CHAPTER 1
INTRODUCTION

If we look at the history, mankind has been influenced by the thought of creating machines to perform the most difficult of tasks. Men has always dreamt of and tried and speculated about achieving the things beyond the scope of the technology of their time.

1.1 ARTIFICIAL INTELLIGENCE:

Artificial intelligence is about making computers capable to perform the thinking tasks that humans and animals are capable of [1]. In academic world, some AI researchers are motivated by philosophy: understanding the natural world of thought and the nature of intelligence and building software to model how thinking might work. Some are motivated by psychology: understanding the mechanics of the human brain and mind processes. Others are motivated by engineering: building algorithms to perform humanlike tasks. This three-way distinction is at the heart of academic AI, and the different mind-sets are responsible for different subfields of the subject.

As developers of the games, we are mainly interested in only the engineering side: building algorithms that make game characters behave human or animal-like.

In the modern age of AI, there was an increasing dissatisfaction with symbolic approaches. There was also an important philosophical argument made that symbolic approaches weren’t biologically believable. The effect was a move toward natural computing: techniques encouraged by biology or other natural systems. These techniques include neural networks, genetic algorithms, and simulated annealing.

The definition of AI is: “AI is the science of making machines do things that would require intelligence if done by men”. It is the field of computer science that seeks to
understand and implement computer-based technology that can simulate characteristics and features of human intelligence.

For a machine to be intelligent, it must be able to reason, learn from experience, set goals for itself, and get used to the world around it. A machine that can do these stuff is the machine that humans have been trying to build for a long period.

1.1.1 AI AND GAMES:

Since the beginning years of Artificial Intelligence (AI) computer games have been used as a testing ground for AI algorithms. Games have been used to test new concepts because of their simplicity compared to real world situations and simulations. Many pioneers in computer science spent their valuable time on algorithms for chess, checkers, and other games of strategy. A partial list includes such researchers as Alan Turing, John von Neumann, Claude Shannon, Herbert Simon, Alan Newell, John McCarthy, Arthur Samuel, Donald Knuth, Donald Michie, and Ken Thompson.

Pac-Man was the first game many people remember playing with AI [1]. Pac-Man had definite enemy characters that seemed to plan against you, moved around the level just as you do, and made our life tough. Pac-Man was relying on a very simple AI technique: a state machine. Each of the four monsters was either chasing you or running away. For each state they took a semi-random route at each intersection. In chase mode, each had a different chance of chasing the player or choosing a random direction. In run-away mode, they either ran away or chose a random route.

Golden Axe had a neat originality with enemies that would rush past the player and then switch to homing mode, attacking from behind. The cleverness of the AI is only a small step from Pac-Man.

Goldeneye 007 tried to show gamers what AI could do to improve gameplay. Still relying on characters with a small number of well-defined states, Goldeneye added a sense simulation system, where in the characters could see their colleagues and would notice if they were killed or wounded.
Warcraft was one of the first times pathfinding real time strategy game. AI researchers were working with emotional models of soldiers in a military battlefield simulation.

AI for a game is basically a combination of adhoc solutions, heuristics and algorithms. A heuristic is a rule of thumb, a rough approximation of solution that might work in many situations but is unlikely to work in all situations. We as humans use heuristics all the time. We don’t try to work out all the consequences of our actions. Instead, we mostly rely on general principles that we have found to work in the past. There is a trade-off between speed and accuracy in areas such as decision making, movement, and tactical thinking. When accuracy is compromised, it is usually by replacing the search for a correct answer with a heuristic.

In many strategic games, including board games, different positions or pieces are given a single numeric value to signify how good they are. This is a heuristic; it replaces complex calculations about the capabilities of a unit with a single number. And the number can be defined by the programmer in advance. The AI algorithm can calculate and find out which side is leading ahead simply by adding the numbers.

The AI in most modern games addresses three basic requirements [1]

- The ability to move characters,
- The ability to make decisions about where to move, and
- The ability to think tactically or strategically.

The game can be defined as a decision problem with two or more decision makers i.e. players, where the outcome for each player may depend on the decisions made by all players. This definition is based on studying the essential elements of many existing game definitions over a broad range, most of which shared following important common points.

- Rules – clearly defined as to what actions and behaviour each player can take at which point in time are legal.
• Play (clash) – players interact with each other using rules of the game and the options the player is having at that time. These decision points actually direct the game.

• Outcome – at the end of interaction between players, the outcome can be described numerically – each player receives pay compensation, which may be negative number also.

The concept of play is common in all definitions. It is very important to understand the context of play – strategic play and tactical play. Strategy is the art of war and concerns the overall administration of forces in clash with the enemy. The ability to conceptually decompose play into meaningful sub goals is important as in the case of chess game is an example of strategy.

While tactics is the low level use of forces to achieve specific goals. Strategy is a long term thought, while tactics involves decisions that change with every game move. Tactics therefore associate with short term local combinatorial play and strategy with long term planning.

In a two-person game there are two players that make moves alternately. Chess, checkers, tic-tac-toe, Othello, Go-Moku are examples of two-person games. The complexity of games can be considered in terms of space complexity and game tree search complexity. Branching factor is one of the parameter as far as game tree search complexity is concerned. Higher the branching factor, higher is the game tree complexity.

Brute force search can not be applied to the games such as Othello, Chess, Go-Moku etc. just because of very high branching factor. One must look at more intelligent and optimized methods of searching where pure search is ruled out. Current Go-Moku programs play at the novice level because of their insistence of traditional methods, hard coded rules, mini-max search and alpha beta pruning methods.

I propose that programs should use evolutionary search in board games such as Go-Moku and other games. Traditional methods of search can be combined with soft computing methods such as genetic algorithm in game play.
Genetic algorithms have the tendency to slow the program response. The problem is therefore to develop a genetic algorithm based game program which performs well without sacrificing the response time. The agent will generate a value for each location on the board. These values are stored in a matrix such that each location on the board corresponds to a place in the matrix. These matrices are then filled with genetic algorithm evolved weights. The resulting move therefore will be at the place where the value is highest.

The study of board games, card games, and other mathematical games of strategy is popular for a number of reasons. In general, they have some or all of the following properties or characteristics:

- Games have well-defined rules and simple logistics, making it relatively easy to implement a complete player, which allows more time and effort to be spent on the actual topics of scientific interest.
- Games have complex strategies, and are among the hardest problems known in computational complexity and theoretical computer science.
- Games provide a clear specific goal, thus providing a clear-cut definition of success, and efforts can be focused on achieving that goal.
- Games provide quantifiable results, either by the degree of success in playing the game against other opponents, or in terms of the solutions to related subtasks.

Many game-playing programming systems have reached an expert level using a search-based approach. Games such as Backgammon, Chess, Checkers, Go, Othello and Tic-Tac-Toe are widely used platforms for studying the learning ability of and developing learning algorithms for machines. By playing games, the machine intelligence can be shown. Some techniques of artificial intelligence, such as brute-force methods and knowledge-based methods were reported. Brute force methods solve the game problems by constructing databases for the games. The best move is determined by working backward on the constructed database. While, for knowledge based systems, the best move is determined by searching a game tree. The above game solving methods depend mainly on the database construction and searching. The
problem solution is arrived by forming a possible set of solutions based on the endgame condition, or searching for solutions from current condition of the game.

1.2 COMBINATORIAL GAME THEORY:

Combinatorial game is understood to be discrete, finite, deterministic and perfect information [3]. Games with one player make up puzzles. The player wins by achieving the required goal before running out of moves, else the inventor of the game wins. All the games involve conflict, whether it involves conflict between direct players of the game or between player and the challenging activity presented by the game itself.

Multiplayer games are those games where there are more than two players. Multiplayer games are distinguished from two player games due to the coalitions that may come up during the play, which can lead to some interesting play dynamics- such as two losing players form a union against leading player. The other dynamics may be that the losing player with no hope for winning may decide the ultimate winner which motivates a losing player to keep playing and the best player may be eliminated without any fault.

Combinatorial game theory is the study of two-player combinatorial games. It is mainly concerned with the mathematical analysis of the games with a vision in mind to forming most advantageous strategies and solving games wherever possible. One of the very important concepts in the combinatorial game is the urgency of making a move positive move (beneficial move) or negative move (forced move).

The main emphasis is of research on combinatorial games is on finding algorithms which can do better than human experts. In answer to the question “Which games will survive?” Allis suggests that “all games will survive at all levels”, even if it becomes a competition between programmers who are expert game players.
1.2.1 Types of combinatorial games [2]:

Classification of board games is as under.

- **Race games**: Involves moving pieces along linear tracks between starting and finishing points. The interaction between players involves ousting opponents pieces back to their starting positions.

- **Space games**: It involves arranging pieces into patterns or to satisfy some geometric requirement such as occupying some territory. It also involves disturbing spatial relationship of the opponents pieces and trying to create spatial relationship for own pieces.

- **Chase games**: The player wins by cornering some critical key enemy piece. These games involve unbalanced forces in the game.

- **Displacement games**: Involve the capture of certain number of enemy pieces or capture certain category of enemy pieces.

- **Games of fixed and variable geometry**: Game of fixed geometry involve the creation of pre-defined fixed pattern on the board, such as lines of N- pieces on the board or a particular type of shape. The games of variable geometry involve geometric goals that may be achieved in many different ways. Here, the property of geometry is important, not its shape. It includes various connection games such as path making, cycle making etc.

1. 3 BOARD GAMES:

Board game refers to any game played with a pre-marked surface, with counters or pieces that are moved across the board. Board games have been always popular, and the origins of board games date back to early times.

Board games usually consist of boards, pieces, a system of rules and players. They have been used as a pastime as well as a tool to teach children and explain to adults over a thousands years. There are many different types and classifications of board games.

The history of board games provides that there are two categories of board games - Strategy games and Racing games. The objective of the strategy games is to gain control of a larger board by using pieces to block or capture an opponent’s pieces.
Chess, Risk and Monopoly are examples of strategy board games. The racing games are meant to begin at a specific point on the board game and race along one or more paths to reach a specific goal finish line before your opponent. Chutes and Ladders, Life and Parcheesi are examples of racing board games.

Board games are normally designed to involve luck, skill or both luck and skill. Following figure 1.1 shows 3 classifications of board games [2].

The classic Snakes and Ladders is a pure luck based board game which doesn’t require the players to make any decisions during the play. These sorts of games are often targeted for children. There are various methods of introducing luck in board games. The most common is using the dice. In the Snakes and Ladders, the number on the dice is used to determine how many steps a player move his or her token. Games like Sorry! use deck of cards to create randomness and Scrabble involves luck by requiring players to pick arbitrarily pick letters.

The popular Chess board game is solely based upon skill. Most of the board games involve both luck and skill. Monopoly comprises both luck and skill as it uses the dice and it requires some thinking. Players may be losing in a game like this due to luck connection, but a player with a superior strategy will win more often. Adult game players usually find purely luck based games quite boring and prefer games that require them to make some decisions.
Types of board Games

<table>
<thead>
<tr>
<th>Luck</th>
<th>Luck and Skill</th>
<th>Skill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Involves spinners, dice, cards etc.</td>
<td>Involves spinners, dice, cards etc but also include choice and strategy</td>
<td>Do not use spinners, dice, cards, but require skill and practice</td>
</tr>
<tr>
<td>Easier to play and develop</td>
<td>Harder to play and develop</td>
<td>Very difficult to play as it involves rules and options to move and to win and also very difficult to develop</td>
</tr>
</tbody>
</table>

1.3.1 Modern board games:
In this modern era of technology, computers have greatly affected the traditional board games, as most of these board games are now computerized. Like any other computer games, board games can be downloaded, installed and played on personal computers. Similar to other online games, these board games can also be played online against other players situated remotely.

Many researchers seem to support the idea of computerizing existing board games for several reasons. By speeding up these boring offline tasks, a computer allows players to focus more on their strategy and actual game playing, and less time on the accounting. Most modern games are designed closely to original board game. Often, the physical look and rules of the game are well preserved but how the players interact in the game differs. For example, instead of having the players to throw the dice, they are now required to click at the dice on the screen. The dice is programmed to randomly display a number ranging form 1 to 6. The use of Artificial Intelligence (AI) allows modern board games to be more intelligent.

1.3.2 Two player board games:
Two player board games such as chess defeated world champion Kasparov in exhibition match. In the class of two-player perfect-information games such as Chess, Checkers, Draughts, there are no strong programs that can challenge a strong human player.
Two player board games such as chess, checkers, Go, Go-Moku, Othello, Bridge etc. have following features or properties [1]:

1. **Perfect Information:**

   The perfect-information property divides the set of games into two disjoint subsets- the set of perfect-information games and the set of imperfect information games. In a perfect-information game, all players, at any time during the game, have access to all information defining the game state and its possible continuations. There is nothing hidden. Any game which is not a perfect-information game is defined to be an imperfect-information game. Best possible play in a perfect-information game always consists of a pure strategy, while in imperfect-information games optimal play may require a mixed strategy. In a pure strategy, for each game state a single move can be determined, which leads to the game-theoretic value of the position. In a mixed strategy, optimal play requires a player to play a move with probability $P$.

2. **Convergence:**

   The convergence property labels games as converging, diverging or unchangeable. For most games, the main conversions involve the addition of a piece as in connect-four, Go-moku and Othello or removal of a piece as in chess, checkers, bridge from play. Othello is an example of a diverging game. Each move in Othello adds a piece to the board. Except for the endgame, the number of legal positions increases as the number of pieces on the board increases. The relevance of the convergence property is that for converging games endgame databases can be created, while this is generally not practicable for diverging or unchangeable games.

3. **Sudden death:**

   Go-Moku is an example of a sudden-death game. The game is terminated if one of the players has created a line of five pieces of his/her colour. Sudden death games need not always terminate through the creation of
a sudden death pattern: Go-Moku is declared a draw when all the 225 squares have been occupied without either player creating a winning pattern.

Othello is an example of a fixed-termination game. Othello lasts until both players run out of moves or one of the players has no discs left on the board. In practice, games last between 55 and 60 moves. Even though a game might be decided within 15 moves by one player capturing all the discs of the opponent, such an variance is only of marginal relevance. The sudden-death property often is an important property in restricting the search tree of a game.

4. Complexity:

The property complexity in relation to games is used to denote two different types of complexity
- State-space complexity and,
- Game-tree complexity

In deterministic games, the nodes in such a tree correspond to game states, and the arcs correspond to moves. The initial state of the game is the root node; leaves of the tree correspond to terminal states.
The size of game tree can be calculated by taking the average branching factor and raising it to the power of the number of ply the game usually lasts before ending. In chess for example, each player has, on average, about 36 possible moves. Thus, the average branching factor for Chess is 36. Since the average game of chess last around 40 moves per player i.e. 80 ply, the game tree is $36^{80}$ or approximately $10^{124}$.

5. Two-player:

Most games are two-player games, as opposed to zero-player games, one-player games (15-puzzle, Solitaire) and multi-player games (poker and diplomacy).

6. Zero-sum:
These are games where one player's loss is the other player's gain. The games like Othello, Chess, Go, Go-Moku etc. are zero-sum games. While the prisoners' dilemma is not zero-sum.

7. Non-trivial:
   A best playing strategy should not be trivially established through enumeration or mathematical analysis. Examples of trivial games are tic-tac-toe and nim.

8. Well-known:
   These are games which have been played by large numbers of people, resulting in the game being known in several countries.

9. Requiring skill:
   Some games serve mainly as a pastime, not requiring much skill. The more experienced player has no real advantage in those games. The games included here should exhibit a strong relation between skill and winning chances. Such a relation also exists in some games which are subjective to a chance element, such as backgammon and bridge.

1.4 GAME PLAYING:
The traditional artificial intelligence approach to game programming is brute-force search. This approach is based on a definition of a game as a kind of search problem with the following components [4]:

- **The initial state**: The board position and an indication of which player is to move.
- **A set of operators**: What legal moves a player can make?
- **A terminal test**: Check is the game over?
- **A utility function**: A numeric value for the result of a game.

A typical example used to illustrate this definition is the simple game of Tic-Tac-Toe. The initial state is the empty board, with nine empty cells. A move consists of placing a marker in one of the cells, the cell has to be empty in order for the move to be legal.
Determining the end of the game is simple - there are only two types of terminal positions. The most general position is the draw where neither player can make a legal move. A position in which either player has managed to place three markers in a row is a winning position for that player, and therefore a losing position for the other player. A utility function for Tic-Tac-Toe could for example assign a value of 0 to a draw position, a value of 1 to a win, and a value of -1 to a loss.

The game-tree for Tic-Tac-Toe is very simple to allow a full expansion, and a small part is shown in Figure 1.2. Utilities for the terminal states can be propagated back to the initial state, the empty board. This makes it easy to construct a program that can play Tic-Tac-Toe without losing. The state space of many other games however, is too large to allow such an approach. When it is not possible to determine the exact utility of a state, it must be approximated.

![Game tree for Tic-Tac-Toe](image)

**Figure 1.2 Game tree for Tic-Tac-Toe**

1.4.1 Evaluation Functions:

When perfect play is no longer an option due to the size of the state space, strong game play is still likely. Instead of searching from root node down to the
terminal states, and propagating the known utility values back up to the original state, search can be cut off earlier [1]. Instead of determining the exact utility for a state, it can be approximated by an evaluation function. Heuristics must be used to consistently evaluate a game position, so that a game-tree search can be cut off at a certain depth. Typically, it is the creation of such an evaluation function that is the problem of building a strong game-playing engine for most games. The number of plies that can be looked ahead is an important factor in the accuracy of the evaluation function - searching deeper into the game tree usually means that the estimated chances of winning the game are closer to the true chances. Using good heuristics as accurate enough evaluation function, together with a search engine performing deep searches forms the heart of most computer programs for two-player games with perfect information.

An important question is how "good" heuristics are derived. Experts of the game are a good source of knowledge, but incorporating this knowledge into a game-playing engine can be a difficult problem. Allowing the machine to automatically find out these heuristics by itself can be an effective solution to this problem.

1.5 OPTIMIZATION:

From the existence of humankind, we strive for perfection in many areas. We want to reach a maximum degree of joy with the least amount of attempt. In our financial system, profit and sales must be maximized and costs should be as low as possible. Therefore, optimization is one of the oldest of sciences which even extends into daily life. Optimization is the process of making something better. It consists in trying variations on an initial concept and using the information gained to improve on the idea [29]. Optimization is the process of adjusting the inputs to or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or result. The input consists of variables, the process or function is known as cost function or objective function or fitness function and the output is the fitness value or cost. Quantum theory suggests that there are an infinite number of dimensions, and each dimension attributes to a decision taken. A computer is considered as the perfect tool for optimization as long as the idea of variable
influencing the idea can be input in electronic format. When we say that we are interested in finding the best solution, then it means there exist more than one solution and the solutions are not equal. The definition of best is relative to problem considered, its method of solution and also the allowed tolerance value.

Global optimization is the branch of applied mathematics and numerical analysis that focuses on optimization. The goal of global optimization is to find the best possible elements $x^*$ from a set $X$ according to a set of criteria $F = \{f_1, f_2, \ldots, f_n\}$. These criteria are articulated as mathematical functions, the so-called objective functions.

An objective function $f: X \to Y$ with $Y \subseteq \mathbb{R}$ is a mathematical function which is subject to optimization. The codomain $Y$ of an objective function as well as its range must be a subset of the real numbers ($Y \subseteq \mathbb{R}$). The domain $X$ of $f$ is called problem space and can represent any type of elements like numbers, lists, construction plans, and so on.

Generally, optimization algorithms can be divided in two basic classes

- Deterministic and
- Probabilistic algorithms.

Deterministic algorithms are most often used if an obvious relation between the characteristics of the possible solutions and their value for a given problem exists. Then, the search space can well be explored using a divide and conquer method. If the relation between a solution candidate and its fitness are not so clear or too difficult, or the dimensionality of the search space is very high, it becomes harder to solve a problem deterministically. Trying this algorithm will result in exhaustive enumeration of the search space, which is not feasible even for relatively small problems.

Then, probabilistic algorithms come into play. The initial work in this area now has become one of most important research fields in optimization. An especially relevant family of probabilistic algorithms is the Monte Carlo-based approaches. They deal in guaranteed correctness of the solution for a shorter runtime. Heuristics used in global optimization are functions that aid in deciding which one from a set of possible
solutions is to be examined next. Figure 1.3 shows rough taxonomy of global optimization algorithms [49].

![Taxonomy of global optimization algorithms](image)

**Figure 1.3 Taxonomy of global optimization algorithms**

Other way of categorizing the optimization is according to variables, constraints and other things are [29]:

1. **Deterministic**
   - State Space Search
   - Branch and Bound
   - Algebraic Geometry

2. **Probabilistic**
   - (Stochastic) Hill Climbing
   - Random Optimization
   - Simulated Annealing (SA)
   - Tabu Search (TS)
   - Parakkek Tenoering
   - Stochastic Tunneling
   - Direct Monte Carlo Sampling
   - Evolutionary Computation (EC)
     - Evolutionary Algorithms (EA)
       - Genetic Algorithms (GA)
       - (LCS) Learning Classifier System
       - Evolutionary Programming
     - Evolution Strategy (ES)
     - (GP) Genetic Programming
   - Artificial Intelligence (AT)
   - Computational Intelligence (CT)
   - Memetic Algorithms
     - Swarm Intelligence (ST)
     - Ant Colony Optimization (ACO)
     - Particle Swarm Optimization (PSO)
   - Harmonic Search (HS)
     - Differential Evolution (DE)
     - Standard Genetic Programming
     - Linear Genetic Programming
     - Grammar Guided Genetic Prog.
• Trial and error optimization refers to the process of adjusting variables that affect the output without knowing much about the process that produces the output.
• Single variable or multiple variable optimizations – as the number of variables increases it becomes more difficult.
• Static or Dynamic optimization – If the output is a function of time, then it is considered as dynamic, otherwise it is considered as static. Normally, the added dimension of time increases the challenge of solving the dynamic problem.
• Discrete or Continuous variable optimization - discrete variable have only a finite number of possible values, while continuous variables have an infinite number of possible values. Discrete variable optimization is also known as combinatorial optimization.
• Constrained or Unconstrained optimization – In constrained optimization, the variables have equalities and inequalities into the cost function. Unconstrained optimization allows the variable to take any value.
• Calculus based or Randomization based optimization – traditional algorithms are calculus based and get easily stuck in local minima, while random methods are slower but have greater success in finding the global minimum.

Root finding is different than optimization, as one root is as equal to other root, since all roots force the function to zero value. The difficulty with optimization is to find out whether the solution is local best or global best.

1.5.1 Structure of Optimization:

The structure of optimization consists of a number of well-defined spaces and sets as well as the mappings between them. Based on this structure of optimization, the abstractions of fitness landscapes, problem landscape, and optimization problem lead to the definition of what optimization is.
Whenever we deal with an optimization problem, we first have to define the type of the possible solutions. The problem space $X$ of an optimization problem is the set containing all elements $x$ which could be its solution. Usually, more than one problem space can be defined for a given optimization problem. A solution candidate $x$ is an element of the problem space $X$ of a certain optimization problem. We call the union of all solutions of an optimization problem its solution space $S$.

$$X^* \subseteq S \subseteq X$$

This solution space contains the global optimal set $X^*$. There may exist valid solutions $x \in S$ which is not elements of the $X^*$. The search space $G$ of an optimization problem is the set of all elements $g$ which can be processed by the search operations.

An optimization algorithm is characterized by

- the way it assigns fitness to the individuals,
- the ways it selects them for further investigation,
- the way it applies the search operations, and
- the way it builds and treats its state information.

1.6 INTRODUCTION OF GO-MOKU BOARD GAME:

Among the two-player board games, Go-Moku has the simplest rules: two players – black and white alternate placing stones on a 15*15 or 19 * 19 square lattice with the goal of obtaining a line of exactly five consecutive stones of the player's colour. While its ancestry lie in China and Japan, it is has also become popular in several countries of Europe.

Go-Moku is a subset of generalized family of k-in-a-row games, called Connect ($k$, $m$, $n$, $p$, $q$) games, where two players alternately place their piece on $m \times n$ board. Go-Moku is a Connect($15$, $15$, $5$, $1$, $1$) game. It is also known as Five-in-a-row game. Go-Moku is a two-player game, related to the well-known small game of tic-tac-toe. Go-
moku has perfect information, because all the game information such as piece position is available to all players [5]. While in tic-tac-toe players must create a line of three consecutive markers of their colour on a restricted 3 * 3 board, in Go-Moku, players must create a line of five on a practically unrestricted lattice. Through the years, several variants of Go-Moku have been developed. 1.4 shows a sample 15 * 15 Go-Moku board initial configuration.

Figure 1.4 Go-Moku 15*15 board

In practice, writing a program to play a legal game of Go-Moku is trivial, but designing and implementing a competent Go-Moku player is a challenging task.

1.7 INTRODUCTION OF OTHELLO GAME:

Othello is a two-person, zero-sum strategic board game with simple rules. But, it takes a lifetime to master the game [6]. According to the standard rules of Othello, the game is played on an 8 * 8 board. There are total 64 double sided pieces, white on one side and black on other. The goal of the game is to have the majority of the pieces on the board at the end of the game flipped to your colour. The player having black piece starts the move. Valid moves consist of placing a piece on the board next to your opponent’s piece such that a line – horizontal, vertical or diagonal can be drawn from the played piece, across one or more opponent's pieces, to another one of your
pieces. When a piece is played in this manner, all of the encompassed or trapped opponent pieces are flipped to your colour. This leads to the board being very dynamic since it is possible to flip many opponent pieces in a single move. When a player has no valid moves, then that player passes. The game ends when neither player has a valid move remaining. At the end, the player having more number of pieces wins. If both players have equal number of pieces, then the game ends as a draw.

Following figure 1.5 shows the initial configuration of the board. The positions shown as A, B, C and D indicate valid moves available to black player.

![Figure 1.5 Othello initial board](image)

Due to the pieces being flipped in the game, the board is relatively dynamic. Othello has a fairly high branching factor and as a result is difficult to search systematically even considering the limited number of turns in a game.

**1.8 ORGANIZATION OF THE THESIS:**

Chapter 2 is about the literature survey. It discusses about computer games, Game programs people have developed for various board games, Solving a game, and searching techniques specific to board games, adversarial search which is often used in games of competitive environments. It also discusses about genetic algorithms as an evolutionary algorithm – its breeding cycle, operations, different techniques adopted in these operations, advantages and limitations of the genetic algorithm.
Chapter 3 is about materials and methods used in implementing the game programs. It explains the architecture of the game program. Discussion about how exactly the game board is represented in algorithm and it’s Graphical User Interface (GUI). Genetic algorithm implementation details such population size, size of chromosome, encoding of the chromosome, weights used to calculate fitness value of chromosomes, selection method used, crossover method and mutation probability. It also explains about the files created as a part of the game play which help in analysing the effect of various genetic algorithm parameters on the game play. Some screenshots of game play are also shown. It also explains how exactly the genetic algorithm enabled move takes place.

Chapter 4 contains discussion about experiments carried out using the materials and methods discussed in chapter 3. The experiments were performed for both the games taken as test bed namely Go-Moku and Othello. The various observations found for each of the game and corresponding results are shown in the form of various charts.

Chapter 5 contains discussion on important findings of the experiments carried out. It also explains the conclusions drawn of the research carried out and also the relevance of the data and its interpretation is explained.

Chapter 6 contains discussion on limitation of the work the work carried out, the recommendations of the research and also discusses on the direction for further research in this area of game playing using genetic algorithm.

Chapter 7 contains bibliography of references used.

Chapter 8 contains details of papers published during the period of research work and the copy of papers.