to win if opponent do not occupy the open position of Four.	In the next move, possibility to convert it into Four or Open Four	In the next move, possibility to convert it into Four or Open Four

3. GENETIC ALGORITHMS

The Genetic Algorithm was developed by John Holland (1975, 1992) and his students, including David Goldberg (1988). The original applications were in engineering and were predominantly optimisation problems. But the GA can search for more general optima. Axelrod (1997) recounts how his colleague at Michigan, John Holland, mentioned that there was this new technique in Artificial Intelligence (or what has become known as Machine Learning) which, by simulating natural selection, was able to search for optima in extremely non-linear spaces. With state-space games, the state summarises all history that is payoff-relevant, and players' strategies are restricted to depend only on the state and the time. [19]Games, naturally are a problem involved with searching of state space. Genetic algorithm is the subset of evolutionary algorithms. It is a natural choice for game playing and learning problems because it provides an algorithmic and logical framework for exploiting all possible board game scenarios through natural-evolution processes like selection, cross over and mutation. It iteratively evolves intermediate candidate solutions to problem domains that have large solution search spaces. It does not use exhaustive search or conventional optimization techniques but uses randomized searching. In practice, Genetic algorithms have been applied to a broad range of learning and optimization problems through a set of Genetic Parameters. [9][10]The program acquires a novel set of evaluation function parameters as generations of the genetic algorithms are executed through a series of experiments.

The process of evolution starts with a random set of candidate solutions also called as chromosomes. These set of candidate solutions is known as population. Using a cross over process and mutation operators, it evolves the population towards an optimal set of solutions. As there is no guarantee of an optimal solution in the case of Genetic algorithm, the main challenge is to design a "genetic" process that maximizes the likelihood of generating such an optimized solution. [11]As Genetic Algorithms is iterative in nature, in each generation, the first step is typically to evaluate the fitness of each candidate solution in the current population of chromosomes, based on the fitness values and to select the fittest candidate solutions to act as parents of the next generation of candidate solutions. After the selection process, the selected parents are recombined or mated through a crossover operator and then mutated using a mutation operator to generate offspring. The fittest parents and their new offspring form a new population, from which the process is repeated to create new populations in the coming generations. Selection, recombination, and mutation are generic operations in any genetic algorithm.

The operations of evaluation, selection, recombination and mutation are usually performed repetitively for each of the iteration. So in a genetic algorithm, a major challenge is the design of the fitness function and the structure of chromosomes which reflects the problem domain. The value returned by the fitness function is called as fitness value. Other important parameters in Genetic algorithms are the size of the population, the portion of the population taking part in recombination, and the mutation rate. The mutation rate defines the probability with which a bit is changed in a chromosome that is produced by a crossover [12]. The mutation operator provides for some randomness and mainly to avoid the problems like local maxima and local minima. It does not aim to complete an exhaustive search, but it has power to quickly identify and converge on useful solutions. It provides an effective means for going beyond the structured conventional engineering approach which is very common to many forms of human design. The solution search process is inherently parallel and can be accelerated significantly by utilizing an evolutionary search algorithmic to find fitter solutions through strong moves. At a stage, all that is required is to be able to compare two solutions and indicate which is better. In Genetic algorithm, the proposed solution does not aim to find global optimum. The 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. In fact, for a large set of such high complexity driven problems, a unique global optimum does not exist. At first glance, automatic tuning of the evaluation function appears like an optimization task, which is very well suited for Evolutionary algorithms like GA. The many parameters associated with the evaluation function which are nothing but the mirroring of the features associated with the game, can be encoded as a bit string.Genetic algorithm explores a large number of points simultaneously in a search space. This phenomenon avoids the chances of poor local optima quickly, thus resulting in a quicker and fruitful search. The mutation rate defines the probability with which a bit is changed in a chromosome that is produced by a crossover [12]. The mutation operator provides for some randomness and mainly to avoid the problems like local maxima and local minima.The space to be searched is huge. Due to the complexity of Go-Moku game, any search algorithm based method which is based on exhaustive search for the problem space is infeasible.

3.1 Fitness Function

An important parameter in the implementation of Genetic algorithm is the fitness value returned by the fitness function. As

the genetic process evolves, the fitness value of each chromosome is found out. The chromosomes with worst fitness values are discarded while keeping the best chromosomes. To avoid the problem of local minima or local maxima sometimes a mutation rate is increased. This paper uses genetic algorithm in game playing by constructing a static evaluation function. The function uses the structures and features of the game of Go-Moku.

4. APPLYING GA TO GO-MOKU

The board is represented as a two-dimensional 19 * 19 elements array. Each board position can have any of the following values: -1= Free position in neighboring zone, 0 =Free position, 1= Computer player (using GA) piece, 2= Human player piece.The neighboring zone of considered board position is the set of all occupied positions i.e. positions with value 1 or 2. The fitness value is derived using fitness function for a move to position (x, y) on the board. The chromosome represents a sequence of alternate plays by computer algorithm and player. The fitness function uses a table of weights to calculate the fitness value for a considered board position. The fitness value of a gene is calculated as the sum of weights of all sequences of pieces surrounding the gene under consideration. The static evaluation function gives more weightage to GA player than the human player. If we reverse the role of weightage, then the program will work in a mode of defense.

For figure 1[20],

FV = value of two structure for GA + 2 * value of one structure for Human play.The algorithm calculates the sum total of first genes which occur as per the prediction value selected. If the prediction value is 3, then sum total of first 3 genes take place.This value is used to decide the next actual move suggested by the first gene of selected chromosome.

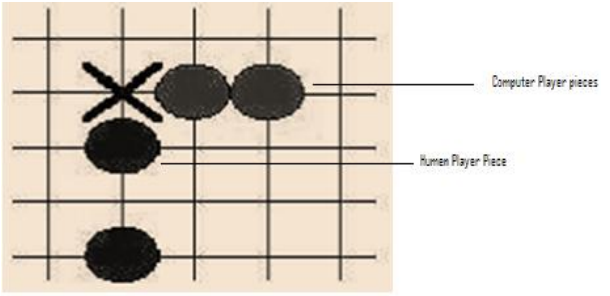

Fig: 1

4.1 Implementation

The game board interface is represented as a square of 19 * 19 and the intersections are used as the valid places for move. The pieces are represented as circles.The chromosome is represented as a structure with three variables x, y and fitness value for that position. The weights for the various structures are used per the static weight table. The fitness value of the considered board position is calculated as the sum total of the weight values of the surrounding genes in the neighbourhood. According to the fitness values found for all considered board positions, the move having the maximum fitness values is selected as the next move by computer.

The parameters of GA used were,

- genes per chromosome =20,
- Population size =20
- Rate of crossover = 0.5
- Number of iterations = 20

5. CONCLUSION

The simplicity of fitness function is heavily based on the feature characteristics of the game. 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. 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.

6. REFERENCES

[1] 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.

[2] J. Clune. Heuristic evaluation functions for general game playing. In Proc. of AAAI, 1134–1139, 2007.

[3] 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.

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

[5] 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.

[6] 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.

[7] Thomas P. Runarsson and Simon M. Lucas. Co-evolution versus self-play temporal difference learning for acquiring position evaluation in small-board go. IEEE Transactions on Evolutionary Computation, 9:628 – 640, 2005.

[8] Yannakakis, G., Levine, J., and Hallam, J. (2004). An evolutionary approach for interactive computer games. In Evolutionary Computation, 2004. CEC2004. Congress on Evolutionary Computation, volume 1, pages 986–993, Piscataway, NJ. IEEE.

[9] A. 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] P. Aksenov. Genetic algorithms for optimising chess position scoring. Master's Thesis, University of Joensuu, Finland, 2004. Y. Bjornsson and T.A. Marsland. Multi-cut alpha-beta-pruning in game-tree search. Theoretical Computer Science, 252(1-2):177–196, 2001.

[11] 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.

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

[13] G. Kendall and G. Whitwell. An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In Proceedings of the 2001 Congress on Evolutionary Computation, pages 995–1002. IEEE Press, World Trade Center, Seoul, Korea, 2001.

[14] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. Ann Arbor, MI: University of Michigan Press.

[15] Goldberg, D. E. (1989). Genetic Algorithms in Search,Optimization and and Machine Learning. Reading, MA: Addison-Wesley.

[16] Buckles Bill P. and Petry, Frederick E. Genetic Algorithms. Los Alamitos, CA: The IEEE Computer Society Press. 1992.

[17] Haupt, Randy L, and Haupt, Sue Ellen. (1998). Practical Genetic Algorithms. New York: John wiley & Sons

[18] L.V. Allis,H.J. van den Herik ,M.P.H. Huntjens. Go-Moku and Threat Space Search.

[19] Robert E. Marks Playing Games with Genetic Algorithms

[20] Sanjay M Shah, Chirag S Thaker, Dharm Singh Multimedia Based Fitness Function OptimizationThrough Evolutionary Game Learning at International Conference on ETNCC 2011 at MPUAT, Udaipur on 22-24 April 2011.