AI for Board Games- Latest Developments

Dennis Kolomiyets, Jai, Konan Heney, Rohan Arya

Abstract— An intelligent system is anybody or being that has a perception of life and its surroundings and is aware to these beings. However, his belief in the possibility of machines being programmed to follow certain rules had marked the start of the journey of Artificial Intelligence and Intelligence programs. Shannon deduced that if the self-playing chess engine, at the time, were to run and play an entire game of chess itself, it would be required to make decisions that often would be based on incomplete and unsure information as the confirmed checkmating positions might be about 50 moves ahead of the current position. Instead, a three-move opening is drawn from a stack of approved beginnings, which give some tiny advantage to one or the other player. GO deals with massively huge numbers, so for us to get a better understanding of the system, we use log (base 10). Support All units individually are the same strength, which would make the capture of a province impossible if these settlements have any defenders.

I. WHAT IS AN INTELLIGENT SYSTEM?

An intelligent system is anybody or being that has a perception of life and its surroundings and is aware to these beings.
Intelligent systems take on many different forms. These can range from a simple being such as the Roomba vacuum, to a more advanced concept such as a fingerprint scanner in the latest line of Android phones.
Intelligent systems have 2 main ideas behind its concept. The first is the way it perceives its surroundings, and the other is how it communicates with the environment surrounding it.
Intelligent systems have progressed at an incredible rate since they were first founded in the 1960's. This technology has since skyrocked in popularity since then and has been used in all forms of day-to-day activities such as government sectors, industries and commercially viable alternatives. The reason for its exponential growth in the last 70 odd years, is the rapid improvements made to processers and therefore, the processing power of most computers today. This has allowed us to develop systems like never before and has drastically improved Intelligent Systems, one example being Artificial Intelligence.
In a more complex environment, we sometimes study how Intelligent Systems interact with human beings. In order to adapt to these circumstances, Intelligent Systems had to 'evolve' over time. This basically means that at one stage, these robotics for example were limited to simple tasks such as repeating the same output, as they had no true perception of the outside world, whereas today, if you were asked to define a robot, or a robotic being, you would describe them as a being that has a true perception of its surroundings, and can interact based on those circumstances and can work towards achieving a goal.

Dennis Kolomiyets is with the University of Limerick, Castletroy, Co. Limerick, V94 T9PX Ireland (e-mail: 20250762@studentmail.ul.ie).
Jai is with the University of Limerick, Castletroy, Co. Limerick, V94 T9PX Ireland (e-mail: 20045247@studentmail.ul.ie).
Konan Heney is with the University of Limerick, Castletroy, Co. Limerick, V94 T9PX Ireland (e-mail: 20242603@studentmail.ul.ie).
Rohan Arya is with the University of Limerick, Castletroy, Co. Limerick, V94 T9PX Ireland (e-mail: 20086377@studentmail.ul.ie).

II. WHY WE NEED INTELLIGENT SYSTEMS?

Intelligent Systems play a massive part in our everyday lives, whether we notice it, ignore it or just don't realize it. We are surrounded by Intelligent beings at every moment.
Some of these applications include the industrial sector, government departments, education, entertainment, military operations and transportation. For some of these sectors, Intelligent systems are needed, for others, it is more for a means of convenience. For example, for the industrial sector, Intelligent Systems is more of a necessity. For uses in education, it is more of a benefit but not necessary.

III. THE GAME OF CHESS

Chess is one of the oldest and most recognized games in the world. It is an exciting game of strategy and planning which is why it is seen as a symbol of intelligence. However, Chess is easy to learn and can be enjoyed by anyone at any level. Chess is played on an 8 by 8 board with 64 squares and 32 pieces, and the game is all about checkmating the King. Once the King is in checkmate, then the game is over. In Chess, both players have identical sets of pieces. Each player takes turns in making a move until the game is over. At the start, each player (one controlling the white pieces, the other controlling the black pieces) controls sixteen pieces: one king, one queen, two rooks, two bishops, two knights, and eight pawns. There are also several ways a game can end in a draw.
Chess is a game that involves both long-term positional strategies as well as sudden aggressive tactics.
Chess has gained tremendous popularity, and it is leading to a revolutionary trend due to more people getting involved with chess at the time of this global pandemic. So, there is no better time to analyse the role of artificial intelligence in improving the quality of chess.
Artificial Intelligence is a revolution with the numerous feats of accomplishments that it has been able to achieve. The use of AI in the real world and real-life scenarios is ample. We can see many examples in today's world like Automated Cars, virtual assistant software, weather, and climate tracking programs, and many more. They have a wide array of use cases to improve the quality of life in general. Another wonderful use of Artificial Intelligence is in the sport of chess.

IV. THE BIRTH OF CHESS AND AI

The ability of a machine to play chess well has taken on symbolic meaning since the first pre-computer devices more than a century ago. In 1890 a Spanish scientist, Leonardo Torres y Quevado, introduced an electromagnetic device that consisted of a wire, switch, and circuit that could checkmate a human opponent in a simple endgame, king and rook versus king. The machine, however, did not always play the best moves and sometimes took 50 moves to solve simple enough positions that would take an average human player less than 20 moves. But it could recognize illegal moves and always delivered eventual checkmate. He acknowledged that the apparatus had no practical purpose. However, his belief in the possibility of machines being programmed to follow certain rules had marked the start of the journey of Artificial Intelligence and Intelligence programs.

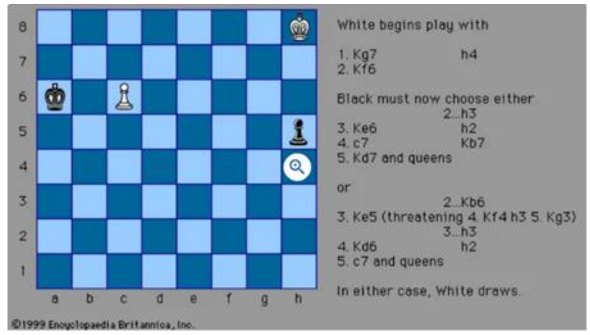
No significant progress in this area was made until the development of the electronic digital machine after World War II. About 1947 Alan Turing of the University of Manchester, England, developed the first simple program capable of analysing one ply (one side's move) ahead.

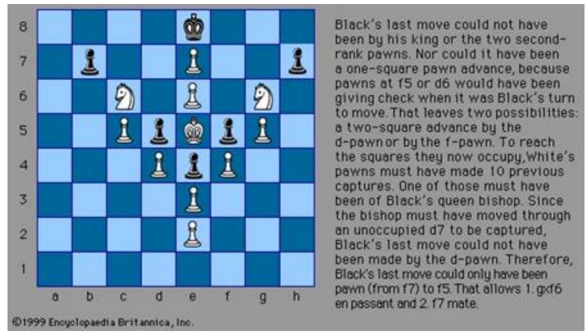
A year later in 1948 came a breakthrough, when the research scientist Claude Shannon of Bell Telephone Laboratories in Murray Hill, New Jersey, presented a research theory paper that had a major impact on the future of programming. Shannon, similar to Turing, theorized that progress in developing a chess engine, essentially a self-playing program could potentially have a wider application making machines intelligible and capable of translating into different languages or making strategic military decisions.

Shannon deduced that if the self-playing chess engine, at the time, were to run and play an entire game of chess itself, it would be required to make decisions that more often than not would be based on incomplete and unsure information as the confirmed checkmating positions might be about 50 moves ahead of the current position. This would require the chess engine to evaluate each possible move that wasn't merely legal, but also the better moves in that particular position which would lead to a definite checkmate. So, to make this work, Shannon published a paper that specified the criteria for evaluating each position a program would consider.

The evaluation function would be an integral part of the program as it would make it capable of determining the relative differences between thousands of different positions. Suppose it is White to play, so on average, it can be said that there are around 30 possible legal moves, to which Black will have 30 possible legal replies. This indicates that the machine would have to compute a combination of 30×30 i.e., 900 positions. This is called a two-ply search. A three-ply search would be where the program also evaluates the White's reply to Black's reply, where the machine would now compute a combination of $30 \times 30 \times 30$ or 27,000 positions of which it would have to consider the best moves in every position.

Turing's evaluation was based on the idea that the chess engine would determine which side had more pieces in the possible future positions. Contrary to Turing, Shannon suggested that it was important for the machine to evaluate other factors as well that would give positional advantages over a materialistic count of pieces. A side could (either White or Black) could have half the number of pieces as the other side and still be completely winning given a definite line of attack or checkmate pattern that could guarantee a win. There are various factors that impact the advantage within the game such as the number of pawns that determine an open or close position, control of central squares, mobility of the other pieces, whether all the pieces have been fully developed or not, specific cases of well-placed pieces, such as rooks with open files or pair of rooks on the 7th rank, knight outposts, or bishop pairs on long open diagonals. A human Grandmaster can easily evaluate these factors, but it was also important to make the machine capable of evaluating and taking these factors into consideration to suggest the best moves in every position.





V. DEEP THOUGHT AND DEEP BLUE

The '70s and '80s saw many technological advances. The computers being developed and programmed now could compute a far greater amount of data in a shorter period. The computer program Deep Thought was introduced in 1988 which could see more than 2 moves ahead giving it a much higher margin in rating points. At the time, it was rated around 2700 Elo points. Over time, the technological advances were exponential, as in the year 1996, the successor program of Deep Thought, Deep Blue was introduced that was an even stronger chess engine than Deep Thought, capable of seeing six moves ahead. To draw a picture, Garry Kasparov, who was the reigning world chess champion at the time normally only needed to look three to five moves ahead.

A key element in the improvement and advancement of these computers that led to significant progress was the availability of microprocessors in the late 1970s. This enabled the programmers to develop microcomputers that were nearly as strong as programs running on mainframes. Multiple different chess engines were being developed now that could beat some of the top-rated grandmasters in the world. HiTech was a chess engine developed at Carnegie Mellon University that had defeated a grandmaster in a chess match of blitz or rapid time format. The technology that the HiTech machine functioned on used 64 computer chips, one of each square on the board, that enabled it to compute 175,000 positions per second. Over the next few years, further improvements were made by the addition of custom-designed chips that made it capable to compute up to 700,000 positions per second. This improved version was named Deep Thought which was later sponsored by IBM in an effort to beat the top Chess grandmasters around the world.

In 1991, Deep Thought had been renamed Deep Blue, and multiple new changes had been made to the program that made the chess engine much stronger. The main improvements that were made to the machine were that now it employed 32 microprocessors where each has 6 programmable chips designed specifically for chess. This new addition to the program made it consider as many as 50 billion positions in three minutes, which is approximately 1000 times faster than Deep Thought's. The first official match that Deep Blue played was in 1996 against the reigning world chess champion at the time, Garry Kasparov, arguably the greatest Chess player of all time. The match consisted of 6 games. Deep Blue won the first game of the tournament which was an aggressive tactical battle between the powerful machine and Kasparov. However, in the games that followed, Kasparov implemented long-drawn-out strategic and positional ideas that gave him an edge over Deep Blue and he managed to draw the next 2 games, and eventually won the last 3 games, emerging victorious in the match with a score of 4-2.

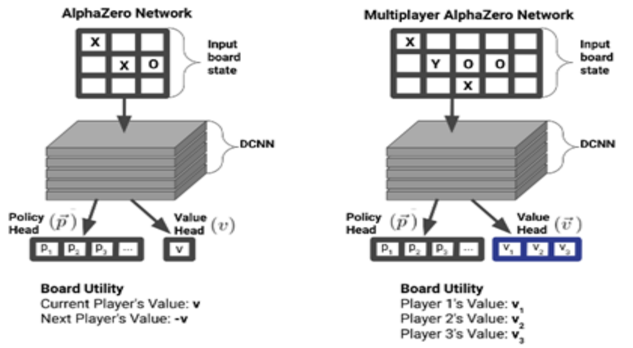
It was the rematch only a year later, in 1997, that shook the world, which is why it is often referred to as the year of a breakthrough and progress in the field of computer science. The newly upgraded version of Deep Blue now had the processing capability of computing 200 million positions per second, i.e., it was now performing at double the rate than it was earlier. The Deep Blue team had also added changes to its functioning algorithm by suggestions from grandmasters. The match itself, however, had some controversies about it regarding the choice of moves that the computer played. In-game 3 of their match, Kasparov played the Caro-Kann Defence against the move 1.e4 by Deep Blue. After the move 4. Nxe4, Kasparov replied with move Nd7 entering the Karpov variation of the Caro-Kann defence. It was Kasparov's shocking choice on the 7th move after playing 7…h6, thinking Deep Blue would be unaware of the obvious human move of sacrificing the Knight to gain a dynamic position, and would instead move the Knight backward, as, at the time, all the computer programs were very materialistic when it came to evaluating positions with the pieces on the board. To Kasparov's shock, Deep Blue sacrificed its Knight after playing the move 8. Nxe6 and suddenly this made Kasparov's position extremely blocked and undeveloped. Kasparov expressed his harsh opinions after the match saying that the program was fully aided by a grandmaster throughout the game and later it was found that the programming team had preinstalled the theoretical database within the program of Deep Blue which made it familiar with the tactics of the Karpov variation of the Caro-Kann Defence. Despite the controversies, it was mostly regarded in the favour of Deep Blue, and it was officially the first time in history, that a chess program had proved to be stronger than humans.

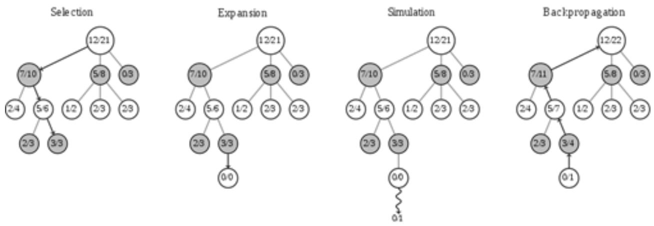
VI. MODERN-DAY AI AND CHESS

Nowadays, modern chess engines are so well-developed that they will not drop a single game to human players. Even the current reigning, defending world champion, Magnus Carlsen would fail to beat the modern best chess engine even once in a span of 100 games. Magnus Carlsen has a FIDE rating of over 2800 across all formats. To draw a picture, Stockfish 9, at the time Magnus was crowned world champion, had a rating of over 3400.
Stockfish 8 was considered the strongest chess engine, which for most top players is their go-to preparation tool, and which won the 2016 TCEC Championship and the 2017 Chess.com Computer Chess Championship. Stockfish 8 at the time used the technology of Brute force to compute and calculate 70 million chess positions per second. Of these millions of moves, Stockfish picked what it saw as the very best one, with "best" defined by a complex, hand-tuned algorithm co-designed by computer scientists and chess grandmasters. That algorithm valued a delicate balance of factors like pawn positions and the safety of its king.

VII. GOOGLE DEEP MIND'S ALPHA ZERO: BIRTH OF NEURAL NETWORKS IN BOARDGAMES

But in late 2017 everything changed. The DeepMind team at Google introduced the AI Alpha Zero. They gave it four hours and the basic rules of chess. After only four hours of training, DeepMind estimated Alpha Zero was playing chess at a higher Elo rating than Stockfish 8; after nine hours of training, the algorithm defeated Stockfish 8 in a time-controlled 100-game tournament (28 wins, 0 losses, and 72 draws). Alpha Zero in some ways was very much weaker than Stockfish—powering through just 1/100th as many moves per second as its opponent. But Alpha Zero is an entirely different machine. Instead of deducing the "best" moves with an algorithm designed by outside experts, it learns strategy by itself through an artificial intelligence technique called machine learning. Its programmers merely tuned it with the basic rules of chess and allowed it to play several million games against itself. As it learned, Alpha Zero gradually pieced together its own strategy. It combined a neural network and Monte Carlo Tree Search in an elegant policy iteration framework to achieve stable learning.





These modern results with the help of chess engines and neural networks and deep learning-based chess networks taking over the chess world by storm is a huge sign of the potential for greater and enormous possibilities.

VIII. CHECKERS / DRAUGHTS

Checkers, also known as Draughts, is a 2-player strategy board game. The aim is to be the first to take all your opponent's pieces by moving diagonally on an 8 x 8 checkered board. The game is known to be more than 5,000 years old and has origins in Iraq as far back as 3,000 BC. There are 2 versions of Checkers that are known. There is the standard American Checkers, which is played on a 8 x 8 board and consists of the pieces moving forward only. Another version is Russian Checkers, which is played on a 12 x 12 board, and International Checkers, which features a 10 x 10 board. These games usually last for anywhere between 5 minutes to 30 minutes. For convenience, I will only speak of the most well-known version of Checkers, which is American Checkers.

A few of the basic rules for Checkers, for example movement. You may only move your piece in a diagonal way and can only take your opponent's piece, if you 'jump' over them.

Crowning or Kinging involves getting your piece all the way to the top of the board, or in other words, to the other side or your opponent's side. This Kinged piece can move in any diagonal direction by one space, unless of course, you are taking a piece from your opponent.

Like Chess, Checkers is played on a N x N board. This means that is can be played on a board of any size without there being any ambiguity of any kind between players.

The man behind the Checkers engine is Jonathan Schaffeur, and his opponent was Marion Tinsley, a maths professor, minister and finally, the master of Checkers.

Tinsley was the best player in the game for the better part of 40 years. He had only lost a handful of games to people in his career, but not a competitive game. His opponent, Chinook, developed by Schaffeur was unbeaten in 125 games before facing Tinsley. After 6 games, all being draws, Tinsley was sent to hospital where he discovered a lump on his pancreas. He was forced to retire, and Chinook became the first Artificially Intelligent program to win at the human world championship. A program as a result of thousands of hours of tuning and research, to win at the world championship, due to its opponent discovering pancreatic cancer.

At the highest levels, checkers is a game of mental attrition. Most games are draws. In serious matches, players don't begin with the standard initial starting position. Instead, a three-move opening is drawn from a stack of approved beginnings, which give some tiny advantage to one or the other player. They play that out, then switch colors. The primary way to lose is to make a mistake that your opponent can jump on.

The first checkers program was developed in the 1950's by an IBM employee named Arthur Samuel. He originally intended for his program to run on an IBM 704. He spent a good 15 years on this idea and at the end had published some well-known papers, some of which include the concept of machine learning.

Machine learning is one of the most modern forms of Artificial Intelligence. It is the basis for the likes of Algorithms and Perception. Samuel's program was never very successful and was famously known for being trounced by the Endicott Johnson club in 1958. But it was important in the grand scheme of Machine Learning for Checkers. Samuel also quoted in one of his papers "The human brain, sometimes lost sight of in an age of satellites, frozen foods, and electronic data processing machines, returned to former glories early today,"

Chinook needed to search through possible moves beginning with the start of a match. Like many similar systems, Chinook would look ahead many possible moves and then try to score each permutation's desirability. At first, Chinook could only look ahead 14 to 15 moves out, but as computers and the software improved, it could look further and further. "As with chess, deeper was always better," Schaeffer said.

IX. GO

Go, or Weiqi is an abstract strategy board game for 2 players in which the aim is to surround more territory than the other opponent. The game was invented in China nearly 2500 years ago and is believed to be the oldest board game continuously played to the present day. The game of Go although at a first glance seems simple enough with black and white stones of uniform type, but as we slowly take a deeper dive into the strategies and tactics involved within the game, we find out that the game is far more complex. The two types of stones, that are the black and white stones are assigned to both the players playing the game, one with the white and on with the black. The players take turns alternatively in placing the stones on the vacant intersections of the board. After the stones have been placed by the players, these stones cannot be moved. However, these stones can be removed from the board in the case where the particular stone or the group of stones is surrounded by the opposing stones on all the orthogonally adjacent points, which leads to the capture of a stone or a group of stones. The game goes on like this until the players reach a point where they are unwilling to make a move. This is when it is determined who emerges victorious by counting the players' surrounded territory along with captured stones and the Komi points. A player can also emerge victorious if the other player chooses to resign.

X. ALPHA-GO

The initial stage consisted of developing and designing the program on the concept of supervised learning. This can be explained as – a database was created in which more than 160,000 real-life professional games have been added and relative positions in each game were paired with the moves that the neural network within the machine learned over time. These connections were established among the networks that are referred to as the deep machine learning technique which enables the program to choose between the better of the given options of the moves.

In the next stage a concept of reinforcement learning was implemented which is better known as behaviourism within a program. It holds that organisms, ranging from worms, flies, and sea slugs to rats and humans, learn by associating a given action with specific stimulus that preceded it. The machine used a similar concept to develop a stimuli of its own just like the other living organisms which it used to learn about GO.

In the final stage of training, the value network that estimates how likely a given board position is likely to lead to a win, is trained using 30 million self-generated positions that the policy network chose. It is this feature of self-play, impossible for humans to replicate (for it would require the player's mind to split itself into two), that enables the algorithm to relentlessly improve.

A peculiarity of AlphaGo is that it will pick a strategy that maximizes the probability of winning regardless of by how much. For example, AlphaGo would prefer to win with 90 percent probability by two stones than an 85 percent probability by 50 stones. Few people would give up a slightly riskier chance to crush their opponent in favour of eking out a narrow but surer victory.
The result is a program that performed better than any competitor and beat the go master Fan Hui. Hui, however, is not among the top 300 world players—and among the upper echelons of players, differences in their abilities are so pronounced that even a lifetime of training would not enable Hui to beat somebody like Lee Sedol. Thus, based on the five publicly available games between AlphaGo and Hui, Sedol confidently predicted that he would dominate AlphaGo, winning five games to nothing or, perhaps on a bad day, four games to one. What he did not reckon is that the program he was facing in Seoul was a vastly improved version of the one Hui had encountered six months earlier, optimized by relentless self-play.

XI. GO, MATHS AND AI

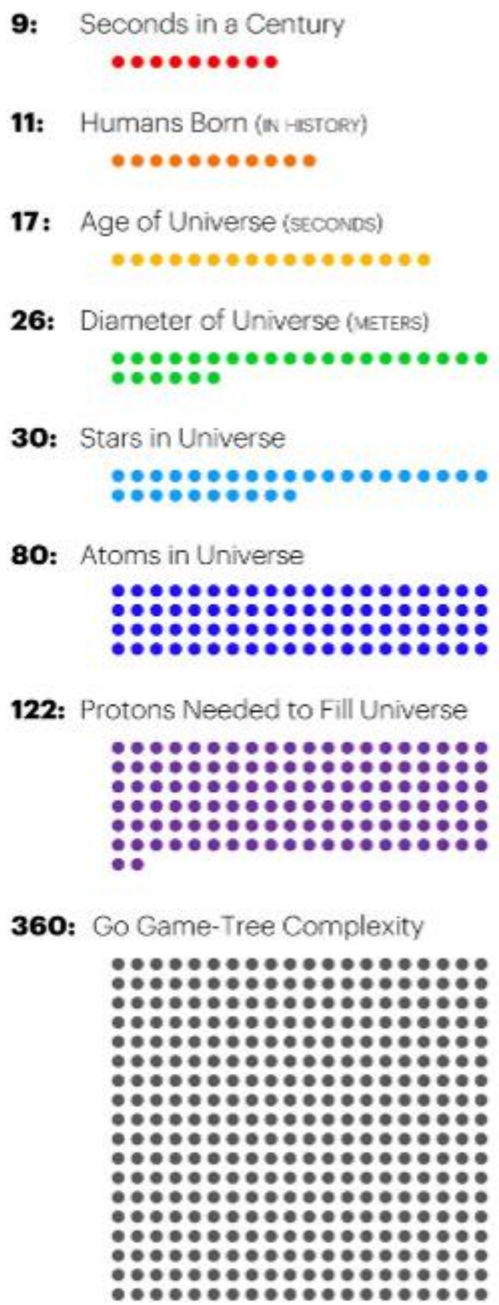
In the field of Artificial Intelligence, computer-generated gameplay suggests how brains operate. Spearheaded by Demis Hassabis, David Silver, and their colleagues at the artificial intelligence (AI) company Google DeepMind, the team actively worked on a computational strategical program called AlphaGo, that used a combination of brute force logic and tree search, and in the later versions, self-learning, that made AlphaGo the strongest engine at the game of Go that defeated all the best Go players in the world Including the World Champion.
To understand the power of this approach in playing board games, consider the search spaces involved. The "game tree complexity" of tic-tac-toe—i.e., an estimate of the number of possible positions that must be evaluated to determine the worth of an initial position—is about 2 × 104.
There are several different ways to measure the complexity of more difficult board games, leading to different estimates. But no matter the method used, as the complexity of board games grows, the relevant search spaces become unimaginably large. The game tree complexity is estimated to be about 10^20 for checkers, about 10^120 for chess, and an astonishing 10^320 for Go (6). For comparison, the estimated number of atoms in the universe is on the order of 10^87. Unlike tic-tac-toe, fully searching spaces of this magnitude is not feasible. In the strategy used by Google DeepMind, success follows not from a logical exploration of possible moves but from the instantiation of trial-and-error play in the network's connectivity that beats a competing machine playing in the same empirical way.
GO and Mathematics
There is a complex system of Mathematics that is involved in the program of GO. GO deals with massively huge numbers, so for us to get a better understanding of the system, we use log (base 10).
As we are aware by the laws of logarithmic functions, that if 10^x = y then log y = x.

What is State space complexity?
Before delving deeper in the mathematical functioning of the system here, it is crucial for us to first understand what is meant by State space complexity. State space complexity is the number of legal positions that can be reached based on the starting position of the game. On the face of it, it sounds simple enough. Whenever you make a move, move a piece, or capture a piece, a new board state is created as each move, we play regardless of its nature, results in a unique position. The aggregation of them complicates the state space of the game. However, due to the two parts of this definition, it is very difficult to calculate these numbers accurately. First, we have to deal with the legal definition. Determining the number of ways in which game pieces can be configured inside and outside the board is not difficult, but determining an acceptable subset is much more difficult.
GO has an upper bound state-space complexity of 172. This can be determined by using the following system of calculations.
Spaces on the board = 19 x 19 = 361
And each space can have a possibility out of 3 unoccupied or occupied by either colored stone or no stone at all.
This would mean that the state space complexity is 3^361 = 17408965065903192790718823807056436794660272495026354119482811870680105167618464984116279288988714938612096988816320780613754987181355093129514803369660572893075468180597603
But since the number is quite huge, we will use log (base 10) to simplify the number log (17408965065903192790718823807056436794660272495026354119482811870680105167618464984116279288988714938612096988816320780613754987181355093129514803369660572893075468180597603) = 172.24077295379814
Which is approximately 172.
On the other hand, Game-tree complexity represents the number of distinct plays, or paths, for a given game. Each path consists of multiple different game states, so these numbers are going to be even larger than state-space complexity.
It is difficult to estimate these values, but a simple equation can provide a reasonable lower bound. This equation requires the branch factor, basically the average number of legitimate movements available to the player each turn and the average game length. In these calculations, the length of the game is measured as a lie. A lie is a single turn performed by one player, and each player usually resolves one lie each turn, which is quite different from the more casual term turn. Therefore, the total number of layers can be effectively equalized to the number of rounds multiplied by the number of players.
GO has an upper bound game-tree complexity of 360 we calculate this using the following facts
GTC >= b^p
The branching factor = 250
Average game length = 150 plies
Hence the GTC for GO is 4909093465297726553095771954986275642975215512499449565111549117187105254721715856460097884037331952277183571565131878513167918610424718902807514824108963452253105464459861928538941810984397307038307189941406250000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Again, this is a huge number so we will use log (base 10) to simplify it log (4909093465297726553095771954986275642975215512499449565111549117187105254721715856460097884037331952277183571565131878513167918610424718902807514824108963452253105464459861928538941810984397307038307189941406250000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000) = 359.6910013008056

Which is approximately 360

To give perspective to this if we look at some other data, we can get a better understanding of this.



The technology utilized in the software of the AlphaGo's algorithm does not contain any novel insights or breakthroughs. The software is simply a combination of neural network algorithms and machine-learning techniques running on hardware that consists of 48 Central processing units (CPUs) augmented by eight graphical processing units (GPUs) developed to render 3-D graphics for the gaming effects as well as running complex and powerful mathematical operations.

The foundation of Computations is neural networks, that are mechanical and digital representation of the neuronal circuits operating in biological brains. Like that, the position of the stones on the 19 by 19 GO board represents something called convolutional network. For any given position in a situation in a game, the two neural networks involved operate sequentially to optimize the performance. One of these two networks is called a value network that reduces the effective depth of the search by estimating how likely a given board position will lead to a win without chasing down every node of the search tree, on the other hand, the "policy network" reduces the breadth of the game, limiting the number of moves for a particular board position the network considers by learning to choose the best moves for that position. The policy network generates possible moves that the value network then judges on their likelihood to vanquish the opponent.

XII. DIPLOMACY

From what was seen from chess and go innovations it is clear, that great steps were taken in the advancement of Artificial intelligence around 2 player games. However, very recently a study undertaken by Deepmind delves into the intricacies of developing competent AI for more, complex games. This more complex game is called Diplomacy.

A. An Explanation on how the game works

Diplomacy can be played by up to seven people on a sprawling map of Europe, where the great powers at the time of the turn of the 20th century battle and scheme to decide on a winner. Battles are played with "army pieces" and are very simplified. The games turns are divided into the seasons of autumn and spring for each year. Turns for each player are simultaneous. At the start of a season player enter the diplomatic phase where players discuss strategy with each other. Here you as a player can negotiate deals, forge alliances, craft schemes and plan attacks among whatever you can think of. Afterwards players write down their orders for that turn when they enter the order phase. This is where they can follow through on the plans or promises they made during the diplomatic phase or betray said promises and backstab would-be allies. Orders are written on paper. They are dated e.g. (Spring 1912) and always start by determining whether an order is for an army or a fleet with a corresponding letter (a or f). Then the order is given a type. These types include Hold, Move, Support and Convoy.

B. Move

A move order is given is given with a dash (-) between two provinces for example (Berlin-Prussia). Armies can move onto land or coast tiles, however, cannot move into water. Fleets on the other hand can move freely on water and coast but cannot traverse land tiles. If two opposing units try to attack-move onto another tile the result will be different based on the scenario.

- If the armies or fleets are equal in strength and are both moving onto the same tile, then they remain in their original positions.
- If an army or fleet are of equal strength, however one of the armies was already on the tile beforehand, the attacker moves back to its original tile and the defender holds

- One unit that is not moving can stop a unit or series of units from moving.
- Units cannot trade places. Unless a convoy is used.

C. *Support*

All units individually are the same strength, which would make the capture of a province impossible if these settlements have any defenders. In response, units can support each other. This is where multiple units stage an attack on one settlement. If the unit that was attacked had no orders to move elsewhere, it is defeated and dislodged from the province. The dislodged unit must retreat or be disbanded. Support need not only be offensive support but can be used on defense too. Through supporting the unit's strength grows. Players can support each other, and support cannot be denied. Finally, support can also be cut from the unit if the supporting unit gets attacked.

D. *Convoy*

A fleet in a water province can convoy an army from any adjacent coastal province to any other coastal province adjacent to that water province. This acts as a parallel to ships that would carry troops overseas in reality. Though in game the fleet piece acts like a sort of a bridge over a body of water. Fleets can only convoy in a sea tile no coastal tiles and can only convoy one army at a time. If you want to transport an army across multiple tiles of water, you need to control each tile of the path with a fleet and then your army can land on the other coast within a turn. Lastly if a fleet is dislodged during movement the army attempting to cross remains stationary.

E. *Hold*

Keep the unit where it is.
When the orders are reveals, they will follow their own order. will result in successful moves, failed moves, standoffs, retreats and disbandment. Afterwards affected units disband or retreat and gain new ones. Then the next season starts, and the entire process repeats again. These rounds go on until one player holds enough important provinces to win.
We see why the team at Deepmind chose to study this game because according to them the game was specifically designed to emphasize tensions between competition and cooperation, never knowing whether your so-called ally would metaphorically stab you in the back. This helps in the learning of mixed motive situations. Due to how supporting works players need to coordinate with others to overcome obstacles and can gain from supporting each other in capturing provinces.
A game like Diplomacy proves a real challenge for RL (Reinforced Learning) agents to grasp. Unlike Chess, Checkers or Go, Diplomacy is typically played with more than 2 players. Which means the agent cannot rely on the basic properties of two player games. They state that Philip Paquett developed an agent called "DipNet", where they recorded a dataset 150,000 human played diplomacy games and using a graph neural network (GNN) the agent would copy moves taken in the dataset. It was this very agent that bested previous "state of the art AI" and not only bested but in a decisive victory. Unfortunately, this wasn't an RL method, and to date RL has not been properly implemented.
Deepminds' contribution is that they teach an agent to play "No Press Diplomacy" which is played in the exact same way as regular Diplomacy except for that there are no direct communication channels between the players. They use a PI (Policy Iteration) approach with SBR (Sampled Best Response), a simple yet scalable improvement operator which allows the handling of Diplomacy's large combinatorial quantity of actions and simultaneous moves. They introduced versions of PI that approximate iterated best response and fictitious play (FP). They show that their agent out-do previous state of the art agents both against head-to-head and reference populations.

F. *RIM(Reinforced Learning Methods)*

PI methods use an improvement operator in the form of BR Calculations. For all players we define a policy π^b and the best response for player i would be the policy $\pi_i{}^*$ that should maximize the return against opponent policies $\pi^b{}_{-i}$. Though it should be noted that best response policies may prove to perform poorly against policies they weren't designed to beat. Though they cannot deny its usefulness. Diplomacy is too big a game for perfect BR operator calculations, so the team at Deepmind proposed an alternative. SBR (Sampled Best Response).
SBR makes 3 approximations:

- They make a single turn improvement over each phase of the game, instead of a full calculation over numerous turns.
- They think about only taking a small number of moves or actions, sampled from a "candidate policy".
- They use Monte-Carlo estimates over the opponents' actions for candidate evaluation. (Monte-Carlo estimates meaning using randomness to solve a problem.)

Consider the calculation of the value of some action a_i for player i against an opponent policy of $\pi_-{}^b{}_i$. Let T (s, a) be the transitional function of the game and V^{π} (s) be the state value function for the policy π. The one turn value to player i of action a_i in states is given by the formula:

$Q^{\pi b}{}_i(a_i|s) = E_{a_{-i} \sim \pi_{-i}{}^b} V_i^{\pi b}(T(s, (a_i, a_{-i})))$

G. *Best Response Policy Iteration*

To implement they used BRPI (Best Response Policy Iteration), which is a "family of PI approaches tailored to using (approximate) BRs, such as SBR, in a many- agent game," as an algorithm. They show off two algorithms for this as seen below:

Algorithm 1 Sampled Best Response

Require: Policies π^b, π^c, value function v

function SBR(s:state, i:player)
  for $j \leftarrow 1$ to B **do**
    $b_j \sim \pi^b_{-i}(s)$ ▷ Sample Base Profile
  for $j \leftarrow 1$ to C **do**
    $c_j \sim \pi^c_i(s)$ ▷ Candidate Action
    $\hat{Q}(c_j) \leftarrow \frac{1}{B}\sum_{k=1}^{B} v(T(s,(c_j,b_k)))$
  return $\arg\max_{c\in\{c_j\}_{j=1}^{C}} \hat{Q}(c)$

Algorithm 2 Best Response Policy Iteration

Require: Best Response Operator BR

function BRPI($\pi_0(\theta), v_0(\theta)$)
  for $t \leftarrow 1$ to N **do**
    $\pi^{\text{imp}} \leftarrow \text{BR}(\{\pi_j\}_{j=0}^{t-1}, \{v_j\}_{j=0}^{t-1})$
    $D \leftarrow$ Sample-Trajectories(π^{imp})
    $\pi_i(\theta) \leftarrow$ Learn-Policy(D)
    $v_i(\theta) \leftarrow$ Learn-Value(D)
  return π_N, v_N

They Trained a variety of agents; however, FPPI-2 was particularly of note. They train the policy networks to predict only the latest BR, and explicitly average historical checkpoints to provide the best strategy so far.

H. *Evaluation Methods*

Lastly, I would like to cover the several evaluation methods that were used to evaluate the results for agents of different algorithms for head-to-head and against fixed populations of reference agents. They checkpointed times in a game and ran the AI through as a form of "Meta-game" to constantly improve said AI and tested for exploitability.
Head-to-head comparison:
They play 1v6 matches against different agents following different implementations of BRPI algorithms and record the results. This also allows the recording of unusual events between two agents. Another point of interest is it allows for an environment where one agent can dominate the board and "invade" the other agents separating it from the pack.
Win-rate Against a Population:
The opposite of Head-to-head, where once an AI is competent enough it is put against 6 independently drawn players from a reference population and its win-rate is subsequently recorded. This

is to mirror how people actually play the game, where each agent is acting in its own self-interest and wants to maximize its own score.

In Results they show how Iterated Best Response does surprisingly well, though it is FPPI-2 that was the strongest method.

References

[1] https://www.elucidate.ai/post/how-ai-revolutionised-the-ancient-game-of-chess
[2] https://www.britannica.com/topic/chess/Chess-and-artificial-intelligence
[3] https://towardsdatascience.com/ai-in-chess-the-evolution-of-artificial-intelligence-in-chess-engines-a3a9e230ed50
[4] Learning to Play No-Press Diplomacy with Best Response Policy Iteration (2022) T. Anthony , T. Eccles, Deepmind
[5] Philip Paquette, Yuchen Lu, Seton Steven Bocco, Max Smith, O-G Satya, Jonathan K Kummerfeld, Joelle Pineau, Satinder Singh, and Aaron C Courville. No-press Diplomacy: Modeling multi-agent gameplay. In Advances in Neural Information Processing Systems
[6] No Press Diplomacy: Modeling Multi-Agent Gameplay, P. Paquette Y. Lu, S Bocco
[7] https://www.theatlantic.com/technology/archive/2017/07/marion-tinsley-checkers/534111/
[8] https://www.unr.edu/cse/undergraduates/prospective-students/what-are-intelligent-systems
[9] https://www.gamesver.com/history-of-checkers-easily-explained-with-pictures/
[10] https://libapp.shadygrove.umd.edu/omeka/exhibits/show/international-games-day/history-of-checkers
[11] http://www.quadibloc.com/other/bo1211.htm
[12] https://www.scientificamerican.com/article/how-the-computer-beat-the-go-master/
[13] https://www.pnas.org/doi/10.1073/pnas.1909565116
[14] https://www.pipmodern.com/post/complexity-state-space-game-tree