Artificial intelligence for games

Programming a Computer to play Chess in the 1950s

term paper by

Jacqueline Wagner

Submitted to

PD Dr. Ullrich Köthe

on May 8, 2019,

spring semester 2019

Abstract

In this term paper we will take a closer look at the beginning stages of artificial intelligence, especially first attempts made in programming a computer to play chess. We will dive deeper into different chess-algorithms formalized by (1) Claude E. Shannon and (2) Alan Turing and will compare brute force and selective search strategies.

For further questions contact:
Jacqueline Wagner
jacqueline.wagner@stud.uni-heidelberg.de
Matriculation number 3390137

Contents

1 Introduction **1**

2 The brute force search strategy **3**
2.1 Playing a perfect game of chess 3
2.2 Playing a skillful game of chess 3
2.3 Shannon's type A strategy 4
2.3.1 A simple example 4
2.3.2 Necessary hardware 5
2.3.3 Constructing an executable program 5
2.4 Ideas for further improvements 7

3 A more advanced selective search strategy **8**
3.1 Shannon's Type B strategy 8
3.2 Turing's strategy 9
3.3 Problems to consider 9

4 Conclusion **11**

1 Introduction

> Could one make a machine which would have feelings like you and I do?
>
> Alan Turing in [8]

The ever evolving goal of creating artificial intelligence has fascinated computer scientists for decades. Whether it's a machine capable of human thought or simply an algorithm capable of solving the game of chess, theories, possible solutions and predictions continue to attract wide spread attention. While it is no longer questionable that computers have much higher capabilities when it comes to processing logic, the question as to whether or not computers will one day be able to feel like humans remains unanswered. Although this question unarguably is of high interest, most scientist agree that we must first lay the theoretical foundation by solving less complicated problems of a similar nature. This realization combined with the well-defined nature of chess soon lead many researchers to devote their time to *programming a computer to solve chess*.

Claude E. Shannon The first paper to ever be published on this topic was written by American mathematician and Nobel prize winner Claude Shannon. His 1949 paper "Programming a Computer to play Chess" [6] laid the theoretical groundwork for generations of researchers to come. Like many scientists alike, Shannon enjoyed playing a game of chess in his free time. However, unlike most of his peers, he was skilled enough to last for 42 moves against one of the most highly regarded chess players in the world. His skill and interest undoubtedly fueled his passion for constructing a superhuman algorithm.

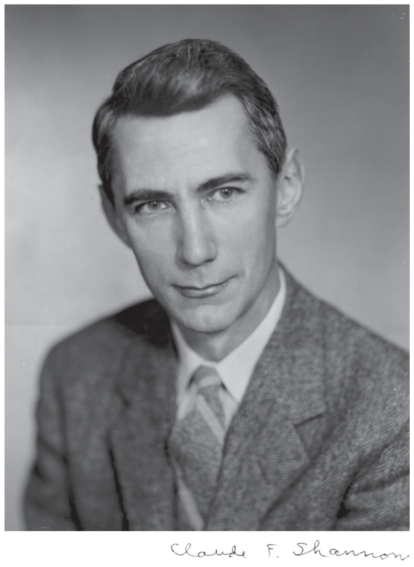

Figure 1: Claude E. Shannon from [4]

Alan Mathison Turing Praised by many historians as the *father of computer science*, Turing's many groundbreaking discoveries still influence and inspire the way computer scientists work today. In his short life Turing not only single-handedly changed the outcome of World War II, he also made many technological advances in the field of Computer Science. Although the term *AI* was not created until 1956, two years after Turing's death, he came up with fundamental theories in AI which continued to be relevant over the decades. Similar to Shannon, he saw chess as a way to test the true ability of emerging generations of AI.

Turing began working on his chess-solving algorithm in 1948 when computers were not yet capable of solving complex problems. After finishing his algorithm in 1950 he resorted to executing the calculations using pen and paper in order to test the ability of his work. He played against his friend Champernowne and Champernowne's wife – a rookie in chess – taking anywhere up to 30 minutes per move. While his algorithm unfortunately couldn't beat his advanced chess player friend, it was able to stand the test against his friends wife. Turing published his ideas for a chess-solving algorithm in his 1953 Paper "Digital Computers applied to Games" [8].

Alan Turing died in 1954 after being prosecuted and subsequently chemically castrated for his relationship with another man. Following his death many of his early advances fell into darkness and weren't rediscovered for decades to come. In June 2012, his algorithm *Turbochamp* finally got the chance to play the world champion chess player at the Alan Turing Conference in Manchester. While it didn't stand a chance against Garry Kasparov, he later stated "I suppose you might call it primitive, but I would compare it to an early car – you might laugh at them but it is still an incredible achievement. He wrote algorithms without having a computer -– many young scientists would never believe that was possible. It was an outstanding accomplishment." [3]

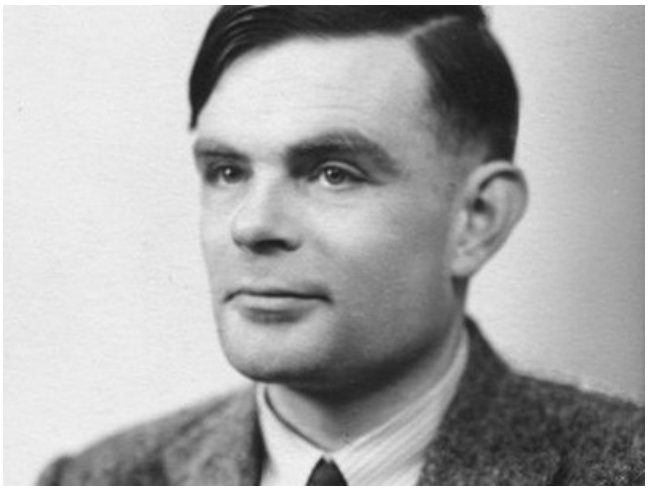

Figure 2: Alan Turing from [5]

2 The brute force search strategy

2.1 Playing a perfect game of chess

For certain games an evaluation function $f(P)$ can be constructed, which can determine for every position P whether the position is lost, won or drawn. Given such a function $f(P)$, it is very easy to design a machine capable of playing a perfect game. Since all chess games end after a finite number of moves such a machine would simply have to consider all possible moves in the game. However, both Shannon and Turing quickly realized that such a machine would likely never exist. They assumed that even with very high computing speeds the number of possible ways of playing chess is undoubtedly too large for any machine to evaluate. In fact, the number of possible game variations in chess is estimated to be around 10^{120}. Therefore, even if a memory cell made up of about $10,000,000$ atoms existed, this strategy would still require a computer the size of earth.

As a result of their findings, both Shannon and Turing decided to give up the idea of playing a perfect game of chess and instead focused on playing a skillful game of chess at a level comparable to that of a good human player.

2.2 Playing a skillful game of chess

Since both Shannon and Turing assumed there would never be a perfect function $f(P)$ they decided to concentrate on constructing a crude evaluation function $f(P)$ based on values such as the numbers of pieces on the board, possible legal moves, rotation and mobility. In addition they wanted to enhance their function by using chess principles with statistic validity, for instance deducting points for an exposed king since this is generally considered a weakness in the world of chess. Although this type of evaluation is not perfect, the strategy used is similar to that of human chess masters. Since skillful players tend to be very good at assessing chess positions, the overall goal quickly shifted to teaching a machine to evaluate like a human chess master.

A regular chess game between two human players is usually played in a manner where the players only investigate a certain number of variations move by move until a calm position on the board has been reached. In this state of stability players tend to invest more time in assessing moves and increase the depth of their evaluation. To further imitate the decision process of a human chess player, Shannon as well as Turing decided to restrict the application of the evaluation function to stable positions.

2.3 Shannon's type A strategy

In his first strategy Shannon opted to apply his evaluation function $f(P)$ after every single move. Despite knowing that this likely would not result in a satisfactory solution he saw this strategy as a stepping stone for a more complex solution.

Shannon decided that it would make sense for the algorithm to calculate all variations out to two moves for each side in order to obtain the resulting positions on the board. His plan was to apply his evaluation function $f(P)$ to each of these positions. In an attempt to evaluate how good or bad a certain move in a given position would be, he concluded that the algorithm would have to move backwards through the decision tree, minimizing the evaluation function's output for moves made on the opponent's side and maximizing on the machine's side. Once the algorithm obtained a value for each possible move in a given position it would choose the one yielding the highest evaluation.

2.3.1 A simple example

To keep it simple the below example is only calculated out to one move deep on each side.

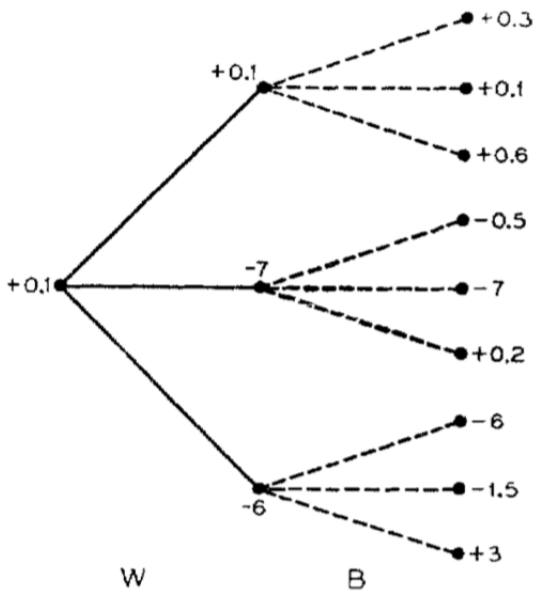

Figure 3: Strategy A demonstrated in a decision tree from [6]

The example starts off in a position P in which the white player has the choice between three different moves to make. Shannon's algorithm determines the resulting three positions and lists all possible ensuing moves for the black player. In a last step the final nine positions are listed, resulting in a completed decision tree. At this point the evaluation

function $f(P)$ is calculated for all nine positions. The three lowest values of the three sub trees are then back-propagated, e.g. +0.1 for the first sub tree, and assigned to their respective nodes. Last but not least the move yielding the highest value out of all three minimum values is chosen, e.g. $+0,1$ meaning that the first move is the one to make.

2.3.2 Necessary hardware

Unlike Turing, Shannon briefly touched on the topic of hardware in his paper.

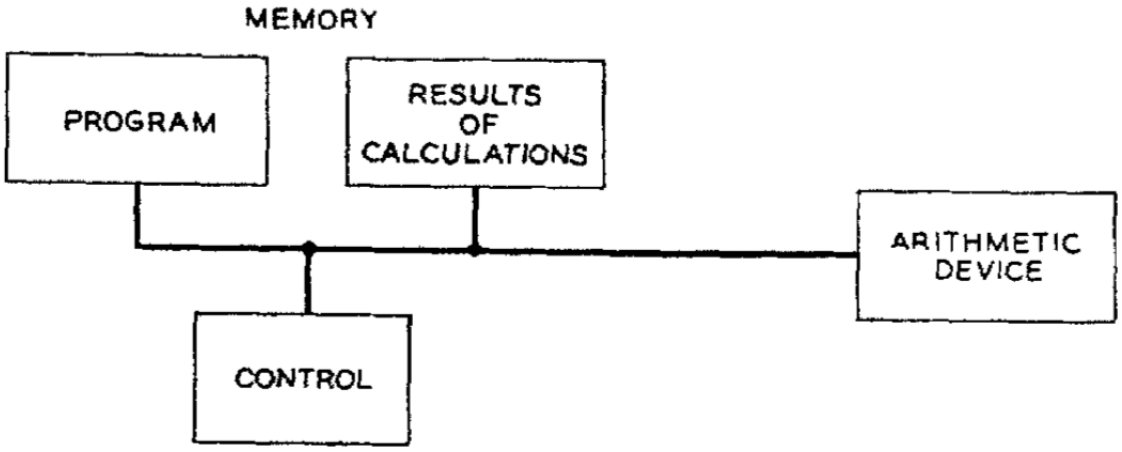

Figure 4: Hardware deemed necessary to execute strategy A from [6]

He concluded that an internal memory to store numbers in numbered boxed would be necessary. In addition, his algorithm required an organ capable of performing arithmetics such as addition or multiplication. He wanted the whole system to operate "according to a program consisting of elementary orders which operate on the numbered boxed in the internal memory" [6].

In order to execute the algorithm the program also required some information relating to the current state of the game. Shannon decided that he would pass the position on the board by assigning each square a number between -6 and 6.The values -6 to -1 and 1 to 6 would correspond to the six different possibilities for black and white figures on the board while 0 would refer to an empty square. Furthermore, a value λ would specify whether it is the black player's ($\lambda = -1$) or the white player's turn ($\lambda = 1$).

2.3.3 Constructing an executable program

In his paper Shannon described the ten building blocks necessary to implement his strategy A algorithm:

- $T0$: Make a move (a, b, c) in position P to obtain the resulting position. In this instance a and b refer to the positions occupied by a piece before and after a move while c specifies a piece in case of promotion.
- $T1$: Make a list of all possible moves for pawn pieces at square (x, y) in position P.
- $T2 - T6$: Perform the above evaluation on all other types of pieces.
- $T7$: Make a list of all possible moves for all pieces in a given position P.
- $T8$: Calculate the evaluation function $f(P)$ for position P.
- $T9$: Perform maximizing and minimizing to determine the overall best move.

Shannon first attempted to only use parts $T0 - T7$ to play a game of chess using an algorithm which would choose any legal move at random. As to be expected, this algorithm performed very poorly and lost nearly every game in roughly four to five moves. This confirmed Shannon's theory that a good evaluation function was crucial to the success of the algorithm. He described his algorithm as follows:

1. List all possible legal moves in the present position using $T7$.
2. Take the first move off the list and apply it using $T0$.
3. List all possible legal moves for the opponent in the, by means of (2) obtained, position.
4. Apply the first and the second possible move in the above list using $T0$.
5. Evaluate the in (4) resulting positions in $T8$.
6. Compare the above values and keep only the lower value using $T9$.
7. Perform steps (3) to (5) until all of the opponent's moves have been evaluated and the move yielding the lowest value has been selected.
8. Perform steps (2) to (6) until all of the machine's moves have been evaluated.
9. Perform the move assigned to the highest value.

2.4 Ideas for further improvements

In his paper Shannon discusses some obvious concerns regarding the strategy A algorithm's performance. He estimated that a machine would take over 16 minutes to perform a single move. Since Shannon's goal was to create a machine to match human skill levels this simply wouldn't suffice. He also disliked the fact that the machine could only see three moves deep at any point in time.
His main point of criticism, however, was that the algorithm behaved exactly the same way in stable positions as it did in unstable positions. He considered an evaluation during an exchange or a combination to be completely unnecessary since the follow-up move would be evident in these types of scenarios. Good human players generally only evaluate a limited amount of selected variations until stability is reached. Shannon wanted his algorithm to be intelligent enough to mimic this human trait.

In order to eliminate these weaknesses he wanted the algorithm to evaluate forceful variations such as captures, recaptures or checks out as far as possible, while at the same time giving up unreasonable positions fairly early. Shannon also saw the need for an additional function which would specify whether the game was in a stable or unstable position. He would use this function to make sure the evaluation function is only applied in relatively stable board positions.

3 A more advanced selective search strategy

3.1 Shannon's Type B strategy

Taking the previously discussed problems into consideration, Shannon came up with a type B strategy. This strategy was equipped with a function $g(P)$ used to determine whether stability exists in a position P or not, e.g. $g(P) = 0$ if a piece is attacked by a piece of lower value. Shannon wanted his algorithm to explore any variation at least two moves deep. If after two moves $g(P) = 1$, he designed his algorithm to continue exploring the variation until either the depth exceeded 10 or $g(P) = 0$.

Lastly, he defined a function $h(P, M)$ in order to decide whether a move M in a position P is worth exploring or not. He acknowledged that such a function would be difficult to construct since it should not eliminate moves which only look bad at first sight. Shannon, however, believed that sufficient research into the theory of chess would yield such a function. The plan was for the function to assume high values for forceful moves such as checks, captures and attacking, medium values for defensive moves and low values for all other moves. In addition, the requirements on h would be set higher and higher as the depth of search increased, resulting in fewer and fewer sub-variations being examined.

Shannon figured that this new strategy would significantly improve computing efficiency and would therefore result in the machine playing a fairly strong game at human speed. In constructing his algorithm Shannon made sure to make use of the four main advantages machines have over humans

- *Speed* – A machine's ability to perform calculations at a very high speed.
- *Freedom from errors* – Assuming that the master has made no mistakes in implementing the algorithm, there should be no errors during the execution.
- *Freedom from laziness* – While humans might make the mistake to pursue an instinctive move without properly evaluating it, machines have no gut feeling and will therefore never perform any operation without analyzing it according to the algorithm.
- *Freedom from nerves* – Unlike humans, a machine will not resign once it feels it has lost, nor will it become overconfident if it thinks it has won the game.

While these represent important advantages, Shannon knew that he must weigh his algorithms abilities against the "flexibility, imagination and inductive learning capacities of the human mind" [6].

3.2 Turing's strategy

Turing followed a similar approach to that of Shannon's type B strategy. He wanted his machine to be capable of deeming certain moves pointless while pursing others quite a long way down the path. At the bare minimum he wanted his algorithm to follow each variation for at least two white moves and two black moves before evaluating whether this has lead to a *dead* or a *considerable position*. Turing defined a dead position to be a position in which "there are no checks, captures or recaptures possible in the next move" [8] while all other positions are regarded as considerable positions. Once the algorithm has evaluated to a depth of four, the evaluation only continues if it results in a considerable position. At the end of such a sequence, Turing's evaluation function is applied to determine the value of the position. Identical to Shannon's strategy, Turing made use of minimizing opponent's values and maximizing the machine's values to back-propagate the values through the decision tree in order to determine the overall best move.

Turing's algorithm calculates the overall value of a position using the *value* and the *positional value*. The *value* is calculated by assigning relative values to every piece on the board and summing the values for all pieces. The positional value, however, consists of a more complex evaluation. Some of the attributes used are listed below:

- *Mobility* - For Q,R,B,N, add the square root of the number of moves the piece can make. In addition count each capture as two moves.
- *Piece safety* - For R,B,N, add 1.0 point if it is defended, add 1.5 points if it is defended more than once.
- *King mobility* Do the same as in (1) for the King except for castling moves.

3.3 Problems to consider

Shannon's described his biggest concern to be the lack of variation in play. Since his machine would always make the same move in the same position and would therefore play the same game if the opponent stuck to the same moves, the opponent could make use of weak positions previously made in the game flow. He suggested to add a statistical element to the game by choosing at random between multiple moves of similar value.

Both Shannon's and Turing's algorithms are incredibly complicated in the opening stages of the game. Human players tend to choose from a set number of variations until one of the players deviates from the book. Shannon wanted to implement this by supplying the machine with a number of standard opening variations saved in the internal

memory. His machine would then rely on these stored variations in the beginning stages of the game instead of calculating the next move in a complicated manner.

4 Conclusion

In assessing his algorithm, Shannon mentioned that it is his belief that the machine would play chess "brilliantly" [6]. However, he admitted that the machine's greatest weakness is its inability to learn from previous mistakes.

> It plays something like a beginner at chess who has been told some of the principles and is possessed of tremendous energy and accuracy for calculation but has no experience with the game.
>
> Claude E. Shannon in Programming Digital Computer to Play Chess, page 273 [6]

While very advanced for their time, both Shannon's and Turing's strategies rely on brute force calculation rather than logical analysis or even intelligence. Shannon agreed that a more intelligent self-improving system could be constructed, he however argued that this would not be a practical approach for the problem of chess.

> The computer is strong in speed and accuracy and weak in analytical ability and recognition. Hence, it should make more use of brutal calculations than humans.
>
> Claude E. Shannon in Programming Digital Computer to Play Chess, page 274 [6]

He suggested instead using a higher level program which could alter the terms and coefficients constructing the evaluation function depending on results of games played by the machine.

It is important to note that the performance of both Shannon's and Turing's algorithm depends greatly on the selected coefficients and numerical factors used in the evaluation function. The style of play as well as the strength of the player can be adjusted by adding, omitting or changing certain terms in the evaluation function. Simply put, the algorithm is only as good as its evaluation function and therefore only as good as its masters understanding of chess.

> If I were to sum up the weakness of the above system in a few words I would describe it as a caricature of myself.
>
> Alan Turing in Digital Computers applied to Games, page 9 [8]

References

[1] Sameep Bagadia, Pranav Jindal, and Rohit Mundra. "Analyzing Positional Play in Chess using Machine Learning". In: 2014.

[2] Casey Chan. "The Rise of Artificial Intelligence Is Absolutely Fascinating". In: *GIZMODO* (Nov. 2013).

[3] Jamie Condliffe. "Watch Garry Kasparov and Alan Turing Play Ches". In: *GIZMODO* (June 2012).

[4] Ioan James. *Claude elwood shannon 30 april 1916—24 february 2001*. 2009.

[5] Irene Meichsner. "Der Urahn der Nerds". In: *Deutschlandfunk Kultur* (June 2012).

[6] Claude E. Shannon. "XXII. Programming a computer for playing chess". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.314 (1950), pp. 256–275.

[7] Chris Smith et al. "The history of artificial intelligence". In: *University of Washington* (2006), p. 27.

[8] Alan M Turing. "Digital computers applied to games". In: *Faster than thought* (1953).

Statement of originality

This is to certify that to the best of my knowledge, the content of this term paper is my own work. This term paper has not been submitted for any other purposes.
I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this term paper and sources have been acknowledged.

Jacqueline Wagner, Heidelberg, May 8, 2019