Board games are one of the oldest branches of Artificial Intelligence (Shannon and Turing 1950). These games represent a pure and quite abstract form of competition involves in 2 players which requires a form of “intelligence”. In such games, states can be represented easily and each possible action is well defined. Implementing AI on such problems are affective way to understand and evaluate AI based algorithms, as all the individual states are fully accessible. As characteristics of the opponent are unknown thus it is a contingency problem. Search trees that are required to solve board games can become astronomically large.

Algorithms used in board games should have following properties:

1. use good evaluation functions for in-between
2. look ahead as many moves as possible
3. delete irrelevant branches of the game tree, states

In the paper “General Board Game Playing for Education and Research in Generic AI Game Learning” by Wolfgang Konen Computer Science Institute TH Koln – Cologne University of Applied Sciences ¨ Gummersbach, Germany, importance of GBG(general board game) study and research work is highlighted in order to understand and analyse AI algorithms. This paper helped us to understand the basics of developing GPG with AI and gave insights to the algorithm such as MinMax. The games on which the AI is evaluated in this paper is 2048 and Hex gameboard. This paper helps researchers in game learning to quickly test their new ideas or to examine how well their AI agents generalize on a large variety of board games.

A Comparative Study of Game Tree Searching Methods

In 2014, a journal ((IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 5, No. 5, 2014) published on Egypt highlight the Comparisons between Game Tree Searching Methods. In complex Board problems many algorithms have been discussed to find the next-move such sequential algorithms such as MiniMax [1], NegaMax [2], Negascout [3], SSS\* [4] and B\* [5] as well as parallel algorithms such as Parallel Alpha-Beta algorithm [6]. Almost all the board games use game tree with the following components-

• Each node represents a game state.

• The root represents the current game state.

• All the branches for a given node represent all the legal moves for that node.

Evaluation function determines score for the path taken to the specified branch. There exist a big set of methods for sequential game tree. The paper has underlined sequential game tree algorithms categorized into depth first and breadth search algorithm.

This paper gave the example of Tic-Tac-Toe game with MInMax , NegaMax, Alpha-Beta, NegaScout Algorithm. These algorithms are different in the terms of Moving orders. These all techniques are categorized under Brute- Force algorithms in Depth- First Search. MinMax tries to evaluate the best move for AI. It maximizes the AI Score along with taking care to minimize human score. Other mentioned algorithms are enhanced in some ways and limited in other ways in order to increase efficiency and decrease computation level of MinMax.

Important Difference between the brute-force algorithms and the selectivity algorithms is that Selectivity algorithms in Depth-First Search doesn't depend on fixed depth to stop looking in each branch. The most common techniques in this category are Quiescence Search and Forward Pruning. Quiescence Search [15] based on the idea of variable depth searching. The algorithm follows the normal fixed depth in most branches and pruning cut of the irrelevant computations. Many algorithms implemented the idea of this technique, including N-Best Selective Search, ProbCut and Multi-ProCut [16].

Parallelism in Game Tree search implementation requires multiprocessors and multi core computers. There are number of algorithms categorised in this section such as A, Parallel Alpha-Beta, Parallel PVS, YBWC and Jamboree and Dynamic Tree Splitting. Some of the algorithms are same as sequence algorithms such as alpha beta but is the parallel version.

After discussion of various algorithms in the paper which includes sequential and parallel algorithms, we chose to continue our deeper dive with sequential algorithms as parallel algorithms require better CPU and GPU resources for parallel computations. Inside sequential algorithms narrowed our search to Brute- Force algorithms in Depth- First Search. We chose to evaluate and compare these algorithms against each other with respect to various components such as time, number of steps, depth etc.

Previous work related to Checkers:

We found a paper submitted by Elmer R. Escandon´ ∗ , Student Member and Joseph Campion∗ (IEEE Members). The paper highlights the success of AI against average human player on Checkers Board game. They compared the simple MinMax algorithm with varying depths. Their studies proved that AI was successful with the depth of 5-6 but struggled hard with the depth of 3-4. They also compared the time with respect to the depth of the model. the efficiency of the AI was evaluated in terms of run time to complete the minimax algorithm and success of the AI against human beings.

The results based in this paper shows the exponential rise of time with respect to the depth of MinMax algorithm and Succes rate direct proportionality with respect to depth.

In this paper we chose MinMax algorithm as our baseline and compared other sequential brute force algorithms- NegaScout, AlphaBeta, Randomized AlphaBeta.

In the paper “Rminimax: An Optimally Randomized MINIMAX Algorithm by Silvia García Díez, Jérôme Laforge, and Marco Saerens”,importance of Randomized Alpha beta is highlighted. This algorithm has proved to be an essential component of board games as behaviour of AI can be turned predictable for regular player. To avoid the predictability there is a requirement of randomization element in AI model. Thus, it is a crucial part to evaluate the success rate of Randomized Algorithm. In this paper we have applied randomization on AlphaBeta algorithm.