CHAPTER 2
REVIEW OF LITERATURE

Games Played by Computer programs have come a long way in the last forty years. From humble beginnings, they have grown to be the focus of human generations. To sustain such a large interest, game developers & researcher community are constantly looking for ways to advance the technology used in games. In recent decades, the idea of pursuing academic research into Artificial Intelligence (AI) for computer games has gained popularity. This has resulted in a community of researchers looking to apply and develop AI techniques that are directly applicable to computer game worlds. Computer games are an ideal domain in which to pursue AI research for a number of reasons, but primarily because their classically character based nature makes them a natural arena for the application of program-based techniques. A computer program has been well-defined as many things in older times, but for this thesis the term will be used to define anything that can perceive and act in a game playing environment.

The aim of this thesis is to explore what techniques make it possible to design and implement a program specifically for computer game worlds. To oblige this aim further, the particular programs attempted to design must display intelligent behaviour in order to enrich the player’s experience of the game. As the term “intelligent behaviour” is not only vague, but variable depending on domain and problem, it is well defined that any behaviour which is goal-oriented or to chase a particular goal or goals is also flexible. Such attitude demonstrates intelligence because it shows that the program has an awareness of the situations it desires to be in, and what changes it can affect to bring about such situations. Goal-oriented behaviour is one of the defining characteristics of an intelligent game playing program.
The remainder of this chapter starts with a brief description of what a computer game is, and then sets out some reasons why board games are suitable for AI research. This is then followed by a game scenario to demonstrate the kinds of problems the thesis aims to solve. Finally, the main aids of this thesis are presented and the structure of the remaining thesis gets outlined.

2.1 INTRODUCTION

Computer game playing is one of the very old areas of exertion in the field of Artificial Intelligence. One main reason for game-playing being called “an exciting activity” lies in the fact that for humans it still bundles intellectual activity with straight competition. It improvises thinking and learning that normally results in winning more games. Though it is learned that majority of research efforts in Computer Game-Playing have been concentrated on a lesser number of specific games (most especially Game of Chess, Checkers, Reversi, Backgammon and Go). This has resulted in the development of computer game playing programs that is capable of playing a solitary game very well. It does no play no other game. This way it can very well test out and refine players’ logical skills by making them play games against all opponents and measure the progress centered on the fallouts of the competition. Still enough research has not been done in the area of general game playing. There is still a growing need for further research on an overall theory of game playing. In this particular direction, there lies a regard for theoretical suitability of implementation. This thesis is a step in that direction. [56]

The same inspiration is accountable for much of the developed interest in Computer program based Game-Playing as a problem domain for Artificial Intelligence (AI): programs which think better, should play better, and so will win more games. Thus it can check out and refine diverse theories of intelligence by scripting many game-playing programs which symbolize these various theories and then play the programs against each other. It contemplates the more intelligent program to be the one that triumphs the most games.
While research in games has steered to many vital artificial intelligence (AI) breakthroughs over the preceding few eras, these have generally come through the study of classical games such as Chess, Go, Reversi and Checkers, and used as their benchmark for success the strength of the artificial player.

Game designers and researchers both are inclined to develop tool that can habitually measure the quality of a given game program. Such tool can be noteworthy beneficial to both. It could not only reduce development time by rapidly detecting failings but also cuts the need for extensive play-testing which can enforce designers to prematurely reveal certain development issues to external sources. Such tools or algorithms could have directed the automated search with new rule-book combinations, with a sharp view to suggesting fascinating avenues for designers to trail or even to fabricating complete game playing programs of meaningful quality. [57]

This thesis refers to that presumed link between winning games and intelligent behaviour as the competitive performance metric for acumen. Such a link would be beneficial for research, because it would mean that it does not have to resort to descriptive theory measures and could instead use competition to evaluate research success.

Many AI researchers working on games have assumed that such a link does exist and have focused their energies completely on building strong game-playing programs. Unfortunately, the use of such a link has proved problematic: ability to produce strong programs for some games through specialized engineering methods, the extreme case being special-purpose hardware, and through analysis of the games by humans instead of by programs themselves. Consequently, it now seems that increased understanding and computerization of intelligent processing is neither necessary nor sufficient for strong performance in game-playing. That is, it look as if that construction of strong game-playing programs without doing much of interest from an AI perspective is quite feasible,
and conversely, it can make significant advances in AI that do not result in strong game-playing programs.[58]

This is a significant problem for AI researchers working on games, and it raises the following questions:

1. How to evaluate good work in AI applied to games in the absence of performance?
2. Basic AI techniques such as learning, planning, and problem-solving should be useful for something—but if not for improved performance, then for what?
3. How to tell whether the performance of a program on some game is due to the universal success of the AI theory it symbolizes, or merely to the skill of the researcher in analyzing a specific problem?
4. Is it possible to discover some game for which enhanced performance on that game would be linked necessarily to improved understanding and automation of general intelligent processing?

2.2 AN OVERVIEW TO AI RESEARCH FOR GAMES

The general theme of this thesis will be the application of AI techniques to the field of computer games. Before the focus is narrowed to deal specifically with the problem of program design and construction, it will be informative to further scrutinize the reasons for carrying out AI research in the computer games domain. These motives stem from the potential benefits to academic AI research and computer game improvement of investigating new AI practices for games. [59]

2.2.1 General academic AI research

Computer games have a number of features which make them pleasing as a domain in which to pursue AI research. Computer game worlds are much related to the test beds developed by many researchers. This resemblance between game domains and existing
AI test beds means that by using games as test beds, researchers can tackle new problems in implemented simulations in game worlds without having to come to terms with entirely new problem domains. This list of comparisons also validates the wide variety of subject matter covered by computer games. Researchers looking to tackle a particular set of virtually realised physical conditions will not have to search for long before finding a game that provides them. [60]

Unfortunately, although many games are model environments for AI research, very few actually provide access to the internal features that would allow researchers to contribute to what is already in a game. Using game worlds in preference to custom test beds has a further benefit: computer games already have a diversity of useful features built into them. This allows researchers to proximately tackle the problems they are interested in without having to build their programs from the ground up, reinventing and re-implementing several “wheels” in the process.

A further advantage of chasing AI research in the computer games domain is that the character-based nature of most games encourages researchers to tackle AI issues that relate to whole agents rather than disembodied systems by situating their systems in an information processing architecture. A less theoretical benefit of pursuing AI research in games is that it has the latent to raise the public profile of academic AI, by being highly visible in an industry that worldwide generates over seventeen billion dollars’ worth of business a year.[61]

Because computer game worlds are simulations of random complexity, certain simulations will present problems to AI algorithms that estimate problems present in the real world. The advantage this provides is that solutions to problems in such game worlds will also be applicable to the same problems in the real world. [62]

The final benefit of pursuing AI research in games is also one of the core motivating factors of this thesis: AI techniques in most games must function in real-time, dynamic
environments. This means that any AI technique applied in a computer game sphere must be able to work within these domain constraints. This will advance the current state-of-the-art in AI. For example, techniques controlling agents within games must be able to answer to stimuli before the game player senses a delay and decision making processes must be aware that the state of the world can be changed at any time by other players.[63][64]

2.3 INTRODUCTION TO GAME

Since AI field’s inception, research has been channelized to develop intelligent or at least pseudo-intelligent game playing programs which plays computer strategy games. The game winning is the bench-mark parameter of excellence for strategy games and it is often considered as a sign of intellectual excellence. Many researchers do feel that an intelligent game playing program development would enable a big step towards evolving and enhancing more intelligent machine development era. There are certain fundamental and measurable existing game quality indicators. These fundamental indicators may be harnessed for attainment of the directed search for new high quality games. There are many ways to define a game; the most useful and appropriate is given by Salen & Zimmerman:

A game is a system in which players gets engrossed in an artificial battle, defined and dictated by rules that results in a measurable outcome. This definition was abbreviated from many prior studies based findings, most of which have been identified for the following key basics: play, rules and outcome. The goal of this thesis is to investigate the computational characteristics of games and puzzles such as Chess, Checkers, Go, Hex, Bridge, sliding-block puzzles, Conway’s Game of Life, . . . but what exactly is a game?

2.4 GAME THEORY
There are innumerable kinds of activities and situations that might be described as games. A large portion of these may be studied within the context of conventional game theory. However, the games considered are both more specialized and more general than what is conventionally addressed by game theory. More specialized, because they are only be concerned with determining the winner of a game, and not with other matters of interest in game theory such as maximizing payoff, studying cooperative tactics, etc. More general, because game theory is concerned only with the communications of two or more players, whereas there are still considerations of games with only one player (puzzles) and even with no players at all (simulations). Other differences are that game theory generally formulates games in either “strategic” form by exhaustively listing the strategies for each player or “widespread” form by analyzing the explicit game tree. But for the games it is mainly considered for both forms would be exponentially large, or even infinite. [65]

The game of chess is always been considered as an intellectual game par excellence; a touchstone of the intellect according to Goethe. The game has very high complexity which is based on its two main sources. First is the size of the search space: after the opening phase, each player has to select the succeeding move from approximately 50 possible moves on average. As single game itself consists of a few dozen moves in natural way, the search space is very-very huge. A second source of complexity stems from the amount of information confined in a single board. Here each player always begins the game with sixteen pieces of six dissimilar categories. The board comprises of sixty-four squares that evaluates a single board (a position) into involving highly elaborate computation, even moves are considered without look-ahead feature. [66]

The first working computer program that was capable of playing the game of chess have been planned for more than 40 years and it’s first full-fledged version was reported in 1958. According to Russell and Norvig’s report, there was an almost linear upsurge in the strength of computer chess programs between from 1965 to 1994 as it was measured in various human-rated tournaments based on their performances. This increase witnessed
climax position when World Chess Champion (former) Gary Kasparov defeated Deep Blue-IBM's special-purpose Chess engine, in 1997. Deep Blue and its descendants Deeper Blue, while playing the game rely primarily on brute-force methods that allow traversing as deeply as possible in the game tree. It gives these machines the advantage over its opponent. These programs have attained astonishing performance levels, Noam Chomsky has condemned their game-playing research logic by quoting as the fact that what a bulldozer can lift more than some weight lifters can. [59][67]

For a given board configuration, the number of feasible games possible (i.e., the size of the game tree), is very huge in size, even if it gets confined to endgames phase only. Endgames naturally comprise but a few pieces, the evaluation difficulty is still very hard because the pieces are free to move all over the board that results in a complex game trees having both deep and with high branching factors. Thus it gets proved that brute-force methods alone cannot be relied upon. It is needed to develop better ways to estimate the outcome of games with clever evaluation functions. The computerized learning of evaluation functions is an accomplished research area if it has to yield stronger artificial players. Genetic Algorithm paradigm can evolve board evaluation functions. The basic idea of GA is to breed computer programs that solve a particular problem. It always starts with a population of random and low-fitness individuals. Every individual member plays few games with its peers. Based on its level of success (or failure), it is assigned a score or fitness. Based on these individuals' fitness values, the next generation gets stochastically constructed. This process gets repeated to achieve the single best individual until it itself gets returned as the solution, at the time of the evolutionary program's termination. This is the path taken up by this thesis.

2.4.1 Combinatorial Games

In order to explore the connection between games and computation, games can be studied from a computational complexity viewpoint. Naturally, then, these games will have a combinatorial aspect. Indeed, they might be simply called “combinatorial games”, except
for the fact that there is already an recognized field called Combinatorial Game Theory and many of the kinds of games will be well-thought-out falling outside the domain of this field. A combinatorial game is well-defined to be a two-player, deterministic and perfect-information game with no chance elements. By so restricting the notion of game, Combinatorial Game Theory is proficient to draw out beautiful and unexpected relationships between games and numbers—indeed; in that framework a number is simply a special kind of game. But many other “games” exist which has combinatorial essence, such as sliding-block puzzles, or Conway’s Game of Life, or Bridge, violate these limitations, yet are interesting to study from a computational perspective. [68]

Combinatorial Game Theory has to be differentiated from all other forms of game theory rising in the economics background. Economic game theory also has many applications avenues in computer science. Popular examples are the forms of auctions and analyzing behavior on the Internet.

A combinatorial game characteristically contains two players and they are mostly known as Left and Right. They have alternate play with well-defined moves. There are some exceptions like a combinatorial puzzle. There is only one player in it. For the example of cellular automata such as Conway’s Game of Life, there are no players. In each of the mentioned cases, there is no arbitrariness or hidden information: means all players know all information about gameplay and thus these classes of games are known as perfect information. The problem is thus chastely strategic: how to carry play in best manner for the game against a faultless opponent. It is valuable and worthy to differentiate many types of two-player perfect-information games. A collective hypothesis is that the game enters termination state after a finite number of moves. Thus the game is finite or short, and it results into a unique winner. [69]

Combinatorial games have given many interesting and clean problems in algorithms and complexity theory. Out of which many are still uncluttered. The thesis has a clear goal to provide an overview of the zone to encourage further research into it. It takes particularly,
general background in Combinatorial Game Theory and then analyzes ideal play in games of perfect-information and Constraint Logic. Constraint Logic provides a framework to show-case hardness. Then assessing is conducted to determine complexity of ideal play in these games and the related solving puzzles problems, in both terms of polynomial-time algorithms and computational intractability results.

There are exceptions games also like Life and Chess can be drawn out forever. Some game cases are also evident for tic-tac-toe and Chess that can have ties in certain cases. In combinatorial-games generally a situation arises when winner gets defined by the notion of who is able to make last move and it is called standard play. But if, on the other hand, the winner gets defined by the first player who cannot move, this it is known as misère play. A game is called loopy if it has the possibility to return arrive at previously seen positions. Such example is there in Chess. Finally, if in a game two players (Left and Right) are both treated identically then it called unbiased and each player in the scenario has the same set of moves obtainable from the same game position; otherwise the game is called partisan.[70][71]

2.4.2 Cooperative, Collaborative and Competitive games

Traditional game theory put games into two primary classes: competitive or cooperative games. Competitive games have the requirement from players to form strategies. These strategies directly compete against the other players in the game. The set goals of the players are completely contrasting. [72] Many traditional board games like Chess and Checkers fall into this category. In contrast to it, a cooperative game has a model situation where two or more individuals have interests that are “neither completely opposed nor completely coincident”. There are existences of opportunities for players which make them able to carry collected set of work to achieve a win-win condition. A cooperative game does not always guarantee that cooperating players will benefit similarly or even benefit at all. Cooperative games has many rules which are enforceable and its main purpose is to negotiate or do brokering that will allow players to recognize
desirable outcome for all the participants involved. The classic and known cooperative game is the iterative version of the prisoner’s dilemma game example. Traditionally in prisoner’s dilemma, two prisoners are given provisions that if they defect each other and rat on their collaborator. Because of such type of reward system, both reasonably defect each other and ends up with a stricter sentence. If they would have collaborated with each other than produced result can be different. In the iterative version of the game, however, collaboration becomes a rational strategy. In a cooperative game, good guys (collaborators) can finish first, but they have to make sure that they are not being taken benefit of. Along with it, there also exist third category and it is sometimes not been recognized in game theory. [73][74]

*Collaborative game* is built on the concept of team work where all the contributors work together by sharing the payoffs and outcomes. In such scenario, if the team wins or loses then everyone associated wins or loses. A *team* is an group in which the kind of information each person has can differ, but the interests and principles are the same. *Collaboration* as a team differs from *cooperation* among individuals in that cooperative players may have different goals and payoffs where collaborative players have only one goal and share the rewards or penalties of their decisions. The challenge in a collaborative game lies in working together for all players by exploiting the *team’s utility*. [75]

Competitive and collaborative models form two extreme and contradicting end points of a given game spectrum. Competitive games avoid the concept of collaboration. Collaborative games are based on core aspect of collaboration. Now the question remains for cooperative games. They fall between competitive and collaborative games. Can collaboration be a sensible strategy in cooperative games? It seems that the answer would be yes. Though, because the fundamental game model is based on design to recognize a solitary winner, cooperative games can still bank on and encourage anti-collaborative philosophies in the participants. They include free riding and backstabbing as trivial examples for such practices. [76][77]
Free riding (e.g., when they are being assessed and rewarded as a group, individual group members do not pull as hard as they can) is a problem that causes group performance to suffer. Free riding sometimes appears in collaborative games also, but cooperative games do intensify this problem as defectors. They are often rewarded for their free riding behavior. Backstabbing is almost inevitable in many cooperative games. It is also the act of defecting when one’s partner starts cooperation. If backstabbing is done at a predominantly good moment then it can be a beneficial competitive exercise in an otherwise collaborative game. Here, the key to diplomatic policy is to establish the right kind of alliances. It is also to know when to backstab one’s buddies. In short, the best tactic in a cooperative game is to know when to act competitively.

Competitive behaviour in a given collaborative situation is not expected to occur in a collaborative game. One of the design problems in a collaborative game is to deal with the competitiveness that players carry to the table. [78][79]

A general game of two-player and perfect-information kind which does not have ties or draws can give four possible kinds of results. These outcomes are of the type of an ideal play and can be expressed as: either player Left wins or player Right wins. Either first player to move wins whether it is Left or Right or the second player to move wins. One main objective in exploring two player games is to regulate the conclusion as one of these four categories. It also aims to find a strategy for the winning player to win. Added objective is to compute a deeper association to games that is defined in the residue of this section.

Analysis of two-player combinatorial games has drafted another stunning theory based on mathematical model. Albert, Nowakowski and Wolfe has also introduced a novel introductory book on the topic called Lessons in Play; another most inclusive reference is the book called Winning Ways by Berlekamp, Conway, and Guy. A more mathematical book presentation is titled as On Numbers and Games by Conway. [80]
The root thought after the theory is modest: a two-player game can be described by a rooted tree. In the tree, each node has zero or more left branches corresponding to options for player Left to move. Simultaneously for player Right to move it can have zero or more right branches corresponding to move options; in the tree, leaves relate to finished games, where winner gets determined by the option of either normal or misère play. The Combinatorial Game Theory has such interesting parts that represent several methods for deploying and analyzing such games/trees. [81][82]

Combinatorial Games focuses on combinatorial games, which are:

- **Finite** that yields a well-defined conclusion.
- **Discrete** this is turn-based.
- **Deterministic** relies where chance has no part to play.
- **Perfect information** that contains no hidden information.

2.4.3 Two-player Games

The two-player game dilutes its two player requirement by for solitaire puzzles. It considers it as combinatorial games by considering that solver of the puzzle contests against the null player. Thus it indirectly challenges the designer who has set the challenge. Multiplayer games which have three or more players fall outside the space of combinatorial play as it has some social aspect of alliances that may arise.

The word *game* now should only refer to a two-player combinatorial game. Such games falls in category of perfect test bed for various game program based experiments. These experiments are having own search depth but can be described by simple, well defined rule sets. It is note-worthy that this work doesn’t get examined in combinatorial game theory. The said theory is only concerned with the games scrutiny along a view to solving them, or at least finding optimal strategies and emerging players with artificial approach which make them able to challenge human experts. This study context also limits the
artificial player as it has little interest to provide self-play simulations. There should also be adequate strength to provide meaningful play outs, which is primary concern for any quality of the game concentrating on it rather than the quality of the player. [83]

The most important point of consideration is that it has finite number of positions. This is what essentially distinguishes games from conventional models of computation, such as Turing machines. In principle a Turing machine is having an infinite tape; in principle, such a machine cannot exist in any finite physical space. The computers that sit on our desktops resemble more closely to space-bounded Turing machines. However, games with finitely many positions can actually exist in our world.

Next, undoubtedly there should be some number of players. The players take turns in some fashion, mapping one position into another by building moves. In the interest of including simulations such as the game of Life, and because such games fit naturally into the framework of zero player games; a zero-player game can be understood as a one-player game where all the moves are forced. Generally, adding players enhances more non-determinism to the corresponding model of computation.

For each game, the customary decision question will be along the lines of, “from this position, does player X have a forced win? The meaning of “forced win” is often not stated explicitly in game problem statements. However, the denotation may be taken to be, does player X have a policy—a rule dictating play in all circumstances—such that for all possible ways the other players play, player X wins? What specifically constitutes a strategy will depend on the kind of game. Also, it is assumed that when one player wins, the game ends, so no other player may then win as well.

Finally, it is likely to find possible and easy steps, to determine the legal moves from a given position. Some actual games skirt the boundaries informally placed out above. For example, some board game as played in China and in the USA has a superko rule, which stipulates that no former board position may be recreated. This rule guarantees that games
will not loop, and will eventually finish. Thus, it has practical usefulness. But it does mean that it is not possible to regulate the legal moves from the current position. Of course, the entire history of the game could be taken to be the current position, but this interrupts our intuition of what normally establishes a position. Also, complexity results would then have to be taken comparative to an exponentially greater space of possible histories, versus conceivable board configurations, and thus would not be predominantly interesting.

Perhaps, an equivalent game can be considered, where moves that recreate former positions are legal, but losing. Does this solve the problem, and with superko thus a valid game in the above sense? It is not so clear. It is worth noting that the difficulty of board game with superko is an interesting open problem; there is reason to trust it could be harder than board game with the traditional Japanese ko-rule.

Each category of game considered will be formally defined. But the notion of generalized combinatorial game sketched here is intended to be more heuristic, and potentially suggestive of kinds of games not treated explicitly.

2.4.4 Zero-Player Games

Study of specific abstract games with deterministic or zero-player games; which may be thought as game of simulations: each move is determined from the preceding configuration. Examples that are frequently thought of as games include cellular automata, such as Conway’s Game of Life. Generally, the class of zero-player games resembles naturally to ordinary computers or deterministic space-bounded Turing machines—the kinds of computation tools or quantum computers get available.

2.4.5 Constraint Logic
The Constraint Logic formalism does not confine the set of moves available on a constraint graph to an exclusive next move from any given position. To deliberate a deterministic version, further constrain of the legal moves is done. Rather than propose a rule which selects a unique next edge to reverse from each position, determinism is applied independently at each vertex, so that numerous edge reversals may occur on each deterministic “turn”.

The elementary idea here is that each vertex should allow “signals” to “flow” through it if possible. So if both red edges reverse inward at an AND vertex, then on the next move the blue edge will reverse outward. For the bounded version, this idea is all that is needed. For the unbounded version, the rule is altered to allow inputs that can’t flow through to “bounce” back. This allows the construction of arbitrary space-bounded computers that are based on unbounded Deterministic Constraint Logic and are called PSPACE-complete.

Furthermore, it turns out to represent a new style of reversible, monotone computation that could potentially be physically built, and could have significant advantages over conventional digital logic.

2.4.6 Bounded Games

A bounded zero-player game is basically a simulation that can only run for a linear time. Admittedly it seems a stretch to call such replications “games”, but they do fit naturally into the overall framework and all the other slots in that table are more possibly game-like. Bounded Deterministic Constraint Logic is included in this thesis merely for completeness. Possibly there could be some solitaire games where the player has no real choice, and the game is of bounded length, but such games not seem very interesting.

2.4.7 Games with Two-Players
With two-player games, they are finally in territory familiar to traditional game theory and Combinatorial Game Theory. Two-player, perfect-information games are also the richest source of existing firmness results for games. In a two-player game, players do make alternate moves, each trying to attain some particular objective. The standard decision question is “does player X have a forced win from this position?”.

The original hardness results for two-player games were PSPACE-completeness results for restricted games, beginning with Generalized Hex, and continuing with several two-player versions of known NP-complete problems. Later, when the notion of alternating computation was developed, there were tools to show unbounded two-player games EXPTIME-complete. Chess, Go, Reversi and Checkers then fell in rapid succession to these techniques.

The linking between two-player games and computation is quite manifest. Just as adding the notion of non-determinism to deterministic computation creates a new useful model of computation, adding an extra degree of non-determinism leads to the concept of alternating non-determinism, or alternation, discussed. Indeed, up to this point it is clear that adding an extra degree of non-determinism is like adding an extra player in a game, and seems to increase the computational complexity of the game, or the computational power of the model of computation. Unfortunately this progression does not generalize in the obvious way: from a computational complexity standpoint, simply adding extra players beyond two does not alter the situation in any fundamental way.

Alternation raises the complexity of bounded games from the one-player complexity of NP-complete to PSPACE-complete, and of unbounded games from the one-player complexity of PSPACE-complete to EXPTIME-complete. Since it is not known whether \( P = NP \) or even \( PSPACE \), with two-player games do finally reach games that are provably intractable: \( P \neq EXPTIME \). In each case there is a normal game played on a Boolean formula which is complete for the appropriate class. For bounded games the game is equivalent to the Quantified Boolean Formulas problem: the “existential” and
“universal” players take turns choosing assignments of successive variables. The unbounded games are similar, except that variable assignments can be changed back and forth multiple times.

<table>
<thead>
<tr>
<th></th>
<th>Unbounded</th>
<th>Bounded</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td></td>
<td>EXPTIME</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Unbounded</td>
<td>P</td>
<td>NP</td>
</tr>
<tr>
<td>Bounded</td>
<td>PSPACE</td>
<td>NEXPTIME</td>
</tr>
<tr>
<td>Zero Player (Simulation)</td>
<td>One Player (Puzzle)</td>
<td>Two Player</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Many classical games computationally intractable: one-player puzzle game like Minesweeper is often NP-complete or it can be PSPACE-complete like in Rush Hour. There are examples of PSPACE-complete games that are two-player games as in case of Othello or it can be EXPTIME-complete like games of Checkers, Chess, and Go. There are many simple puzzles and games that seem too hard. Other such outcomes can be positive and proves that some games can be played in polynomial time with optimality. In other cases like for one-player puzzles which are computationally tractable game and still holds interest for humans to play. [84]

2.4.8 Two Player- Perfect Information-Deterministic and Zero-Sum Games

A game is defined as a decision problem with two or more decision makers – players. Here the each player game outcome may depend on the decisions they make during the span of game. In a two-person game there are two players that make alternate moves. Examples of two-person games are Chess, checkers and Tic-Tac-Toe. A game where each of the players has complete information of the current situation in the game is called a perfect information game. Othello is perfect information game. Imperfect information game examples are card games because neither player knows what order the cards are in.
Table 2.2 Different Categories of games for four combinations

<table>
<thead>
<tr>
<th></th>
<th>Perfect Information</th>
<th>Imperfect Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Checkers, Chess, Go</td>
<td>Battleship, Mastermind, Stratego</td>
</tr>
<tr>
<td>Stochastic</td>
<td>Backgammon, Blackjack, Monopoly</td>
<td>Poker, Risk, Scrabble</td>
</tr>
</tbody>
</table>

A deterministic game is one in which the outcome of each move, or transition from one state in the game to another, is known beforehand. Deterministic game example is game of Chess. Both players have knowledge of move consequences when specified move happens from one game position to another. Backgammon falls in the category of non-deterministic game because neither player knows what the roll of the dice will produce and thus do not know what subsequent moves will happen on forthcoming turns.

A zero-sum game is a game in which one player's winning equals the other player's losses. If winning is counted as positives and losses in a game as negatives then adding them up will give sum as zero for each set of strategies followed then the game is a zero sum game. Almost all known games like Bridge, Chess and Checkers are zero-sum games.

### 2.5 BOARD GAMES

In this century, Board games have proved to be a significant research objects in the sciences. Games and board games were studied from a historical perspective. In 1944, Von Neumann and Morgenstern established field of game theory by providing basis for expanding games and board games in the branch of computer sciences. Research on Chess played pivotal role and charted path for all kinds of research on board games. These efforts were led by chess masters in particular. These got proved to be fundamental in the cognitive sciences from the days of de Groot. It was well followed by Newell & Simon and others researchers. Game of Chess is still foremost in majority of fields but
now other game championship has started entering these fields since last two decades and been set as good examples or tools in research.

Recently other board games research apart from game of Chess became possible and widespread. Days of Thomas Hyde and then after can be considered historical works on variety of board games. Even in 1952 Murray pioneered by printing A History of Board Games which was based on other games than Chess. But these researches did not ensemble to bring attention by making an important shift. These other games were also containing boards, pieces, rules, players and contexts unidentified to the educational world. These parts never got to be studied, as it became proven by the fact that first Ph.D.- thesis on the subject of draughts (or checkers) came to light only in 1997. [85]

Board games are a multifaceted form of games, consisting of boards and many types of pieces like dice, pawns, counters, etc. along with specified and well drafted system of rules and most obviously players. The board games playing environment also comprises of referees, spectators (interfering and non-interfering types), special occasions’ boards, formalities’ rules or rules of etiquette, club houses and societies and many more. A board game play presents certain kind of movement atmosphere, sound and other elements which are rather well described by poets than academics. Context with players, pieces, boards and rules are considered then it appears that these essentials cannot be detached for a complete understanding of a board game. The rules may impact the board and vice versa. The players may dictate the form and type of boards and rule specificity. They in totality make and define a complex ‘being’ which a board game is.

Board games in their very own intricacy poses the investigator with various inquiries like the (inter)relationship of the board game aspects are only understood in a small way. Historical growth and delivery mechanisms of board games have been a point of discussion. It was well started since historical works by Murray, Bell and contributed by Falkener and Hyde to name a few. Board games studies and related collections are exceptional and hardly ever accord with context and rules based fieldwork. The
accumulated results of studies, fieldwork, rule analyses and the study of players still need to be deliberated within their communication, addiction and significances for the uplift, expansion and spreading of board games. The classification methodology looks very fundamental for systemic answering to all these questions.

A conventional brute force game program describes a board evaluation function that heuristically measures the merit of each board position. It find the best computer response by thoroughly searching through a game tree, whose edges are the moves and nodes are the resulting board positions. The oriental board game does not succumb to this brute force approach, because of the high branching factor at the leaf nodes of the search tree, and a computationally intractable evaluation function. Reinforcement learning has been extremely successful for a few games through the use of evaluation functions like games of Checkers, Othello and backgammon. [86]

Some of the board games e.g., Chess, Reversi (Othello), Checkers, etc., are open to Shannon-A strategy, and the programs using this strategy are referred to as "conventional" game programs or "brute force" approach. They build huge game trees comprising moves as branches and positions as nodes evaluate the terminal nodes and backup the resulting values on to higher levels of the tree. At the highest level or root node, the program selects the move with the best value as its choice in the current position. A detailed account of the minimax search tree algorithm can be used in carrying out the mentioned process.

The game specific knowledge of a conventional program is located in the evaluation function and is usually simple to compute. This method has been used in developing Checkers and Reversi programs that are stronger than the best human players and in chess, the top rated programs beat 93" of human chess players. The mini-max game tree search for a 2-person adversary game has four components:
(1) Representation of the game state,
(2) Generation of the possible moves,
(3) Determination and recognition of the goal states, and
(4) A static evaluation function to determine the relative merits of states.

Fig. 2.1 Computer Game-playing with Genetic Approach

Programs are quite poor in the last three items of the above list, although they are fairly good at game state representations and move generation. Computer games not only symbolize as closed systems but they are also highly complex for dense study and in-depth analysis. With the use of strategies variations they can be well studied and played.
in a successfully way. Games being multifaceted and sometimes complex in nature is always challenging for the purpose of analyzing collaboration and game method for the purpose of nurturing success. In contrast to that the nature of board games entails a transparency regarding the core mechanics of the game and the way they are interrelated. This transparency makes them more reachable for in-depth analysis.

2.5.1 Unsupervised Game-Learning

A much smaller proportion of learning work has considered how programs might become strong players while relying neither on active analysis nor on experience with experts. Most of these approaches can be considered as *self-play*, in which either a single player or a population of players evolves during competition on large numbers of contests. A related technique, which can also be viewed as a form of self-play, is that of Abramson, who developed basic playing programs which learned to predict the *expected-outcome* of positions if played by random players.

This was shown to be effective for constructing evaluation functions for some games. In principle, approaches based on self-play could give fully satisfactory and general advice to the hypothetical researcher, for example:

2.5.2 Game Program Analysis

Use an off-the-shelf self-training technique. When given the game rules, suitable states representation and play strategies need to be chosen to analyze given game. Have programs play against each other many times, and by the time of real competition, select the strongest evolved program. It would seem that this advice requires minimal game-analysis on the part of the human. The major issues with this approach are as follows:

1. How much time is necessary to evolve a strong player?
2. How effective is the training method at developing good strategies on different types of games?
3. How much game-analysis must the human perform in order to design an appropriate representation?

With respect to the first and second questions, these are at present unanswered; while the methods appear useful for certain types of problems. Unfortunately, these questions are all very hard to answer. The reason being is variance with games, their representations, associated learning methods and volume of knowledge engineering required with each learning system. With respect to the third reason question in particular, it elaborated “fixes” associated with “trick” where many developers to find a game representation and its learning systems devote much of their time and logic to allow their systems to learn how to play it well. An extremely strong backgammon program using a training scheme which is claimed to be “knowledge-free” is produced, yet so far this method has been demonstrated only for one specific game (backgammon), with which the author was familiar.

2.5.3 Game Specialization

To summarize, concentrating on particular games can be disadvantageous for the following reasons:

Labor: Every time a program is developed much logical-human effort is needed for every different game to play. It comes with inadequate advice on the real problems.
Generalization: It is difficult to say what we have learned from our research, beyond performance on particular games.
Evaluation: Search evaluation is difficult. Game play if carries with good efficiencies then it is considered expertise on part of the researcher-developer that they have analysed the game well and not considered for the program. It is observed conversely that when
program gets developed on the line of general analysis and such programs tends to have lower quality play against highly-specialized machines.

Game Analysis: Learning programs can be written successfully even devoting much effort behind understanding the game analyse ability and it is one of the major thirst area in AI research for the field in game-playing. Despite of having field expertise to write computer programs very efficiently and developing programs of the level of game specific and championship level program against human world champion, it is still been forced to human developer-researcher to carry the perfect real time game analysis rather than relying logical and machine capability of the computer program.

2.6 A NEW CLASS OF GAME PROGRAMS

Many computer programs get created that are play-capable for different games. This new class of games is primarily having following assets:

a. It has large span that is sufficient to include many fascinating games;

b. It has description simplicity to describe which get them manageable in its own context to work with;

c. It has both types of games: deterministic and non-deterministic.

d. The class contains games that are zero-sum, two-person, perfect information.

Since beginning, novel game class has one basic idea that each player during his/her turn can move none, certain or all of their available pieces each turn. This results into very very high branching factor. If this scenario gets possibly applied in the game of Chess, then the average branching factor would shoot over 10,000 as compared to 35, which is a general value. The results that have given successful results in other game problem-domains might prove less acceptable for this domain.
Following are the game program factors which are important to consider while developing board game playing programs.

2.6.1 Simple Board

Basic Board Games has primary characteristic concern like moving pieces all around the span of the board. This also resolves in fight amongst pieces. Game board contains three major elements of consideration for any given game: the board itself, the pieces and resolution of move. Board game rule definitions are indigenous and game specific. The primary focus is to inculcate constituents which are very basic and universal to hefty number of existing games. [87]

2.6.2 Order of Play

Board games have alternate move policy amongst the two players. Each player can explore as few or as many of their pieces as they like during his/her turn. The move specific conditions can give following possibilities in game end scenario:
(a) One player has no pieces;
(b) One player has scored the winning number of points or more;
(c) The maximum number of turns for the game has been reached.

2.6.3 The Pieces

Players rely on pieces to alter the state of the game. Pieces are having have a specific board position and various attributes are attached and can be described for the importance of the state. Any given location may not have more than one piece. Every piece has some points which state its coherence capability it can withstand before getting removed from its location and/or board. Designated points for each piece would be one for simple game of Chess. Each piece also has an associated score that helps in determining the game winner. [88]
2.6.4 The Board

A board is characterized by group of positions where pieces can be placed. Regular matrix gets formed for depicting the locations with two types of formations. First type is *square* it has similarity with grid kind of structure which is very common with very popular set of games like chess, checkers, or Othello. The second type is *offset* which has hexagon board kind of arrangement and that is mostly used by military war games. It permits a more realistic movement representation. [89]

2.6.5 Movement

Each piece has an associated movement permission that elaborates the distance it can move or cover in one turn. Each board location comes with its movement cost. A piece can travel from one position to another provided it has enough movement points. In each turn, the player may move some specific or all of their pieces. One location to another movement is only permitted if the movement cost of the path is less compared to movement allowance of the piece. Movement points do not get protected from one turn to the next.

2.6.6 Fight

Board game like Chess has a very simple fight philosophy; when one player’s piece bout, the opponent’s gets eliminated. But in a realistic kind of game scenario some of the opponent warrior only got wounded, or three men in enemy squad got killed and five remaining soldiers got survived. Fight in this case may be either deterministic (like chess) or non-deterministic one. (E.g. rolling the dice and see the result) If the combat damage has more points than hit points of defending pieces, then defending piece can be removed from the board. Then attacking player’s score is augmented by the exact score value of the eliminated piece. [90]
2.6.7 Legal Moves

Each individual legal move is itself composed of a sequence of pseudo-operators, each of which are applicable in different stages of the move sequence. The pseudo-moves applicable at each stage are as follows:

Assignment: If the game begins with a decision-assignment stage (where players place pieces on the board before moving anything), the only legal move for each player is to allocate a piece and end his move. When there are no more pieces to assign, the stage is set to move.

2.6.7.1 Initial-promote

If the last player moved a piece into elevation region, and the piece has an opponent-promotes promotion power, then this move begins with the player now in control init-promoting the piece on its square, after which the stage is set to move.

2.6.7.2 Move

In an ordinary move, a player resolves whether to place or move a piece. The two operators, either of which can be selected, are as follows:

2.6.7.3 Place a piece

A player has a piece in-hand (i.e. one given to him as a result of an earlier possession capture-effect) he can now place it on a vacant square of his choice. This ends the move.

2.6.7.4 Move a piece

A piece is moved from one square to another, possibly enabling some capture effects. For games which have a global must-capture constraint, if some capturing moves are available, any one of them can be made; otherwise the player is free to make any non-capturing move. For games which have only local must-capture constraints, any piece can be moved, subject to the constraint that if it moves and has some capture choices, it
cannot make a non-capturing move. In this case, unconstrained pieces can make either capturing or non-capturing moves.

2.6.7.5 Capture
If the movement chosen ended in some pieces being subjected to capture effects, these are now executed. Capture effects can be of two types that may get applied to many pieces as a consequence of a given move:

- Removed pieces just disappear.
- Possessed pieces go to the hand and colour of the possessing player.

2.6.7.6 Carry on captures
If a piece that just captured is permitted to continue capturing, a next capture movement for it may be chosen but only if this actually captures, after which the stage goes back to capture. If the piece is essential to continue capturing when it is both allowed to continue, and forced to capture whenever it can, then if it can do so, it must. When a player cannot endure capturing, or decides to stop unless forced to continue, of course, the stage transfers to promote.

2.6.7.7 Promote
If a piece has just ended being moved as opposed to being placed, and is now on a square in promotion terrain for the moving player, the promotion effect is then executed. There are two types of promotion effects:

- Player-promotes: the moving player selects a legal piece to promote into (which depends on the piece definition), which replaces the original piece on its final square. The player’s turn then ends, and control transfers to the opponent, whose move will begin in the move stage.
- Opponent-promotes: the piece is detached from the board, and the player’s turn ends. The opponent will begin the next move in the init-promote stage, and will thus begin the next turn putting a valid promoted piece on the indicated square.
The set of legal moves available to a player in a position is thus the set of all possible sequences of legal pseudo-moves obtainable to that player which sack in a transfer of control to the other player. A move begins with one of the players being in control, after which a sequence of sub-moves is made, which ends in handover of control to the other player. A move-count is tracked throughout the game.

2.6.8 Moving and Capturing Controls

Pieces have separate powers of moving and capturing. A moving power describes a disjunctive set of movements, each with a disjunctive set of directions, based on the uniformities attached to the base direction of the movement. A capturing power defines a disjunctive set of capture definitions, each containing a set of movements, capture-methods, and capture effects. The main difference between using moving and capturing powers is as follows:

Moving Powers: It finds a final square such that there is a path from the current square using one of the defined movements within the moving power definition, and is presently empty.

Capturing Powers: It finds a comparable path, but using a movement within the capturing power definition, and after defining the seized pieces based on the defined capture-methods for this capturing movement, something must get captured and must convert empty if it was not empty already.

2.6.8.1 Moving Powers

It is True when player possesses a piece and there is some legal moving-power defined for that piece, a movement within that power, and a path within the symmetry-set for the movement, which can be used to move the piece from minus capturing anything. This
routine does not require that the piece is on the board, as apparently it has already been lifted.

Piece Movements: There are three types of movements, characterized as follows:

- **Leap**: The piece moves to the next square along a given direction vector.
- **Ride**: The piece moves along an open line of squares along a given direction vector.
- **Hop**: The piece leaps through a number of empty squares, then through a number of squares occupied by certain types of pieces, and then through a number at least 1 of empty squares.

Symmetric chess-like games have three types of goals:

**Arrival**: Achieved when a piece matching a description has arrived on one of a set of square.

**Eradicate**: Achieved when no pieces identical a description are currently on the board (even though they may be in the hands of a player). This only applies after the assignment stage has passed; else all such goals would be true at the start of games which have initial-assignment stages.

**Stalemate**: Achieved when a player is in control but has no legal moves. Goals are evaluated directly before each player makes a move. A game ends when either of the following is true:

1. Some player (possibly both) has achieved a goal, or
2. The game-specific maximum number of moves has been played.

When the game is over, the result is then based on which players have achieved one of their goals, or is a draw if the game-specific move boundary has been exceeded.
2.7 GAME COMPLEXITY

Game complexity can be measured by many parameters where one important one is game’s average branching factor. It is a number which shows regular quantity of moves that a player can make during his/her turn at any given game point. One other way is by measuring the game tree size and thus determining game complexity. Deterministic games related game trees shows game states through such nodes and moves through arcs. Game always starts with root node which depicts the initial state of the game. Tree leaves relate to terminal states. Non-deterministic games do have chance nodes which represent possibility variant of the game like dice rolls in game of Backgammon. Tree hierarchy in game tree uses one level for each single ply, or one player’s turn. [91]

Game tree size gets determined through design consideration of the average branching factor and raising it by taking number of ply power for the game generally lasts before ending. For example, game of Chess has on an average nearby 36 possible moves for each player. Thus, its average branching factor is 36. As the average chess game last for nearby 40 moves per player (or 80 ply) the game tree gets calculated at 3680 or around $10^{124}$. The entire game tree exploration, all the way down to all possible terminal states would give best move selection strategy for playing the entire game. But, breeding the complete game tree is completely infeasible for any complicated games, like Chess. [92]

2.8 EVALUATION FUNCTION

The search engine in a brute force program chooses a move that will leave it in the strongest position. Evaluation functions are used to determine this which is fairly simple formulas. Rank numbers can be allocated for each piece in game of chess like pawns can have number 1, knights and bishops are worth 3, rooks gets 5, queens is settled at 9. This value is multiplied by another digit representing the board position strength of particular piece. These heuristic rules give the computer a rough sense of the game state and a basis for the decision making. Position evaluations in board game require determining life and
death status of a group, which triggers tactical search, life and death search, even full board evaluation showed that determining the territory ownership from an arbitrary board position is a polynomial space (PSPACE – Polynomial SPACE) problem, and they proved it by drawing an analogy between a board position and the generalized geography game. Also, since the game is not bounded by a fixed length.

Any PSPACE-hard problem running in $f(n)$ space can take at most $2^n$ steps (for binary state-space $c=2$). For example, each of the grids on an $XxX$ board can take one of the three states. Thus, territory ownership can be computed in exponential time.

Generality: How do we make a program broad enough to play legally all the games in the class without human assistance irrespective of playing well?

![Fig. 2.2 Layers of Evaluation Function](image)

Efficiency: Does generality necessarily imply incompetence, or can the program make itself more efficient once given the rules of a particular game?
Search: Is it easy to build a robust player for this whole class using just game-tree search and a naive evaluation function, or are more sophisticated techniques essential?
Knowledge Acquisition: How to find general knowledge and strategies that might be beneficial for many games in this class, when at present player only have knowledge and strategies that are useful for precise games within the class?

Knowledge Representation: How to represent general knowledge to a playing program without knowing the particulars of specific game rules?

Competitive Advantage: Does the program provided knowledge and search actually give any competitive advantage in advance on unknown games to its programmer prior to competition?

By separating the assumptions relevant to a game-assumptive concept, it was in some cases probable to generalize the concept to apply to the class as a whole. Three significant and general features emerged in this way: mobility, centrality, and promotion.

2.8.1 Mobility

The concept of mobility is used in some respect in virtually all game-playing programs. The common factor in most mobility features is that they compute a set of properties which are necessary, but not always sufficient, for a player to have a legal move in a position. Both chess and checkers programs contain terms in their evaluation functions which count the moves available to each piece owned by each player in the current position. Note that this is not adequate to guarantee that the player really has those moves accessible. One example comes from checkers: though many pieces have conceivable movements, a player might be forced to capture an enemy piece, in which case he may in reality have only one legal move. Another example comes from chess: a player may be credited with a large number of queen moves, although none of them might actually be legal if the player’s king were in check. [93]

When trying to grow a general concept of mobility which could be used in this class, it was necessary to regulate whether mobility was always desirable, all else being equal, or if its validity expected some properties of the game. For example, in both chess and
checkers, a player wins by capturing all of the enemy pieces. Thus it seemed possible that for positive goals of this form, in which a player wins by weakening the enemy, mobility might be useful. In negative goals, in which a player successes by weakening himself, it seemed that mobility might be damaging. One example of such a negative game is lose-chess. In this game, each player must make a capture whenever possible, and the goal is to have no more moves. Another negative game is the ordinary version of Reversi, in which a player wins by having the most pieces on the board at the end of the game, but mobility often decreases with each piece a player captures.

After examining fruitful strategies in both the win and lose versions of these games, the opposite conclusion was reached. That is, mobility seems to be valuable for either type of goal, all else equal. Some evidence for this is that in both win and lose versions of Chess and Othello, the openings are almost identical regardless of the final goal. That is, both players struggle for increased mobility in the opening, as this gives them greater control. With greater control, they then go on to achieve advanced goals. Conversely, it turns out that attempt to reach the final goal in spite of reduced mobility often result instead in early defeat. For example, a player who directly tries to give away all pieces in lose-chess quickly winds up with only a few moves available. Although this means the player has almost attained the final goal, the opponent is then in such control of the game that he can force the first player to seizure the entire opponent’s residual pieces. [94]

2.8.1.1 Centrality and Eventual Mobility

The concept of mobility as used by current chess and checkers programs can be named immediate dynamic mobility. It is calculated by computing any number of moves that are immediately available to that piece / pawn in the present board position. The effect of this is to encourage programs to place pieces on squares where they have many moves, and to restrict the moves accessible to enemy pieces. However, this concept alone is inadequate for strong performance in two respects. First, it provides no direction when a player has reached a square which maximizes immediate mobility.
To help programs distinguish squares and pieces having the equal value in terms of immediate mobility, they are often provided with piece-square tables or centrality bonuses which provide bonuses for having pieces on more central squares. As programs play unknown games, it was necessary to enable them to construct and use similar tables directly from the rules of the game. This resulted in the eventual-mobility player and table, which are discussed in more detail later.

2.8.1.2 Static Mobility

A second problem with the traditional use of mobility is that it is merely dynamic, based on the present position. If this is used as the sole basis for defining the value of having a piece on a square in a position, it leads a program to intensely underestimate the value of pieces which are gridlocked in a particular position. Thus if an enemy knight attacks a program’s queen which has only one available move, a program using only dynamic mobility would not transfer the queen to its safe square. The reason is that in the location under deliberation, the queen would be worth only one point of mobility, while the knight would be worth several more. This problem was solved by the addition of a static-mobility guide, which credits a piece on a square with the quantity of moves it would have from that square on an otherwise empty board. Using this guide, a program would protect its queen against the attack as it realizes that the pieces blocking the queen may ultimately move, after which the mobility of the queen would be worth more than that of the knight.

2.8.1.3 Constrained Mobility

A further complication in mobility considerations is that while it may be legal for a piece to transfer from one square to another, this may in some cases always lead to direct loss. This came up when demonstrating Knight-Zone Chess and Chinese-Chess as instances of this class. In both of these games, pieces are accepted from a portion of the board by rules
declaring that the opponent wins whenever such a piece reaches one of the excluded
squares.

When presented with a set of game rules, the game-analyzing component of constructs a
set of *static analysis tables* based on the rules of that game. It gives Game Move selection
process as shown in fig 2.4 when later evaluating a position, while playing that game, the
evaluation function component takes as input these static tables and the rules of the game.
The program then hypothesizes a set of *dynamic analysis tables*, based on the existing
position. The dynamic tables, along with the static tables, the position and the game rules,
are then used by a set of *consultants*, each of which may provide specialized *advice* to be
used in evaluating the position. The whole set of advice is then passed to an arbitrator,
which combines it into a global evaluation of the position. This evaluation is a number,
which is the estimate of the position from the perspective of the *white* player. This
number is used by the search engine in the standard manner.

![2.3 Game Move Selection Process](image-url)
Fig. 2.4 Flow of Board Position Evaluation Process
2.8.2 Material Value

Specialists groups are employed to assign static material value to each piece type. This is later passed on to the player who actually owns each of the pieces in a given board position. This value does not rest on the position of the piece or of the other pieces on the board.

As a player’s score from these material values does not change whenever the piece makes an normal move, the effect is that the program is especially sensitive to moves which change the material balance. In chess and checkers, these changes are captures as pieces are removed from the board and promotions as the type of a piece changes.

One global consultant computes a set of material values for each type of piece in a given game, after the static investigation tables used by all the other advisors have been constructed. Weighted sum of the collected values get returned by experts to calculate the final material value for a given piece for this section.

2.8.3 Automatic Feature Generation

Automatic extraction or generations of game based features from the rules of games is recent major development in emerging programs. Such Chess based programs produce features which are based on counting each piece type. Same program analogy for the game of Othello produces features that are based on board position summary analysis keeping mobility in mind. [95] These methods can be operated on any set of problems which are prearranged in an extended logical representation, and are more general than the methods currently used by; however, these methods does not aim in producing features’ values but instead aid as input to systems which may learn their weights from practice or through observation of expert problem-solving. While ’s analysis is specialized to the class of symmetric chess-like games, and thus less general than these
other methods, it produces piece values which are narrowly worthwhile and worthy for such programs that does not perform any learning task. [96][97]

Figure 2.5 Genetic Game-playing Work flow

2.8.4 Evaluation Function Learning
Experience or observation can provide valuable work on various learning feature based values. These kind of passive analysis examples are not useful for program until program enriches itself with good experience with strong players.

Any game can be defined as a directed graph made up of nodes or states, \( s_1, s_2 \ldots s_n \), and rules for moving from one state to another. Some of the states are first states and some are terminal states. If game playing program is interested in learning to predict the "goodness" of a given position; two main approaches may be considered:

1. To learn from the positions this occurred in expert games
2. To learn by playing the game.

The first method is a supervised learning exercise, where it is implicitly made two assumptions:

1. A large number of documented games are available, where each move and some variation moves at each position are graded by experts
2. When human experts do make best of the moves. It can accept the second assumption in most of the cases; but the first assumption is far from true.

To consistently score moves and their variations by hand in thousands of games is almost impossible. The tactic of learning by playing the game requires the learner to solve a credit assignment problem, i.e., to divide the credit for wins (or, punishment for losses) among the moves made. This makes a reinforcement learning problem, because it learns the desired behavior through trial and error interactions with its own environment. [98]

2.8.5 Goal Determination

The terminal states are not well defined in board game. The game ends when both players pass in succession, and even many good board game players are not sure exactly when to leave the game. The terminal states are clearly defined in chess or tic-tac-toe. But in
board game, territory is increased by accumulation and is accurately determined only at the very end of the game. All the difficulties related to correctly predicting the life and death status of a group are responsible for making goal determination so difficult in board game. Like many human players. Board game programs, too, struggle in identifying the exact moment to abandon the attempts to save one's own group or to stop attacking opponent groups.

All these factors make computer board game very difficult. Experts have the human tendency to evaluate board state by capturing various pattern recognition skills. These are very rare and difficult to imitate through algorithms.

2.8.6 Selective Search

The idea is to continue promising lines of play to great depth, but to terminate bad lines early to avoid wasting time. Since a board game player, most often than not, makes a move depending on the stone shapes on the board; move selection is a problem of preferring one pattern of stones over another. Thus, we propose to train neural networks on high quality games by supervised learning to predict the expert moves. The goal is to keep the tree width around 30-40, making it the same as in chess. [99][100]

Thesis approach combines search methods with evaluation function algorithms and can be used in many other computationally challenging problems requiring search and evolutionary algorithms.

This thesis focuses on two major bottlenecks in computer board games, e.g., the formulation of an evaluation function and the selection of plausible moves for limited tree expansion. Machine learning approaches are proposed to explore to implement them. Machine learning is defined as the changes in a system which allows it to perform better the second time on repetition of the same task or another task drawn from the same population. [101]
Here the evaluation function is developed for 8x8 board game using techniques intensioned in game playing programs. But this research proposes some new ideas, previous' not tried in computer board game (which uses genetic algorithm), e.g., the utility function and the combination of experts.

The utility function in board game is defined as the profit or the loss incurred to a player by making a move. The determination of the exact amount of per step utility is as hard as calculating the board evaluation. A simple utility function, which issues a set reinforcement signal by comparing between the features of two consecutive boards, is proposed. Evolutionary learning through genetic optimization is used to learn the sum of future board position values at any position.

The approach advocates that several specialist genetic strings, each trying to learn only specific portions of an evaluation function, will learn a simple and linear function in learning to play board game on 8x8 boards.

An evolutionary machine learning model for computer board game is proposed. The idea is to separate the mechanisms of playing the game into genetic modules; where each module can effectively use the vast board game knowledge available through board position exploration. The games positions are searched using search algorithms in through principle game tree (the moves actually played) and the variation tree (alternate move suggestions).

To win, it is very important to plan in advance all the moves during the game, because the outcome usually depends from the first move, that is, from which the player will start. If player starts the game by placing his last counter in his store in his very first move, and he has another turn, the player can gain initial advantage. Likewise, by planning his subsequent moves, by adding up the captured counters and by predicting his opponent’s moves, the player creates a strategy that leads him to victory.
2.9 BOARD EVALUATION

Game program developers started exploring human board game play analytical skill by developing evaluation strategies in that direction. These strategies prefer to traverse fewer nodes but want to analyze each node more thoroughly. It avoids banking only on looking deep into the game tree. These strategies are based on partial look ahead only. The player in charge of making a move compute all possible and exposable board configurations that is accessible from the current board by making one legal move. After all possible boards’ positions evaluation, the one move position that attained the highest reward score gets selected and that move is made. This way an artificial player can get all board configurations accessible from the current position. Very popular philosophy combines an evolved board evaluator with a program to generate all possible next moves. Many game-strategies use this approach positively for different game evolving scenarios. [103]

2.9.1 Tree nodes

Game player studies various characteristics of the board when he/she undergo evaluating a board position. Some features are simple, while others require a profound knowledge of the game. It is observed that that game experts were better at recollecting meaningful board formations than beginners. This helped them to theorize game skill which is dependent on broader and deeper knowledge base which is indexed by huge board position patterns having familiarity amongst them.

Complex assemblies of game board are made up of smaller and simpler units. These units involve less game knowledge and can be interlinked in some way or the other. Game programs practice terminals that represent comparatively simple facets and functions. They also incorporate no game knowledge, but provide methods to combine those features. Search tree yields real number values minus certain arithmetic function descriptions. First, a large portion of present-day research is focused on machine learning and game theory. It takes perfect-information games particularly that revolve around in
research domains that induct logical rules to activate learning games. Second, expert players reiterated when consulted that various aspects of the board need to be considered while evaluating positions. Some aspects are more vital than others and logical operations on these aspects are natural way of performing. The same does not hold for mathematical operations. Third, it is noted that extremely large values are observed when numeric functions return values. These get restricted with indirect calculations. Thus series of board position values need to be carefully ordered. For any player, a simple strategy is to capture his/her opponent's piece when they are not sufficiently protected. This means that attacking pieces number for the player is more than number of protecting piece count of the opponent's. So player gets greater control over the number of pieces. It gives equal or higher material value of the shielding pieces than the player's. These considerations are not simple references and a terminal count that performs this comprehensive computational feat by itself also belongs to the subsequent assembly of complex terminals. [104]

The terminals are derived through comprising and refining process for logical resolution for many reasoning explained in previous paragraph. The issues it addresses are: Is an attack on any opponent's piece? How many pieces of the players are attacking that particular piece? How many protecting pieces for a given opponent's piece? What is the fitness material value of pieces which are involved in attacking and defending a given opponent's piece? All these mentioned questions are personified as terminal questions inside the second group. It gives capability to effortlessly represent such kind of reasoning through functions and terminals that to within the GA setup. It is a major asset of GA.

2.9.2 Fitness evaluation

Success of fitness landscape for evaluation function of an individual can be determined only by comparing them by its accomplishment against its peers. To achieve it, a fixed number of games are played and these encounters start from a randomly generated
position. Sometimes these random starting positions are uneven for example that permits the opening player to attain a seizure position. Sometimes, every starting position gets played twice for each player who plays with both black and white.

The outcome of the game gives the score for each game. Players who manage to mate and tame their opponents earn more points. It is way ahead than those that achieve only a material advantage. Draws are compensated by a low score value and no points at all for entailing losses. Mostly final fitness for each player is the sum of all points he/she received during entire tournament span for that generation. It uses the standard evolutionary cycle comprising of reproduction, crossover and mutation operators. The major parameters of evolutionary processes are: population size, reproduction probability, crossover probability, mutation probability and generation count.

The thesis work has led to the extension both research areas of Artificial Intelligence and Machine Learning where each game node and problem domain aspect get exposed to the control of genetic optimization process to generate a global solution.

2.9.3 Building a Game Playing Program

Since comprehensive search of the entire game tree is usually not feasible. So many other techniques have been developed that rely on partial search of the game tree. Majority computer game playing programs practice variant of popular tree search algorithm called Alpha-Beta algorithm. The algorithm is a proven valuable tool for the category of zero-sum, two-player, deterministic games with perfect information. Since its inception since 1960’s, its basic structure has changed bit-little. Numerous algorithmic based on it has been developed and got enhancements also to improvise the search efficiency.

Developers have used game database to expand game playing programs’ performance. Game position information which is stored in a database is a base on which games are drawn, won or lost. The endgame databases create impact to successfully outspread the
search ability for game playing programs. This thesis highlights the use of genetic move-making optimization in games like Othello and Checkers.

As earlier foreseen also that it is usually not conceivable to search the entire game tree, it is usually necessary to find evaluation index or value to analyze different positions in order to select the next move. This is done with an *evaluation function*. The evaluation function returns an estimate of the expected outcome of the game from a given position. The performance of a game-playing program is reliant on the quality of the evaluation function. If it does not accurately reflect the actual chances of winning, then the program will choose moves that lead to losing positions. The actual numeric values of the evaluation function are not important so long as a better position has a higher value than a worse position.

2.10 HISTORY BOARD GAME PLAYING PROGRAMS

Popular board game and their playing programs are listed here.

2.10.1 Chess

Deep Blue is arguably the best computer Champion. It uses a relatively simple evaluation function with a 10-ply search. Custom built for chess by a team of IBM scientists, Deep Blue weighs 1.4 tons, and has 32 microprocessors that give it the ability to look at 200 million chess positions each second.

2.10.2 Backgammon

With 30 pieces, 26 locations and all possible combinations of the dice roll, backgammon has branching factor of several hundred per ply. Tesauro's TD-Gammon has attained a Champion level of performance using a neural network, trained by self-play. The
program is better than any other computer program playing backgammon and plays at a level nearly equal to the world's best players. TD-Gammon uses only a 2-ply search.

2.10.3 Go

Go has nine rules, two kinds of pieces and a 19x19 board. Despite its simplicity, it has an average branching factor of 250 and the size of the game tree is about 10360. Computer programmers have found it challenging to create programs that can compete against average players. Go4++ and Handtalk are among the strongest of all go programs. Go4++ uses a process of matching 15 high level patterns with the current game state to generate about 50 candidate moves that are then analyzed to find the best move. Handtalk also uses pattern matching to evaluate a very small number of candidate moves.

The following are two board games which are under consideration of the thesis.

2.10.4 Game of Checkers

The standard 8 × 8 game of Checkers (Draughts), like many classic games, is finite and hence can be played optimally in constant time (in theory). Indeed, Schaeffer et al. recently computed that optimal play leads to a draw from the initial configuration (other configurations remain unanalyzed). The outcome of playing in a general n × n board from a natural starting position remains open. On the other hand, deciding the outcome of an arbitrary configuration is PSPACE-hard. If a polynomial bound is placed on the number of moves that are allowed in between jumps (which is a reasonable generalization of the drawing rule in standard Checkers, then the problem is in PSPACE and hence is PSPACE-complete. Without such a restriction, however, Checkers is EXPTIME-complete.

2.10.4.1 Checkers Playing Rules
Checkers (also known as “draughts”) is played on an eight-by-eight board with squares of alternating colors. There are two players who take sides denoted by “red” or “white” (or “black” and “white”). Each side has 12 pieces (also called checkers) that begin in the 12 alternating squares of the same color that are closest to that player’s side, with the rightmost square on the closest row to the player remaining open (Fig. 1). Checkers move forward diagonally one square at a time or, when possible, may jump over an opposing checker into an empty square. Jump moves are forced, although if more than one possible jump is available, the player may choose which jump to execute. When a checker advances to the last row of the board, it becomes a king and may move forward or backward diagonally. The thesis takes this form of rules for as its test bed.

The game ends when one side cannot make a legal move, which is most commonly brought about by removing that player’s final piece. The player who cannot move loses and the other player wins. Alternatively, 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.

Fig 2.6 Initial Checkers Board

2.10.4.2 History of Checkers Game
Checkers, as it is known in Great Britain, has ancient roots. It is thought that the earliest form of checkers was a game discovered in an archeological dig at Ur in Iraq. Carbon dating makes it appear that this game was played around 3000 B.C. However, the game used a slightly different board, a different number of pieces and no one is quite certain of the exact rules.

In ancient Egypt a game called Alquerque, which had a 5X5 board was a common and much played game. Historians have traced it as far back as 1400 B.C. It was a game of such fame that it was played all over the western world for thousands of years.

Around 1100 a Frenchman got the clue of playing the game on a chess board. This meant expanding the number of pieces to 12 on a side. It was then called "Fierges" or "Ferses". It was soon found that making jumps compulsory made the game more challenging. The French called this version "Jeu Force". The older form was considered more of a social game for women and was called "Le Jeu Plaisant De Dames".

Now the rules for checkers were established and the game got exported to England and America. In Great Britain the game was called "Draughts". Books were written on the game in Spain as early as the mid-1500s and in England a mathematician name William Payne wrote his own treatise on Draughts in 1756.

Types of Checkers: There are more than 150 variants found worldwide. Some of them are:

2.10.4.3 International Draughts

In the International draughts variety of checkers, the game is played on a 10x10 board with 20 checkers pieces being given to each player. In this variant, kings are permissible to move across several squares just as long as the squares are open. This rule is also commonly known as "flying kings". If any player has the option to take more than one path to jump and capture his or her opponent's checkers pieces, he or she must take the option that will result in the capture of the most checkers pieces. If any checkers piece lands in the king’s row during a jump, it must continue with another capture backward if
the option is available. If the move does not end with the checker in the king row, it will not be crowned even though it has passed through that row.

**Canadian Checkers**
This variety of checkers is played on a 12x12 board with 30 checkers pieces given to each player. However, the rules are the same as those of International draughts.

**Brazilian Checkers**
This type of checkers is played on an 8x8 board. Again the rules are just the same as International draughts.

**Italian Checkers**
This checkers type is played on an 8x8 board, with the main difference from British-American checkers being that regular checkers pieces are not allowed to capture kings.

**Chinese checkers**
Although it shares a similar name, Chinese checkers is not actually a checkers variety, and is played on a star-shaped board with marbles or pegs.

2.10.5 Game Complexity

The game of checkers has roughly 500 billion- billion conceivable positions ($5 \times 10^{20}$). The task is very formidable to solve the game, determining the finishing result in a game with no error made by either of the player. Since last three decades, almost ceaselessly, dozens of computers have been working on solving Game of Checkers, applying state-of-the-art soft computing based techniques to improve the learning process.

Game of Checkers represents the most computationally challenging game to be solved to date. Evolutionary Learning challenges in Game of Checkers are:
The space to be searched is huge. It is estimated that there are up to $5 \times 10^{20}$ possible positions that can be searched. So any search algorithm based method which is based on exhaustive search for the problem space is infeasible.

The search space bulk is not smooth and straightforward. An evaluation function’s parameters which are feature construction based are highly inter-dependent. In some cases increasing the values of optimization parameters will result in an inferior performance, but many a times the controlled set of evolutionary parameter is also upsurges performance, then an improved overall performance would be attained.

The problem is not well understood by researchers. Even though all top performing programs parameters are hand tuned by their program designers, finding the best value for each parameter is mostly based on operational genetic alternatives.

High-performance game-playing programs have been a main success story for AI. In games such as Chess, Checkers, Othello, and Scrabble, mankind has been humbled by the machine. However, although these programs are very strong, they can still drop a game to a human. They are strong, but not perfect. Perfection requires one to “solve” a game.

Allis defines three levels of solving:

1. Ultra-weakly solved. The game-theoretic value for the game has been determined. An ultra-weak solution is mainly of theoretical interest. For example, Hex is a first player win, but no one knows the winning strategy.

2. Weakly solved. The game is ultra-weakly solved and a strategy is known for achieving the game-theoretic value from the initial position, assuming rational computing resources. Several well-known games have been weakly solved, including Connect Four.
3. Strongly solved. For all likely positions, a strategy is known for determining the
game-theoretic value for both players, assuming reasonable computing resources.
Strongly solved games include 6-piece chess endgames and Awari.

Note the qualification on resources in the above definitions. In some sense, games such as
chess are solved since the minimax algorithm can in principle determine the game-
theoretic value; given a huge enough amount of time. Resource constraints prevent such
impractical “solutions”. How difficult is it to solve a game? There are two dimensions to
the: decision complexity, the struggle required to make correct decisions and space
complexity, the size of the search space.

Checkers is measured to have high decision complexity and moderate space complexity.
All the games cracked thus far have either low decision complexity, low space
complexity and in the United States. The rules are simple (pieces move one square at a
time diagonally, and can capture by jumping) and the number of piece types is small
(kings and checkers), yet the game is fairly challenging. Checkers has high decision
complexity (more complex than 9x9 Go and the play of bridge hands, on par with
backgammon, but less complex than chess) and moderate space complexity (10^{20}
positions versus unsolved games such as backgammon, 10^{19} positions, and chess, 10^{44}
positions).

The best checkers programs are stronger than the best human players (e.g., CHINOOK
won the World Man-Machine Championship). The number of possible placing of
checkers pieces on the board is roughly 5\times10^{20}. This number is deceptive since it includes
positions that are not legally reachable from the start of the game. For example, although
there are 9\times10^{19} possible positions with 24 pieces on the board, only a small fraction can
be reached in a game (e.g., 12 kings versus 12 kings is not possible).

Our effort to solve the game of checkers began in 1989! After virtually 16 years of
research and development, and countless thousands of computer hours, we are pleased to
report the first major milestone in our quest. The White Doctor opening (shown in Figure 2.6) has been proven to be a draw. This is a difficult opening, as Black begins the game with a huge positional disadvantage so large that humans consider Black’s best play to be to sacrifice a checker early on to relieve the pressure. Notably, this opening was played in the decisive game of the 1992 Tinsley–CHINOOK World Championship match.

2.10.6 Chinook

Chinook is the world’s sturdiest computer checkers player, and the second strongest checkers player in overall. As it is a highly optimised and specialized program, it is not surprising that always loses to it (on checkers, of course!) when playing an even game, even when Chinook plays on its easiest level and responds virtually instantly, and without its opening or endgame database. However, to get a baseline for Chinook’s performance relative to other possible programs when playing against Chinook, it is evaluated that when programs are given various piece handicaps (number of men taken from Chinook at the start of the game), in order to regulate the size of the handicap and necessary to draw with Chinook on its easiest level. The results were that contends evenly with the weakest possible Chinook (without opening or endgame databases, searching 5-ply, and responding almost instantly) when given a handicap of one man.

This is compared to a deep-searching greedy material program’s search engine using the minimal evaluation function with only the feature for general material difference, which requires a handicap of 4 men, and to the random player, which requires a handicap of 8 men. These experiments were by no means thorough, and are only provided to indicate that drawing with Chinook with a one-man handicap even on its easiest level is by no means easy. [104]

On the other hand, certain simple questions about Checkers can be answered in polynomial time.
Can one player remove all the other player’s pieces in one move (by several jumps)? Can one player king a piece in one move? Because of the notion of parity on \( n \times n \) boards, these questions reduce to checking the existence of an Eulerian path or general path, respectively, in a particular directed graph. However, for boards defined by general graphs, at least the first question becomes NP-complete.

In the case of Chinook, “knowledge” is generally encoded in terms of its evaluation function. Apparently, at the US Open, the move that was said to be positionally “best” by the program after a 5-ply search (which is the minimum search depth in Chinook) turned out to be the recommended move even after a 20-ply search, implying that the strength of the program was not based on wholesome search (although it does make the difference when it comes to competing at an expert level).

Additionally, Chinook was able to correctly forecast the opponent's move 80% of the time, which would have been ever higher, if the mistakes made by human opponents were factored in. In computer chess, an estimate rate of 50% would be really good, but the fact that the number of pieces and squares in checkers are less makes it easier to predict moves too, as compared to chess.

Chinook divides the game into five phases - opening, middle game, early endgame, late endgame and database. The first four stages each have twenty two user-adjustable parameters that influence the evaluation function. A position evaluation is the linear sum of the 22 heuristics multiplied by their user-defined weights. The last phase has flawless knowledge, and therefore has no parameters. The parameter settings used for the US Open were determined manually. A database of 800 classic Grandmaster games was created and used to develop and automatically tune the program's knowledge.

Initial attempts revolved around tuning the parameters in order to exploit the number of times Chinook would select the same move as the Grandmaster, over all games. The very same practice employed by the Deep Thought chess team - which treats the problem as
an over determined system of equations to be resolved where one dimension is the number of positions used, and the other is the number of heuristic parameters - was used by the Chinook team, too.

It was found, however that more often than not, Chinook selected substitute moves that were either as good as, or even superior to, the Grandmaster moves - which adds to its strength by being able to select moves without partialities of human preconceptions, thereby allowing Chinook to continue to surprise the human opponent.

Endgame databases are a computer-generated collection of positions with a proven game-theoretic value, and were established in Computer chess. At the US Open, Chinook had access to the entire 6-piece database, comprising roughly $2.5 \times 10^9$ positions. What this meant is that whenever a position with 6 pieces or less was encountered, the program could look it up in the database, and return an exact win, lose or draw value. In computer chess, endgame databases are of partial utility since most games never get far enough to utilize them.

In checkers, however, since capture moves are forced, some positions near the start of the game can be scrutinized right to the end of the game. For instance, a nominal 15-ply search at the start of the game is sufficient for Chinook to start reaching out to positions in the endgame databases. At the time of writing, the Chinook team had announced that they had succeeded in building the entire 10-piece database. Computing such huge databases and efficiently representing them in a dense form that can be used in real-time are both challenging issues.

The endgame databases represent perfect knowledge that does not necessarily guarantee perfect play. Chess databases often store distance to win (or mate), for each position. Once in a winning database position, the program can win by only playing those moves from inside the database that minimize the number of moves to win. Since the checkers databases are used throughout the game (even at the beginning), they have to be
compressed enough to fit into RAM. To enable this, the Chinook team only stores the results of a position, not its remoteness to win. This means that Chinook could find itself in a winning 6-piece position, but will still have to search to find the winning line. A lot of techniques have been strained to reduce the size of the endgame database, including neural networks to build an evaluation function that can predict the game theoretic value.

2.10.7 Game of Reversi

Reversi is a classic game on an $8 \times 8$ board, starting from the initial configuration shown in Figure 2.7, in which players alternately place pieces of their color in unoccupied squares. Moves are restricted to cause, in at least one row, column, or diagonal, a consecutive sequence of pieces of the opposite color to be enclosed by two pieces of the current player’s color. As a result of the move, the enclosed pieces “flip” color into the current player’s color. The winner is the player with the most pieces of their color when the board is filled. Generalized to an $n \times n$ board with an arbitrary initial configuration, the game is clearly in PSPACE because only $n^2 - 4$ moves can be made. Furthermore, Iwata and Kasai proved that the game is PSPACE complete.

Fig 2.7 Initial Reversi Board
2.10.7.1 **Rules of the game**

1. The game begins with black discs on d5 and e4, and white discs on d4 and e5, as shown in Diagram 1-3.
2. Players alternate taking turns, with Black moving first.
3. A legal move consists of placing a new disc on an empty square, and flipping one or more of the opponent’s discs.
4. Any of the opponent’s pieces which are ‘sandwiched’ between the disc just placed on the board and a disc of the same color already on the board should be flipped.
5. Sandwiches can be formed vertically, horizontally, or diagonally. To form a sandwich, all of the squares between the new disc and the disc of the same color already on the board must be occupied by the opponent’s pieces, with no blank squares in between.
6. Pieces may be flipped in several directions on the same move. Any pieces which are caught in a sandwich must be flipped; the player moving does not have the right to choose to not flip a disc.
7. A new disc cannot be played unless at least one of the opponent’s discs is flipped. If a player has no legal moves that is, if no matter where the player places a new disc he could not flip at least one disc, that player passes his turn and his opponent continues to make consecutive moves until a legal move becomes available to that player.
8. If a player has at least one legal move available, he must make a move and may not pass his turn.
9. The game continues until the board is completely filled or neither player has a legal move.

2.10.7.2 **Scoring**
Scoring is done at the end of game. The usual way to determine the score is to simply count the number of discs of each color, e.g., if there are 34 black discs and 30 white discs, then Black wins 34-30. If both players have the same number of discs, then the game is a draw.

In tournament play, if one player captures all of his opponents’ discs, the game is usually scored as a 64-0 victory for that player, regardless of the number of discs on the board. Further, in certain tournaments, such as the World Championship, empty squares are awarded to the winner. For example, if at the end of the game there are 32 black discs and 29 white discs, with 3 empty squares, the score is recorded as a 35-29 victory for Black.

2.10.8 Othello Playing Programs

The game of Othello has received great attention within Computer Science for more than ten years. The following is the far from complete outline of several pieces of research work in this domain.

2.10.8.1 IAGO

It was Paul Rosenbloom who pointed out that although the game of Othello has an average branching factor 5 and limited length (less than 64 moves) it still cannot be solved exactly and has a great deal of complexity to be a subject for scientific analysis. Rosenbloom analyzed the game of Othello into a pair of major strategic concepts (stable territory and mobility), each decomposable into sub-concepts.

Quantitative representations of these concepts were combined in a single evaluation function. Together with the alpha beta search algorithm, iterative deepening, and move ordering, this function formed the basis of Rosenbloom’s program IAGO. Although IAGO displayed a World Class performance it did have several drawbacks.
• First, the concept set used by the program is rather limited.
• Second, the concepts, also called features, were assumed independent and therefore combined into a linear evaluation function.
• Third, the application coefficients in the evaluation function were selected by hand, which left a significant margin for error.
• Fourth, IAGO used a single evaluation function for the whole game, though it is now well known that different strategies are needed for different stages of the game.

2.10.8.2 BILL

K.-F. Lee and S. Mahajan addressed these drawbacks by creating a program named BILL, which used a slightly extended set of features. These feature representations were significantly improved through use of pre-computed tables that allowed BILL to recognize hundreds of thousands of patterns in constant time. The authors applied Bayesian learning to combine concepts in BILL’s evaluation function, which directly estimates the probability of winning. BILL learned several evaluation functions, one for every move between the 25th and 48th, inclusive. These evaluation functions were trained using a large database of 3000 games created by an earlier version of the program.

These properties and improved search techniques and timing algorithms allowed BILL to surpass completely IAGO, but as I mentioned, BILL requires a big game database for training, and the quality of the evaluation function completely depends on the quality of this database. BILL’s evaluation function is a quadratic polynomial, which takes into account linear inter-relationships between features; the question of existence of more complex interactions among these concepts remains open.

2.10.8.3 LOGISTELLO
LOGISTELLO has been one of the top Othello programs ever since its tournament debut in October 1993. It is still a sequential C program running on ordinary hardware from the beginning. The main focus of development has been on deep searches and reasonably good evaluation functions. In what follows, the latest program improvements are described. First, a new table estimation technique is presented which significantly improved the evaluation function. Quality at no additional run time cost. The selective search procedure PRoBCuT is generalized enabling the program to cut off even more variations in advance that probably have no impact on the move decision.

LOGISTELLO’s previous evaluation function details of LOGISTELLO’s previous evaluation function recently have been described in what follows. A brief outline of the techniques used is given allowing the comparison of the major aspects with the new method.

The classical approach for constructing evaluation functions for game-playing programs is to combine win correlated evaluation features of the position linearly:

This type of evaluation function is chosen very often since the combination overhead is relatively small compared to the time for computing the features and there are efficient methods available for determining the feature weights.

LOGISTELLO’s previous evaluation features fall into two classes, namely mobility measures and patterns. These approximate important concepts in Othello, like striving for stable discs, maximizing the number of moves, and parity. ROSENBLOOM and LEE & MAHAJAN introduced a table-based evaluation scheme, in which values of all edge configurations were precomputed by (probabilistic) minimal algorithms and stored in a table for a quick evaluation of the edge structure.

Furthermore, several local mobility features defined on the lines of the board (horizontals, verticals, and diagonals) were evaluated by fast table accesses. The pattern
approach introduced in generalized this technique by permitting the automatic evaluation of pattern configurations of any shape. The second feature subset dealt with mobility and potential mobility. Here, the simplest approach is to count legal or potential moves which unfortunately are relatively time consuming compared to the time needed for all other features and making/undoing moves during the game-tree search. In order to speed up the computation the globally defined mobility measures were approximated by the sum of mobilities local to the lines of the board, i.e. the horizontal, vertical, and diagonals.

2.11 SUMMARY OF THE CHAPTER

Game-analysis is the general process that exploits the abstract depiction of a specific game to produce a competitive advantage on that game. It is the course which relates structure to supremacy in game-playing, and is of central importance to Artificial Intelligence (AI) as a discipline of intelligence. Some of the problems currently faced by the field of Computer Game-Playing stem from the fact that, in all existing work done in this field, humans have full information about the rules of the games their programs play. This makes it implausible to infer, even from strong performance on those games that the theories which a program embodies are suitable to anything but the specific games the program has played.

In particular, success on a known game is no indication that the program, and not its programmer, has achieved game analysis. Therefore, success on a recognized game is not indication that the program is stimulating from an AI perspective. These problems are eased by employed within the new paradigm of Game Playing in which rather than designing programs to play an present game known in advance, program are considered and developed to play new games, from a well-defined class, captivating as input only the rules of the games produced by a game playing program.

The performance criterion is still competition: all programs ultimately compete in a competition, at which point they are provided with a set of games produced by the
creator. The programs then compete against each other through many competitions in each of these new games, and the winner is the one which has won the most competitions by the end of the tournament. While humans still have complete information about the generator and thus can still perform analysis instead of the program, the particulars of the specific games must be preserved by the program alone. The class can also be made more general in a controlled fashion as scientific knowledge advances. Analysis of the class of games as reserved by the generator showed that it measures reasonably well in terms of coverage, diversity, structure, varying complexity, and extensibility.

The conclusion from this is that the problem of board game is a good instance of a game research problem, and that modest performance on this problem will be evidence of increased general capability in game-playing. The results of an extensive human analysis of the class of games have been personified in a program, called, which plays –board game.

The program takes as input the rules of a detailed board game and a set of parameter settings (or weights), and studies the rules to build
(a) A more well-ordered representation of the general semantics of the class of games, specialized to just the input game
(b) Its own evaluation functions for that game, for use with a broad search engine. The strategic analysis performed by the program narrates a set of general knowledge sources (called consultants) to the particulars of the particular game in a manner which can be observed as a form of knowledge-compilation.

Among other properties, game’s analysis determines the proportional value of the different pieces in a given game, and for each piece the relative value of placing it on different board squares. Although does not acquire from experience, the values resulting from its analysis are qualitatively equivalent to values used by experts on known games, and are suitable to produce modest performance the first time the program actually plays
each game it is given. This appears to be the first time a program has derived useful piece values directly from analysis of the rules of different games.

An extensive experiment was carried out to assess the strength of different versions of across a set of unknown games produced by the creator. The conclusions from the experiment were that (a) the knowledge implemented in is useful on games unknown to its programmer in advance of the competition, (b) represents the state-of-the-art in board game and is a useful starting point against which to check future developments in this field, and (c) future programs that include learning and more sophisticated active-analysis techniques will have a demonstrable competitive advantage on this new problem.

Examination of the performance of the known games of chess and checkers, when playing against humans and specialized programs, suggested that has a reasonable positional sense for both games. It originates from more general principles some policies which are familiar to players of those games and which are hard-wired in many game-specific programs. However, is weak in strategies compared to the specialized programs, and also weak in logical problem solving and planning. These are therefore talented areas for future work.