Die Funktionsweise von Schachcomputern

Vorwissenschaftliche Arbeit verfasst von:

El-Shennawi Mahmoud

Klasse 8iz

Schuljahr 2019/2020

Betreuungslehrer: Mag. Hans-Jürgen Koller

Abgabedatum:

BORG Linz

4020 Linz, Honauerstraße 24

Eidesstattliche Erklärung

Ich erkläre, dass ich die vorwissenschaftliche Arbeit eigenständig angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

Ort, Datum: Unterschrift:

_____ _____

Abstract

Schach ist mittlerweile ein sehr altes Spiel. Über die letzten Jahrhunderte wurden viele Techniken, Stellungen und Strategien entwickelt, um das Beste aus dem Schachspiel herauszuholen. Mit der Erfindung des Computers hatte man angefangen auch Algorithmen zu entwickeln, die, durch Berechnungen, Schachzüge und Techniken nicht nur entwickeln können, sondern auch Vorhersagen können.

Wie hat sich die Geschichte der Schachprogrammierung entwickelt? Wie ist die Funktionsweise von Schachcomputer? Wie sind sie programmiert? Was sind Zuggenerator und Minimax-Suche? Wie sieht der Code zu einem Schachalgorithmus aus? Welche Probleme treten auf bei Schachalgorithmen? Welche Stellungen kann ein Schachprogramm nicht lösen?

In dieser vorwissenschaftlichen Arbeit befasse ich mich literarisch mit diesen Fragen.

Danksagung

Ich möchte mich an erster Stelle bei meiner Mutter, Assma Mohamed, bedanken, da sie stets für mich da war und ich mich immer auf ihre Unterstützung verlassen konnte.

Weiterhin möchte ich mich bei Herrn Mag. Dominik Maresch bedanken, da ich, durch ihn, meine Leidenschaft zum Schach entdecken konnte.

Ich möchte mich bei meinem Betreuer, Mag. Hans-Jürgen Koller, für eine ausgesprochen gute Betreuung während des Arbeitsprozesses bedanken.

Ich möchte mich bei meiner Klassenkameradin Faten Metwally bedanken, da sie mich in den letzten 4 Schuljahren mental unterstützt hat und ich mich immer auf sie verlassen konnte.

Anschließend möchte ich ein großes Dankeschön an meiner Klassenvorständin Mag. Erika Salcher richten, da sie mich, seit der siebten Klasse, in jeder Hinsicht unterstützt hat.

Inhalt

1 Einleitung

Schachprogramme sind etwas Zentrales in der Schachszene. Sie werden verwendet, um Stellungen zu analysieren und, um für bestimmte Stellungen den bestmöglichsten Zug berechnen zu können.

Die momentan am meisten benutzten Schachprogramme basieren auf der Bewertungsfunktion. Die Bewertungsfunktion versucht im Prinzip, durch Bewertung der Figuren auf dem Feld, zu errechnen welcher Spieler im Vorteil ist.

Das Hauptziel dieser Arbeit ist es, die Komponenten des klassischen Schachcomputers näher zu erläutern. In der folgenden vorwissenschaftlichen Arbeit werde ich zuerst Schach allgemein definieren, über die Entwicklung von Schachcomputern schreiben, danach die einzelnen Komponenten eines Schachalgorithmus erklären und gebe gleichzeitig Beispiele dafür, was Schachprogramme können und woran sie scheitern. Daraufhin kommt ein Beispiel für ein Schachalgorithmus anhand von Pseudocode. Anschließend werde ich noch weitere Beispiele anhand von Schachpositionen bzw. Schachsituationen geben.

2 Definition von Schach

Schach leitet sich vom persischen Wort šāh Schah / شاه /„König“ ab. Schach ist vermutlich im persisch/arabischen Bereich entstanden. In diesem Strategiespiel bewegen zwei Spieler abwechselnd Figuren auf einem 64-Felder Brett. Jeder Spieler*in besitzt 16 Figuren: Acht Bauern, zwei Türme, zwei Springer, zwei Läufer, einer Dame und einem König. Es gibt für gewöhnlich weiße und schwarze Figuren, um sie unterscheiden zu können. Das Ziel des Spiels ist es, den gegnerischen König schachmatt (persisch: شاه مات, ,,Der König ist gefallen/tot“) zu setzen. Ein Schachmatt wird erreicht indem der König so angegriffen wird, dass keine Möglichkeit zur

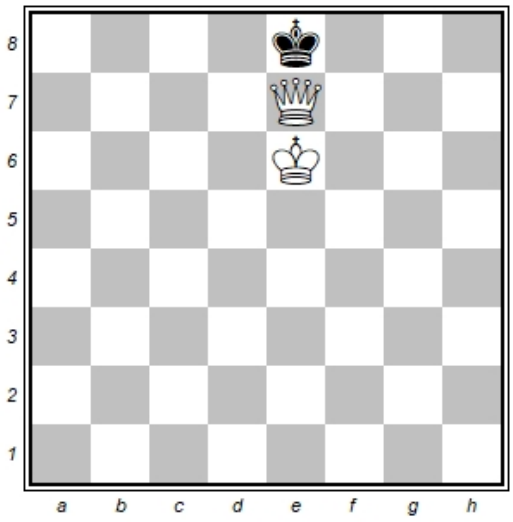

Abbildung 1: Beispiel für ein Schachmatt

Abwehr oder zur Flucht besteht (siehe Abbildung 1).[1]

Sollte kein Spieler den Gegner in ein Matt setzen können, so endet die Partie als unentschieden (Remis). [2]

Nun beschäftigen wir uns mit der Gangart der Schachfiguren.

2.1 Aufbau des Schachbretts und Gangart der Schachfiguren

Ein Schachbrett besteht aus 8x8 (64 gleichgroße Quadrate) Felder. Die Farben dieser Quadrate wechseln sich zwischen Weiß und Schwarz ab. Im Regelbuch steht, dass das Schachbrett so zwischen den beiden Schachspielerinnen bzw. Schachspielern gelegt wird, dass das rechte Quadrat unten rechts weiß ist (siehe Abbildung 2).[3]

Für die Gangart der Figuren gelten strikte Regeln.

Man darf mit keiner Figur auf ein Feld ziehen, welches von einer Figur der gleichen Farbe besetzt wird. Solle eine Figur auf ein Feld ziehen, worauf sich eine gegnerische Figur befindet, wird diese geschlagen und vom Schachbrett entfernt. Eine Figur greift eine andere Figur an, wenn sie droht diese im nächsten Zug zu schlagen.[4]

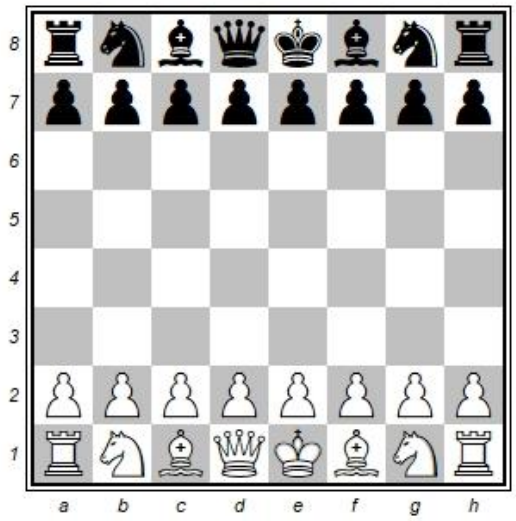

Abbildung 2: Klassischer Aufbau eines Schachbretts

[1] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]
[2] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]
[3] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]
[4] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

Der Läufer darf entlang der Diagonale, auf die er steht, fahren. [5] (siehe Abbildung 3)

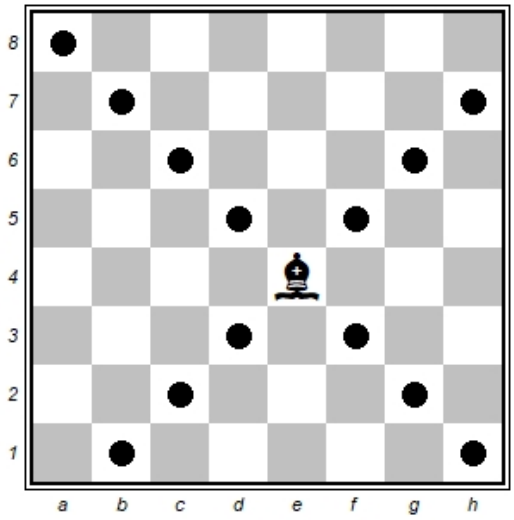

Abbildung 3: Gangart des Läufers

Der Turm darf beliebig weit vertikal oder horizontal fahren, ausgehend vom Feld, auf dem er steht. [6] (siehe Abbildung 4)

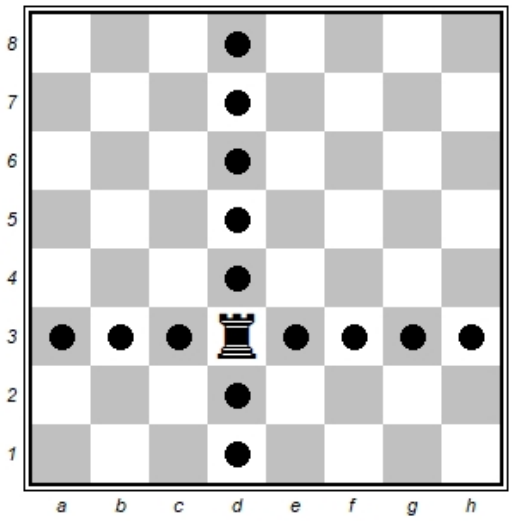

Abbildung 4: Gangart des Turmes

[5] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

[6] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

Die Dame darf beliebig weit horizontal, vertikal und diagonal fahren, ausgehend von dem Feld, auf dem sie steht. [7] (siehe Abbildung 5)

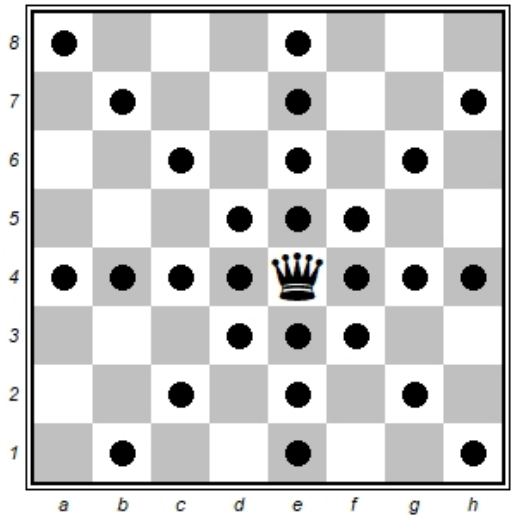

Abbildung 5: Gangart der Dame

Läufer, Türme und Damen dürfen über dazwischen liegende Figuren nicht ziehen.[8]

Der Springer ist die einzige Figur, die über dazwischen liegende Figuren ziehen darf:

> „Der Springer darf auf eines der Felder ziehen, die seinem Standfeld am nächsten, aber nicht auf gleicher Linie, Reihe oder Diagonalen mit diesem liegen.“ [9] (siehe Abbildung 6)

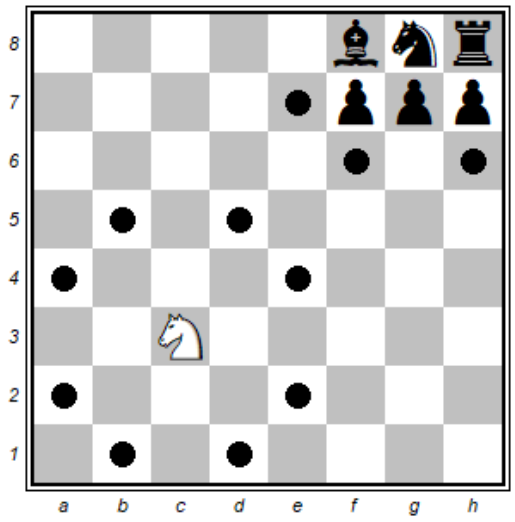

Abbildung 6: Gangart des Springers

[7] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

[8] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

[9] Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

Der Bauer bewegt sich Vorwärts auf das Feld, das direkt vor ihm ist, wenn es nicht besetzt ist. In seinem ersten Zug darf er jedoch auch 2 Felder ziehen, wenn diese nicht besetzt sind. Ein Bauer darf, die von ihm diagonal liegende Figur, schlagen (Siehe Abb. 7).[10]

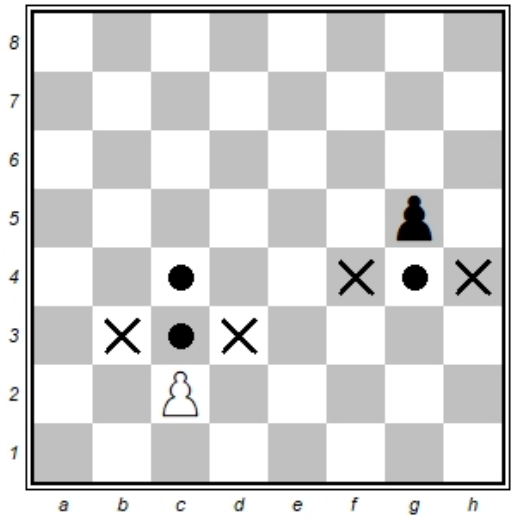

Abbildung 7: Gangart des Bauers

Die Regel, die zulässt, dass ein Bauer anfangs 2 Schritte gehen darf, lässt sich von einem gegnerischen Bauern kontern:

„Ein Bauer, der auf derselben Reihe auf einem unmittelbar angrenzenden Feld wie ein gegnerischer Bauer, der soeben zwei Felder von seiner Anfangsstellung vorgerückt ist, steht, darf diesen gegnerischen Bauern so schlagen, als ob letzterer nur um ein Feld vorgerückt wäre. Dieses Schlagen ist nur in dem unmittelbar nachfolgenden Zug regelgemäß und wird „Schlagen en passant“ genannt.“ [11] (siehe Abbildung 8)

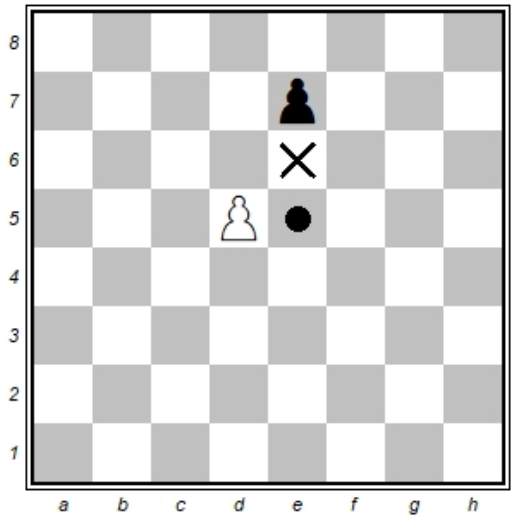

Abbildung 8: En-passant

[10] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

[11] Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

Sollte man es schaffen mit einem Bauern auf die andere Seite des Feldes zu kommen, darf man den Bauern gegen eine bereits geschlagene Figur austauschen.[12]

Der König darf auf ein beliebiges Feld ziehen, dass neben ihm ist.[13] Dies wird in Abbildung 9 dargestellt.

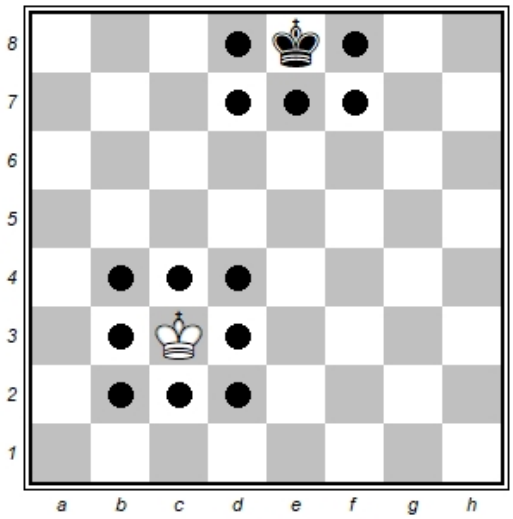

Abbildung 9: Gangart des Königs

Eine andere Art den König zu bewegen wäre die Rochade.

Bei der Rochade wird der König zwei Felder in Richtung einer seiner Türme bewegt und der Turm wird auf das Feld gesetzt, welches der König überqueren musste.[14]

Hierbei unterscheidet man zwischen der großen und der kleinen Rochade (siehe Abbildung 10 & Abbildung 11)

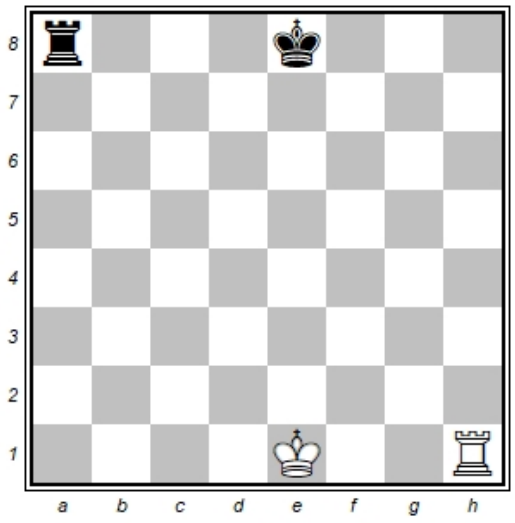

Abbildung 10: Weiß vor kleiner Rochade, Schwarz vor großer Rochade

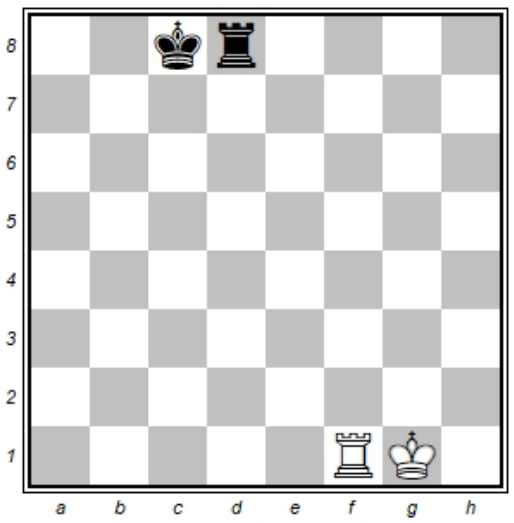

Abbildung 11: Weiß nach kleiner Rochade, Schwarz nach großer Rochade

[12] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

[13] ebd.

[14] vgl. Alt, Ralph; Deventer, Klaus; Klünser, Jürgen; Strobl, Thomas; Wiedmann, Thomas; https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 [letzter Zugriff: 02.02.2020]

2.2 Koordinatensystem

Durch ein Koordinatensystem kann man im Schach bestimmte Züge beschreiben, ohne sie visualisieren zu müssen. Dabei wird die x-Achse eines Schachbretts mit Kleinbuchstaben (von a bis h) und die y-Achse mit Zahlen (von 1 bis 8) gekennzeichnet. Die Figuren kriegen jeweils ein Kürzel, welches meistens der Erste Buchstabe der Figur im Englischen ist (z.B.: Dame/Queen – Q). Der Bauer hat jedoch kein Kürzel. Die Position einer Figur setzt sich somit aus dem Kürzel, der Position auf der x-Achse und der Position auf der y-Achse zusammen (z.B.: Qe7 – siehe Abb. 13). Zieht eine Figur von einem Feld auf ein anderes so wird die alte Position und dann die neue Position angegeben. Schlägt eine Figur einen andere, wo wird das meistens mit einem „x“ oder einem „:“ zwischen den jeweiligen Positionen angegeben. Rochaden werden mit „0-0“ (kleine Rochade) und mit „0-0-0“ (große Rochade) gekennzeichnet. Wenn eine Figur Schach gibt dann wird dies meistens mit einem „+“ gekennzeichnet und wenn es ein Matt ist dann wird dies meistens mit einem „#“ gekennzeichnet

Beispiele: e2-e4 Der Bauer auf e2 fährt nach e4

Sf6xd5 Der Springer auf f6 schlägt nach d5

Lf1-a6 Der Läufer auf f1 zieht nach b5

Te8xe1 Der Turm auf e8 schlägt nach e1

Dd1-h8 Die Dame auf d1 zieht nach h5

Kg8-h8 Der König auf g8 zieht nach h8

e4xf3ep Der Bauer auf e4 schlägt nach f3 „en passant“

0-0 Kleine Rochade

0-0-0 Große Rochade

Df1+ Dame auf f1 gibt Schach

Da7# Dame auf a7 gibt Matt

Im nächsten Kapitel befasse ich mich mit der Geschichte von Schachcomputern.

3 Die Geschichte von Schachcomputern

Der erste bekannte Fall eines „Schachautomaten“ stammt von 1769. Der Schachtürke wurde vom ungarischen Hofrat Wolfgang von Kempelen auf Wunsch der österreichischen Kaiserin Maria Theresia gebaut. Er bekam diesen Namen, da er wie ein Türke zu dieser Zeit angezogen war (siehe Abbildung 12). Er hatte Lebensgröße und war hinter einem Kasten mit einem Schachbrett darauf. Vor jedem Spiel wurde das Innere des Kastens gezeigt. Man konnte Schrauben, Gestänge und Zahnräder erkennen. Der Schachtürke zog so viel Aufmerksamkeit auf sich, dass sogar bekannte Menschen (z.B. Joseph II.) in der damaligen Zeit gegen ihn spielen wollten. Später fand man heraus, dass die Schachspieler Mouret, Johann Baptist Allgeier und Schlumberger im Kasten saßen und dem Schachtürken somit zum Sieg verholfen haben. 1854 wurde der Schachautomat durch ein Feuer im Chinesischen Museum in Philadelphia zerstört.[15]

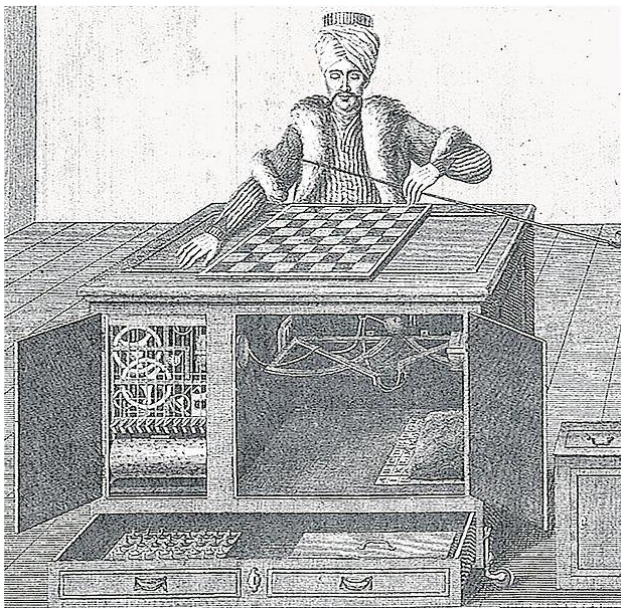

Abbildung 12: Der Schachtürke

Man hatte immer wieder Ideen, was eine Schachmaschine angeht. Diese wurden aber nie realisiert. Erst mit der Erfindung des Computers schien es möglich. 1947 gelang es Alan Turing und Champernowne tatsächlich das erste Schachprogramm zu schreiben. Da Turing aber nicht genug Computer zur Verfügung hatte musste er vor jedem Zug alle Möglichkeiten des Computers händisch bewerten. Um das zu vereinfachen, gab er den Figuren einen „Materialwert“: Schachfiguren sind unterschiedlich stark, somit ist es Beispielsweise gut, wenn man eine schwache Figur opfert, um eine stärkere Figur des Gegners zu schlagen. Er gab dem König 1000 Punkte, der Dame 10 Punkte, dem Turm 5 Punkte, dem Läufer und dem Springer 3 Punkte und dem Bauern einen Punkt. Er bewertete auch die Positionierung der Figuren.[16]

[15] vgl. Bauermeister, Karsten, http://www.schachcomputer.at/gesch1.htm [letzter Zugriff: 01.02.2020]
[16] vgl. Bauermeister, Karsten, http://www.schachcomputer.at/gesch2.htm, http://www.schachcomputer.at/gesch3.htm [letzter Zugriff: 01.02.2020]

1950 begannen amerikanische Wissenschaftler an der Entwicklung von Schachprogrammen. „Los-Alamos“ wurde 1956 fertiggestellt. Damit dieses Programm aber richtig funktioniert musste das Schachspiel eingeschränkt werden. Man spielte auf einem 6x6 Schachbrett, es gab keine Läufer, keine Rochaden und Bauern durften keine Doppelschritte machen. Trotz dieser Einschränkungen und sehr starker Hardware brauchte das Programm für eine durchschnittliche Partie knapp 10 Stunden.[17]

Ein weiteres wichtiges Programm ist Mephisto. Das Programm erschien Anfang der 80iger Jahre. Über die Jahre wurde es immer mehr verbessert, sodass es auf der Mikro-WM (Weltmeisterschaft der Schachprogramme) mehrmals zum Weltmeister wurde.[18]

Das Problem für den Computer hierbei ist die Menge an Züge. Denn bereits nach 2 Zügen können über 70 000 verschiedene Stellungen entstehen.

Von da an entwickelten sich Schachcomputer immer weiter. Mittlerweile sind Schachcomputer nicht mehr wegzudenken. Mit Schachcomputer werden Züge berechnet, Strategien entwickelt, Partien analysiert und viel mehr.

Fritz ist eine Serie von Schachprogrammen, die (bis zur 13. Version) von Frans Morsch und Mathias Feist im Auftrag von der deutschen Schach-Software Firma „Chessbase“ entwickelt wurde. Die aktuellste Version von Fritz ist „Fritz 17“, welche auch „Fat Fritz“ beinhaltet. Fat Fritz basiert auf einem neuronalen Netzwerk und ist somit eine künstliche Intelligenz, welche dazu lernen kann.[19]

Stockfish ist ein open source Programm (bei open source Programmen kann man frei auf den Quelltext zugreifen), das von Tord Romstad, Marco Costalba, Joona Kiiski und Gary Linscott entwickelt wurde und da es in C++ geschrieben ist, lässt es sich ohne Probleme auf mehrere Plattformen ausführen (Android, iOS, Linux, OS X und Windows)[20]

[17] vgl. Bauermeister, Karsten, http://www.schachcomputer.at/gesch4.htm, http://www.schachcomputer.at/gesch5.htm [letzter Zugriff: 01.02.2020]
[18] vgl. Bauermeister, Karsten, http://www.schachcomputer.at/gesch14.htm, http://www.schachcomputer.at/gesch16.htm, http://www.schachcomputer.at/gesch18.htm, http://www.schachcomputer.at/gesch21.htm [letzter Zugriff: 01.02.2020]
[19] vgl. Isenberg, Gerd, https://www.chessprogramming.org/Fritz [letzter Zugriff: 19.02.2020]
[20] vgl. Isenberg, Gerd, https://www.chessprogramming.org/Stockfish [letzter Zugriff: 19.02.2020]

Houdini ist ein closed source Engine und wurde zur nicht-kommerziellen Nutzung von Robert Houdart entwickelt. Das Programm kam 2010 raus, Houdini 2 kam 2011 raus, Houdini 3 2012 und Houdini 4 kam 2013.[21]

AlphaZero ist ein Schachprogramm, welches mit einem neuronalen Netz funktioniert, statt mit linearen Funktionen und Bewertungsfunktionen, wie es bei klassischen Schachprogrammen üblich ist. Diese Arbeit beschäftigt sich jedoch ausschließlich mit Schachprogrammen, die die Bewertungsfunktion benutzen.

Im nächsten Kapitel befassen wir uns mit dem Aufbau und Teilalgorithmen des Schachalgorithmus.

4 Grundlegender Aufbau eines Schachalgorithmus

4.1 Die Bewertungsfunktion

Die Bewertungsfunktion ist eine Funktion, die auf Basis eines Spielbretts (hierbei spielt die Spielsituation keine Rolle) eine Bewertungszahl zurückliefert.[22]

Im Prinzip gibt kann die Bewertungsfunktion 3 Werte zurückgeben: 1 für „Weiß gewinnt“, -1 für „Schwarz gewinnt“ und 0 für „unentschieden“. Das setzt aber voraus, dass die Funktion immer alle Möglichkeiten und Ausgänge bis zum Ende der Partie berechnen muss, welches aufgrund der Anzahl der Möglichkeiten, lange dauern würde. Die Bewertungsfunktion muss also für einzelne Züge oder Situationen funktionieren. Um das zu lösen basiert die Bewertungsfunktion auf die Idee des „Materialwerts“ von Turing, welches in Kapitel 2 bereits erwähnt wurde. Jedoch bekommt die Dame einen Wert von 9 Punkten, statt 10 Punkten. Es werden die Materialwerte der Figuren, die sich noch auf dem Spielfeld befinden, aufsummiert. Sollten Schwarz und Weiß nun Figuren exakt gleichen Wertes besitzen, dann gibt die Bewertungsfunktion den Wert 0 zurück. Das bedeutet, dass die Situation ausgeglichen ist. Bei einem positiven Wert ist Weiß im Bezug auf die Materialien am Gewinnen und bei einem negativen Wert ist Schwarz in Bezug auf die Materialien am Gewinnen. Die Funktion erkennt

[21] vgl. Isenberg, Gerd, https://www.chessprogramming.org/Houdini [letzter Zugriff: 19.02.2020]
[22] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

auch ein Schach, Matt, Remis oder Patt. Zudem wird bei der Bewertungsfunktion nur das Material bewertet und nicht die Lage der Figuren. [23]

Für die Stellungsbewertung sind andere Algorithmen notwendig, die jedoch nicht Teil dieser Arbeit sind.

Die folgenden Beispiele wurden durch das Schachprogramm Stockfish 11+ auf der Webseite www.lichess.org erstellt.

Abbildung 13: Beispiel für die Bewertungsfunktion

In Abbildung 13 sieht man ein Beispiel zu einem Endspiel. Oben rechts sehen wir Folgendes: Der Name des Schachprogramms, das hier arbeitet (in diesem Fall Stockfish 11+), darunter die Tiefe, also die Anzahl der Züge, die das Programm berechnet. Man könnte durch das weiß/blaue „+“ Zeichen daneben die Tiefe erhöhen. Somit würde es zwar länger rechnen, da durch das erhöhen der Tiefe nun mal mehr Züge berechnet werden, würde aber höchstwahrscheinlich einen besseren Zug empfehlen. Wir sehen außerdem noch den Wert -5,3. Das ist einerseits der Wert, den die Bewertungsfunktion zurückgegeben hat und weil der Wert negativ ist bedeutet das, dass Schwarz um 5,3 Bauern (Die Einheit, in der das gerechnet wird) besser steht als Weiß. Der vertikale Balken zwischen Brett (links) und der Konsole

[23] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

(rechts) visualisiert den momentanen Stand. Er ist überwiegend schwarz. Das heißt, dass Schwarz in der momentanen Stellung besser steht. Schwarz ist aktuell am Zug. Auf dem Spielbrett sieht man, dass der beste Zug, der vom Schachprogramm ermittelt wurde, als Pfeil auf dem Spielbrett zu sehen ist. Der Pfeil zeigt vom schwarzen Läufer, der auf d8 ist, auf e7. Nach den Berechnungen des Programms wäre der beste Zug somit Le7.

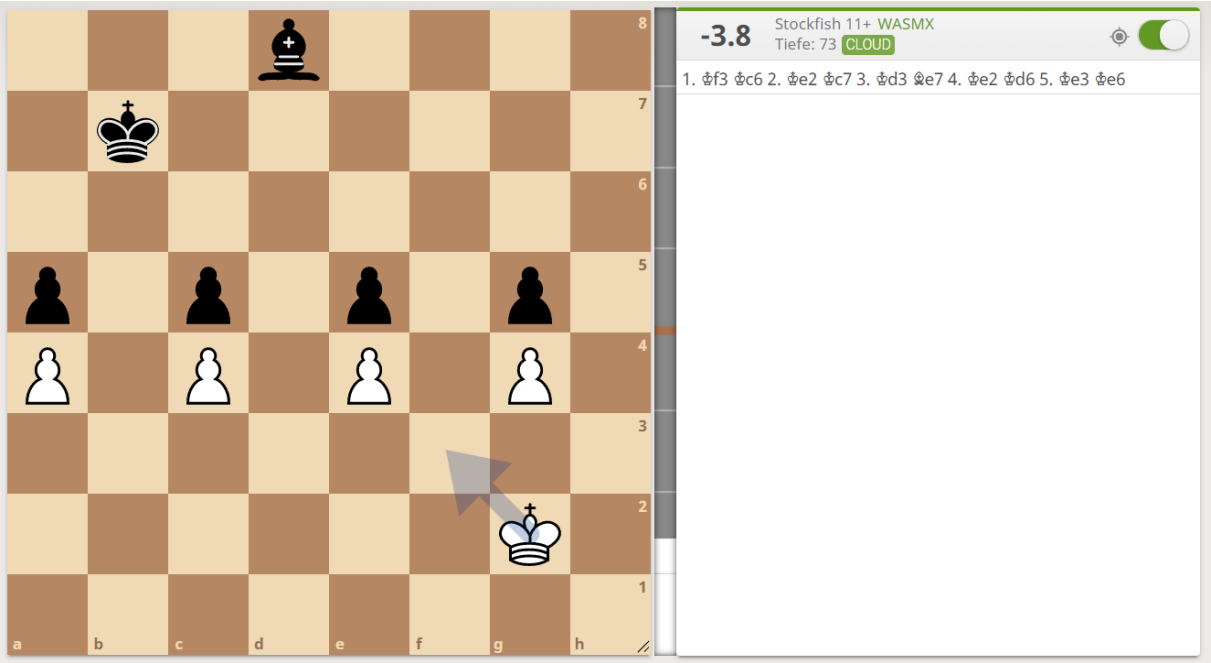

Abbildung 14: Beispiel für das Versagen der Bewertungsfunktion

In Abbildung 14 sieht man ein deutliches Beispiel für das Versagen der Bewertungsfunktion. Die Bauern blockieren sich gegenseitig, der schwarze Läufer wird von den schwarzen Bauern blockiert und die Könige können nicht auf die andere Seite, da sie sich sonst selber, durch die gegnerischen Bauern, ins Schach stellen würden. Der Ausgang dieses Spieles ist ein Remis.

Dennoch gibt das Programm einen Wert von -3,8 aus, weil Schwarz einen Läufer mehr hat als Weiß. Das Problem bei der Bewertungsfunktion ist somit, dass sie fast nur das Material bewertet, obwohl die Stellung in jeder Hinsicht ein Remis ist.

4.2 Der Zuggenerator

Ein Schachprogramm braucht einen Zuggenerator. Der Zuggenerator ist eine Funktion, welche für eine Farbe alle möglichen und legalen Zügen (rekursiv) ermittelt (siehe Abbildung 15). Dies geschieht, um danach den besten Zug berechnen zu können.[24]

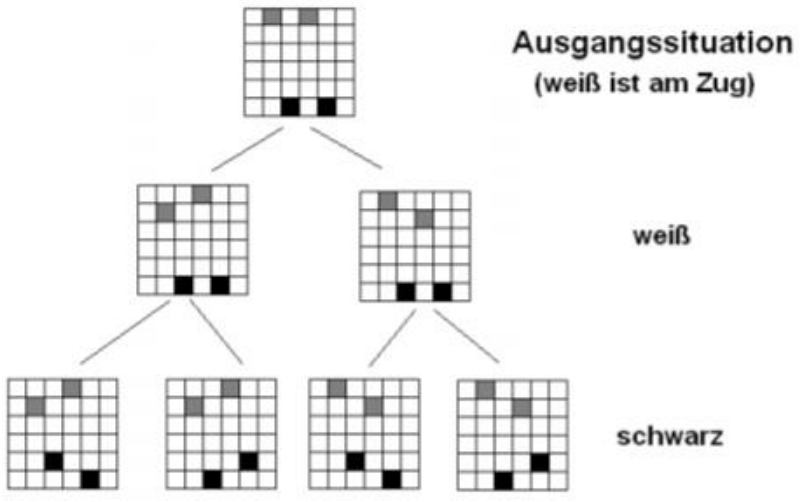

Abbildung 15: Rekursiver Aufruf des Zuggenerators

Beispielsweise gibt es beim ersten Zug 20 mögliche Züge für Weiß und 20 für Schwarz. Die Aufgabe des Zuggenerators ist, alle möglichen Züge durchzugehen, sich zu vergewissern, dass diese Züge auch Regelkomfort sind und die Züge anschließend zu notieren. In Abbildung 15 steht die Ausgangssituation ganz oben, dann werden die Möglichkeiten für Weiß erfasst und anschließend die Möglichkeiten für Schwarz.

4.3 Die Minimax-Suche

Um aus den Zügen, die beim Zuggenerator ermittelt werden, den besten Zug aussuchen zu können, wird die Minimax-Suche benutzt.

Das Prinzip der Minimax-Suche besteht darin, dass jeder Spieler den Zug spielen würde, der am besten geeignet ist. Für den weißen Spieler bedeutet das, dass er den Zug spielen muss, der bei der Bewertungsfunktion einen möglichst hohen positiven Wert ausgeben würde und für den schwarzen Spieler bedeutet das, dass er einen Zug spielen muss, der bei der Bewertungsfunktion einen möglichst hohen negativen Wert ausgeben würde. [25]

[24] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]
[25] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

Die Minimax-Suche arbeitet in Halbzügen. Angenommen Weiß wäre am Zug. Wie in Abbildung 16 zu sehen ist hat Weiß 3 Auswahlmöglichkeiten, die rekursiv bearbeitet werden, bis Suchtiefe zwei erreicht wird. Angefangen wird unten bei den Blattknoten, in welchen die Stellungen bewertet werden. Jetzt wird abwechselnd maximiert und minimiert. Das heißt, dass die Knoten jeweils den höchsten bzw. den niedrigsten Wert der Knoten darunter übernehmen, bis der beste Zug ermittelt wird. Wäre Schwarz am Zug, dann würde zuerst maximiert und dann minimiert werden.[26]

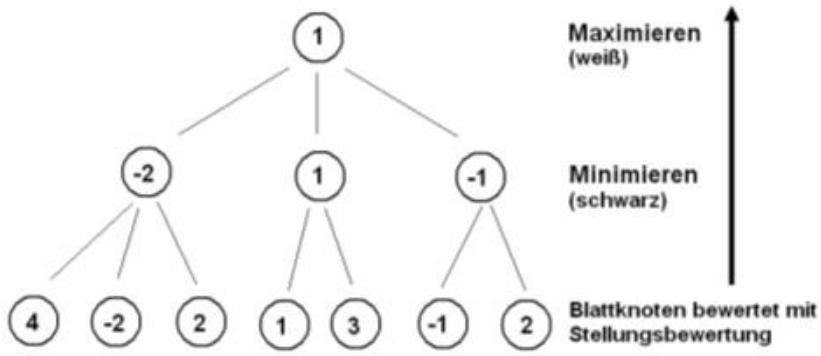

Abbildung 16: Die Minimax-Suche

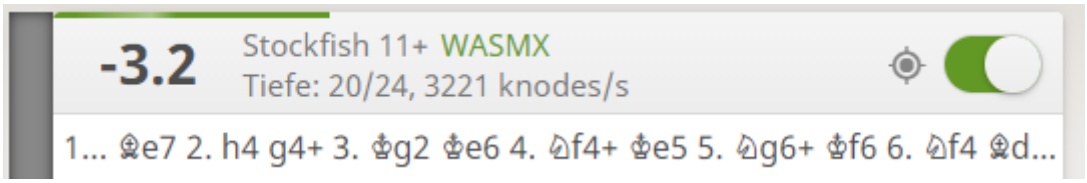

Abbildung 17: Die Konsole während des Rechnens

In Abbildung 17 sieht man die Konsole, während sie einen Zug berechnet. Man sieht auf welcher Tiefe das Programm gerade ist und wie viele Knoten pro Sekunde berechnet werden (3221 knodes/s).

[26] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

Abbildung 18: Beispiel zur Minimax-Suche

In Abbildung 18 ist ein Beispiel der Minimax-Suche zu sehen. Die weiße Dame hat sich auf das Feld c3 bewegt und ist somit durch den Bauer, der aktuell auf d4 steht, gefährdet. Die Minimax-Suche erkennt, dass der beste Zug für Schwarz das Schlagen der weißen Dame mit dem Bauern wäre, weil das Materialverlust für Weiß bzw. Materialgewinn für Schwarz ist und zeigt dies durch den Pfeil.

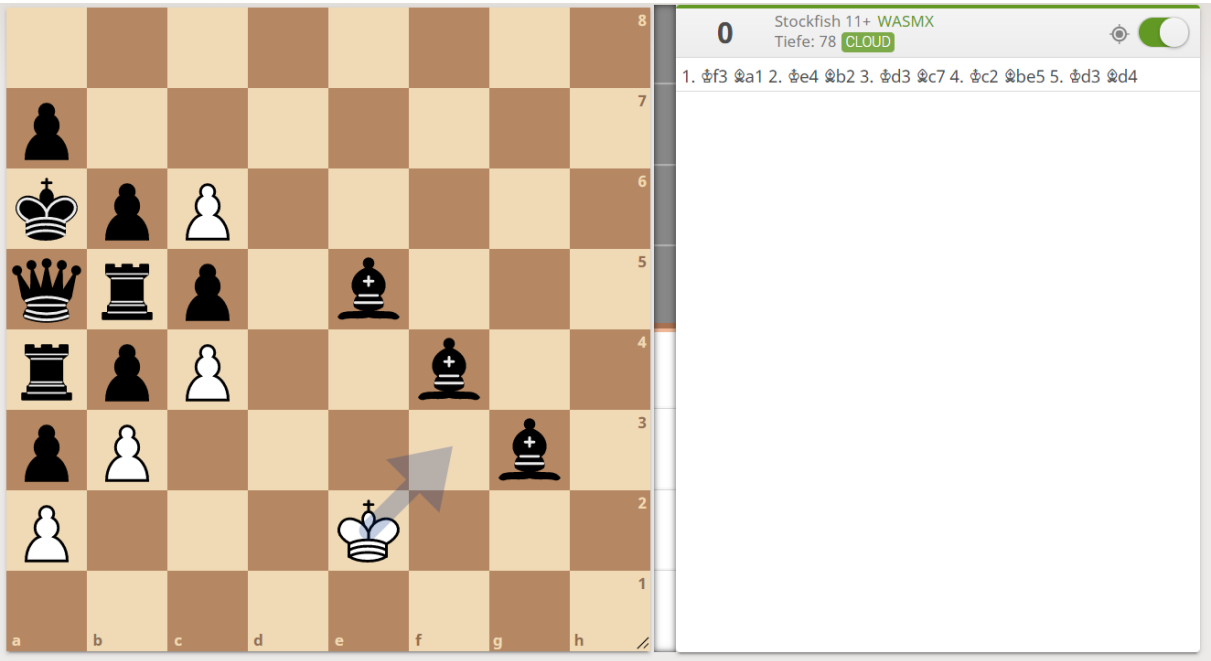

Abbildung 19: Versagen der Minimax-Suche

Betrachten wir nun ein Beispiel, bei dem die Minimax-Suche versagt. In Abbildung 19 sehen wir, dass das Programm den weißen König und die schwarzen Läufer um die „Festung“ auf der linken Seite herumspielen lässt, jedoch tut sich bei dieser Partie nicht mehr viel, wenn man nur das Programm spielen lässt.

4.4 Die Alpha-Beta-Suche

Mit der Alpha-Beta-Suche muss der Spielbaum (Abbildung 14) nicht komplett durchgespielt werden, wenn bestimmte Bedingungen erfüllt sind. Dies führt zu einer Geschwindigkeitserhöhung und Verringerung der benötigten Leistung ohne, dass ein schlechter Zug gespielt wird. Die Idee besteht darin, dass Äste abgeschnitten werden, wenn die Minimax-Suche ohnehin nur schlechte Züge ermitteln würde. Die Alpha-Beta-Suche wird auch Alpha-Beta-Cutoff genannt.[27]

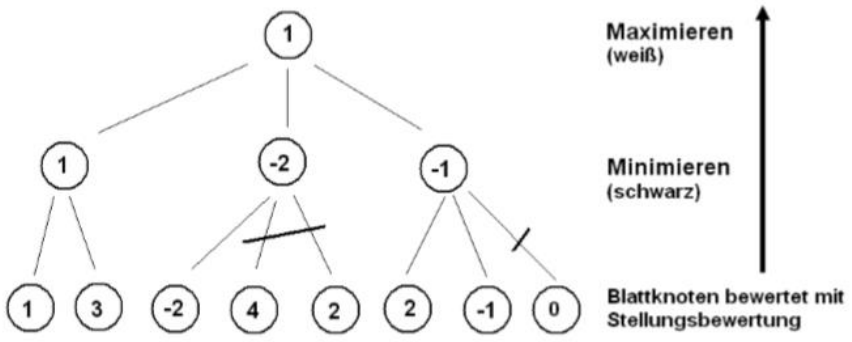

Abbildung 20: Alpha-Beta-Suche

[27] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

In Abbildung 20 sieht man ein Beispiel zur Alpha-Beta-Suche: Der Baum wird von links nach rechts bearbeitet und es wird minimiert. Der erste Knoten liefert den Wert 1. Diesen Wert gilt es zu überbieten. Im nächsten Knoten wird der Wert -2 geliefert. Da auf dieser Ebene minimiert wird, werden nur noch schlechtere Werte geliefert werden. Somit kann der Wert 1 nicht überboten werden, also können die restlichen Äste von diesem Knoten übersprungen werden und wir können den nächsten Knoten durcharbeiten. Der dritte Knoten liefert als erstes den Wert 2. Dieser ist größer als 1 und deshalb muss weitergerechnet werden. Als zweites wird der Wert -1 geliefert. Da wir immer noch minimieren und deshalb nur noch schlechtere Werte geliefert werden können, können wir auch hier mit der Berechnung aufhören. [28]

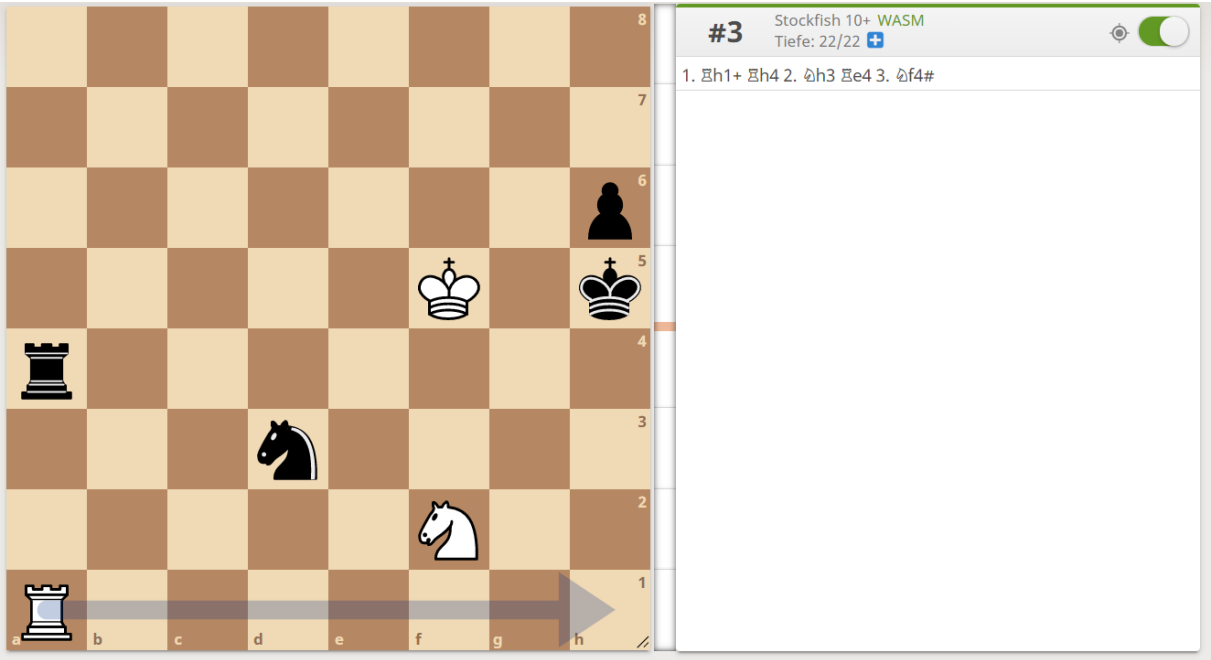

Abbildung 21: Beispiel für den Alpha-Beta-Cutoff

In Abbildung 21 ist zu sehen, dass das Programm direkt erkennt, dass es ein Matt in 3 Zügen für Weiß ist. Durch den Alpha-Beta-Cutoff muss es nicht alle Züge bewerten, sondern nimmt direkt den Zug, der zu einem Matt führen würde.

[28] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

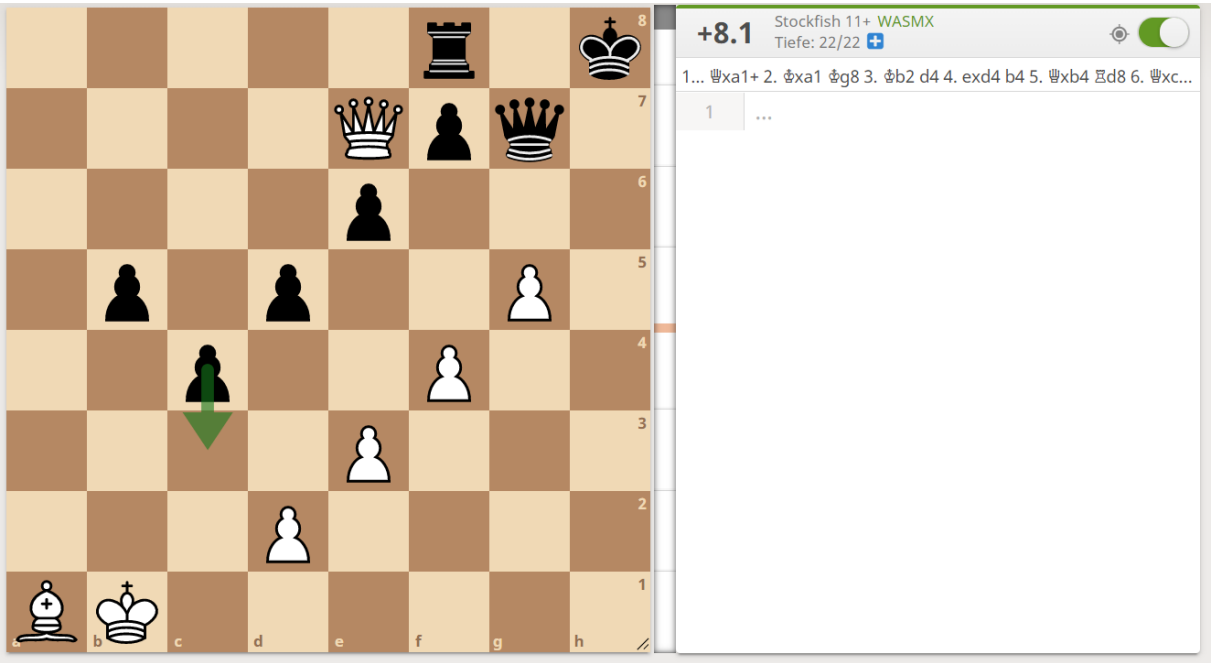

Abbildung 22: Versagen des Alpha-Beta-Cutoffs

In Abbildung 22 ist ein Beispiel zu sehen, bei dem der Alpha-Beta-Cutoff versagt. Schwarz ist am Zug, Weiß bedroht die schwarze Dame und Schwarz entscheidet sich dafür den Zug c3 zu spielen. Weil das Programm durch den Alpha-Beta-Cutoff nicht alle Züge berechnet, gilt die Dame für das Programm als „nicht verloren“, weil jeder Angriff vom Läufer auf die Dame von einem Bauer abgewehrt werden kann, bis der Läufer auf dem Feld f6 landet und vom Turm geschlagen wird. Das führt aber dazu, dass die weiße Dame auf e7 den Turm schlagen kann. Somit wurden 4 Bauern und ein Turm geopfert, um die Dame zu „retten“. Solches Versagen ist auch unter dem Namen „Horizon Effect“ bekannt.[29]

[29] vgl. Isenberg, Gerd, https://www.chessprogramming.org/Horizon_Effect [Letzter Zugriff: 19.02.2020]

```
int Max(int tiefe, int alpha, int beta)
 {
     if (tiefe == 0)
         return Bewerten();
     GeneriereMoeglicheZuege();
     while (ZuegeUebrig())
     {
         FuehreNaechstenZugAus();
         wert = Min(tiefe-1, alpha, beta);
         MacheZugRueckgaengig();
         if (wert >= beta)
             return beta;
         if (wert > alpha)
             alpha = wert;
     }
     return alpha;
 }
```

```
int Min(int tiefe, int alpha, int beta)
 {
     if (tiefe == 0)
         return Bewerten();
     GeneriereMoeglicheZuege();
     while (ZuegeUebrig())
     {
         FuehreNaechstenZugAus();
         wert = Max(tiefe-1, alpha, beta);
         MacheZugRueckgaengig();
         if (wert <= alpha)
             return alpha;
         if (wert < beta)
             beta = wert;
     }
     return beta;
 }
```

Abbildung 23: Pseudocode für die Minimax-Suche mit implementierten Alpha-Beta-Cutoff

Betrachten wir nun einen Pseudocode für die Minimax-Suche mit einem implementieren Alpha-Beta-Cutoff. In Abbildung 23 sind 2 Funktionen zu sehen. Die linke Funktion ist eine Funktion für das Maximieren und die rechte Funktion ist für das Minimieren. Im Folgenden werde ich diese Funktionen erklären.

Funktionen haben die Eigenschaft, dass sie einen Rückgabewert besitzen. Das heißt, dass man, nachdem die Funktion ausgeführt wurde, einen Wert, der in der Funktion berechnet wurde, geliefert bekommt. In diesem Fall kriegen wir einen integer (=int) zurück. Ein Integer ist eine ganze Zahl. Somit kriegen wir negative Zahlen, 0 und positive Zahlen. Negative Zahlen würden bedeuten, dass Schwarz am Gewinnen ist, positive Zahlen würden bedeuten, dass Weiß am Gewinnen und 0 steht für ein Unentschieden.

Die linke Funktion, welche fürs Maximieren zuständig ist, gibt einen integer zurück. Das ist an der ersten Zeile „int Max(…)“ erkennbar. Weiterhin benutzt diese Funktion insgesamt 3 Parameter: tiefe, alpha und beta (die Parameter „alpha“ und „beta“ sind für den Alpha-Beta-Cutoff zuständig). Alle 3 Parameter sind vom Typ integer. Es wird mit einer if-Abfrage überprüft, ob die Tiefe 0 ergibt. Sollte das der Fall sein, dann wird sofort eine Bewertung zurückgeliefert (return Bewerten();). Sollte das nicht der Fall sein, werden die möglichen Züge generiert und eine while-Schleife wird gestartet. Solange Züge übrig sind, werden folgende Schritte eingeleitet. Der nächste Zug wird ausgeführt, eine Variable namens „wert“ bekommt der bei der Minimierungsfunktion rauskommen würde und dann wird der Zug Rückgängig gemacht. In der Schleife finden dann noch 2 weitere if-Abfragen statt zum Vergleichen und feststellen vom Endergebnis. Sollte die Variable „wert“ größer oder gleich dem Parameter

„beta“ sein, so wird beta als Endergebnis zurückgegeben. Sollte die Variable „wert“ jedoch größer als der Parameter „alpha“ sein, so bekommt alpha einen neuen Wert: die Variable „wert“. Sobald keine Züge mehr vorhanden sind, endet die Schleife und der Parameter „alpha“ wird ausgegeben.

Die Funktion zum Minimieren (rechts auf Abbildung 23) funktioniert im Prinzip genauso wie die linke Funktion nur mit folgenden Unterschieden. Mitten in der while-Schleife, wird nicht die Minimierungsfunktion, sondern die Maximierungsfunktion aufgerufen. Außerdem sind die if-Abfragen umgekehrt. Das heißt, dass wir bei der ersten if-Abfrage nicht die Bedingung „wert >= beta“, sondern die Bedingung „wert <= alpha“ haben und demnach wird auch nicht „beta“ zurückgegeben, sondern „alpha“ („return alpha;“). Bei der zweiten if-Abfrage wird überprüft, ob die Variable „wert“ kleiner als „beta“ ist. Sollte das der Fall sein, dann bekommt der Parameter „beta“ den Wert von der Variable „wert“. Auch diese Schleife wird so lange wiederholt, bis die Tiefe den Wert 0 erreicht. Dann wird der Inhalt von „beta“ aus- bzw. zurückgegeben.

4.5 Eröffnungsbibliotheken

Um die Geschwindigkeit noch weiter zu erhöhen, stützen sich sehr viele Schachprogramme auf Eröffnungsbibliotheken. In diesen Bibliotheken stehen für den Anfang der Schachpartie schon bereits sehr gute Züge zur Verfügung, sodass die Schachprogramme erst später anfangen müssen die Züge zu berechnen. Es gibt auch Endspielbücher, die bereits fertige Endspielzüge beinhalten.[30]

Ein weiterer Vorteil von Eröffnungsbibliotheken ist, dass es Stellungen gibt, in der eine Seite vom Material her benachteiligt ist, aber dennoch eine bessere Stellung bzw. Aufstellung hat und der Computer dies nicht erkennt, weil er nur auf das Material schaut und durch die Eröffnungsbibliotheken kann der Computer dennoch erkennen, dass es sich hierbei um eine sinnvolle Eröffnung handelt.

[30] vgl. Reimers, Torben; Tietjen, Tobias; Wehr, Christian; http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf [Letzter Zugriff: 20.01.2020]

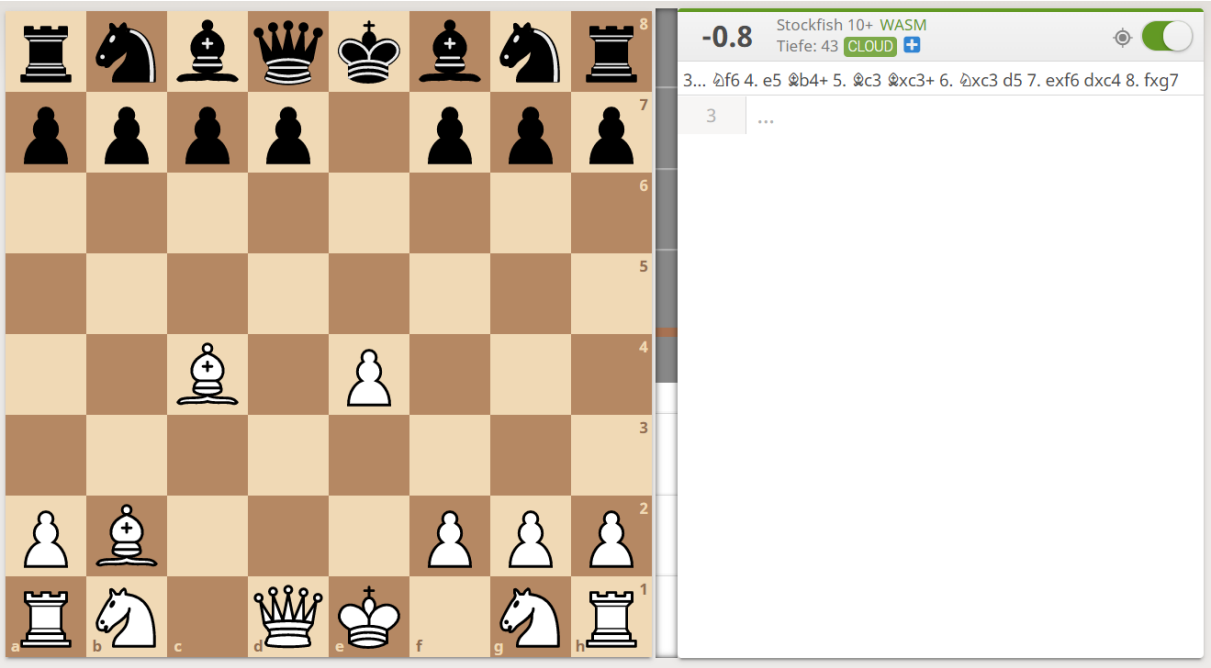

Abbildung 24: Nordische Gambit

Ein Beispiel hierfür ist das auf Abbildung 24 zu sehende „Nordische Gambit“. Beim „Nordischen Gambit“ opfert Weiß zwei Bauer und kriegt nur einen Bauer von Schwarz. Der Computer zeigt durch den Bewertungswert -0,8 ganz klar, dass Schwarz im Vorteil liegt. Tatsächlich ist Weiß im Vorteil: Weiß hat zwar zwei Bauern weniger als Schwarz, hat aber zwei entwickelte Läufer, die den Königsflügel (die Felder f7 und g7) bedrohen und die Damenlinie (alle Felder mit d) steht für Weiß offen, während Schwarz keine einzige entwickelte Figur auf dem Feld hat.

Im nächsten Unterkapitel werden weitere Beispiele aufgezählt, aber auch Beispiele für das Versagen von Schachcomputern

4.6 Beispiele von Schachzügen und Versagen vom Schachcomputer

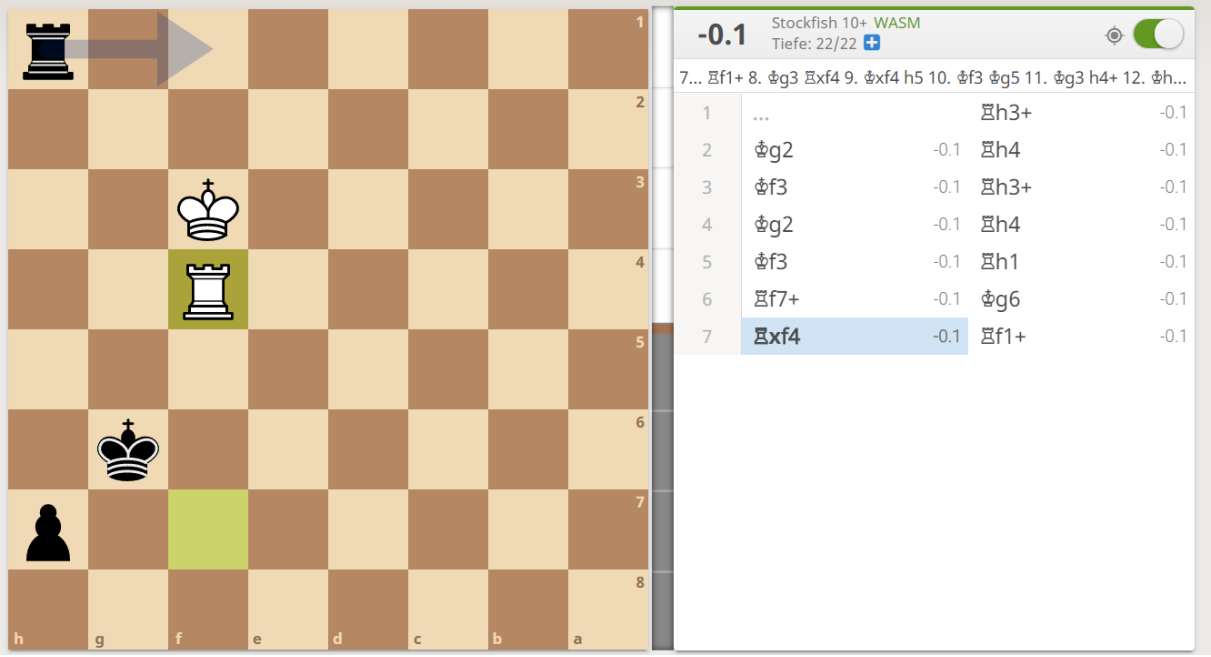

Abbildung 25: Beispiel zu einem Endspiel

In Abbildung 25 ist ein Beispiel zu einem Endspiel zu sehen. Schwarz ist am Zug, Schwarz steht um 0,1 Bauern besser als Weiß (Bewertungsfunktion gibt den Wert -0,1 zurück). Das Programm hat den Zug Turm auf f1 berechnet, da einerseits der weiße König in Schach gesetzt wird und der schwarze Turm den weißen Turm schlagen kann, nachdem sich der weiße König aus dem Schach entfernt hat. Daraufhin kann der weiße König den schwarzen Turm schlagen und Schwarz steht dennoch mit einem Bauer mehr auf dem Feld während Weiß nur noch den König hat.

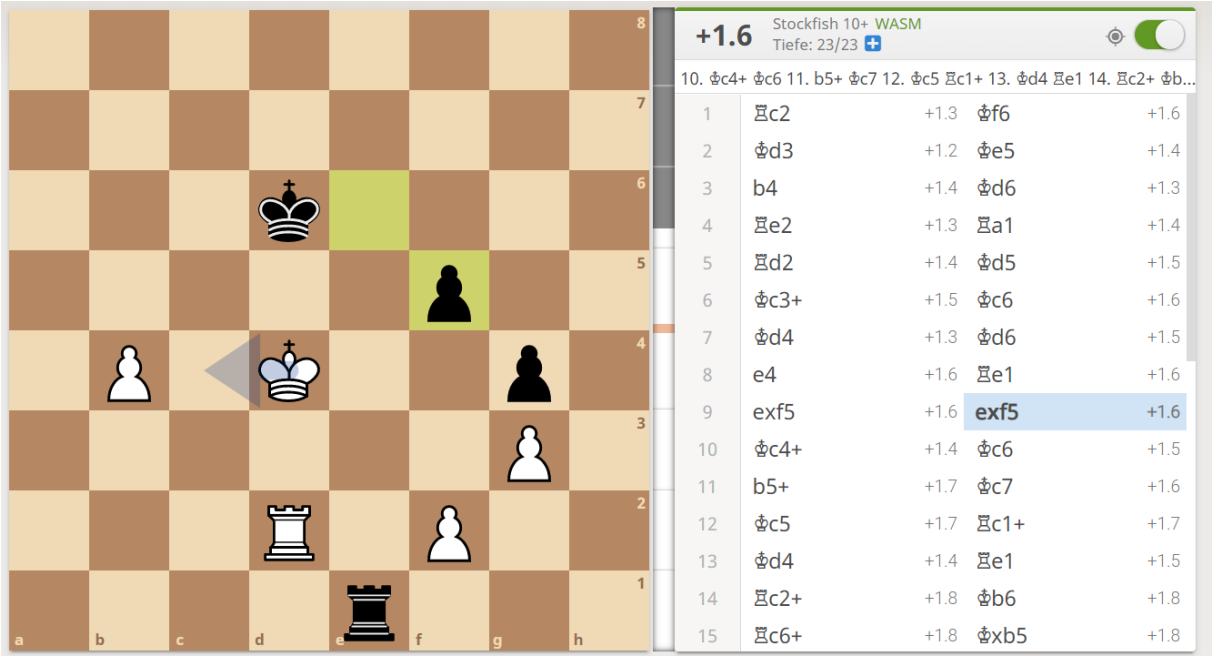

Abbildung 26: Beispiel 2 zu einem Endspiel

In Abbildung 26 ist ein weiteres Beispiel zu einem Endspiel zu sehen. Weiß ist am Zug und steht um 1,6 Bauern besser als Schwarz. Das Programm hat den Zug Kc4 berechnet, da durch diesen Zug einerseits der König näher zum b4 Bauer gebracht wird, welcher dann versuchen kann auf die andere Seite des Brettes zu kommen, andererseits wird der schwarze König ins Schach gestellt, da die Linie zwischen dem weißen Turm auf d2 und dem schwarzen König auf d6 frei wird.

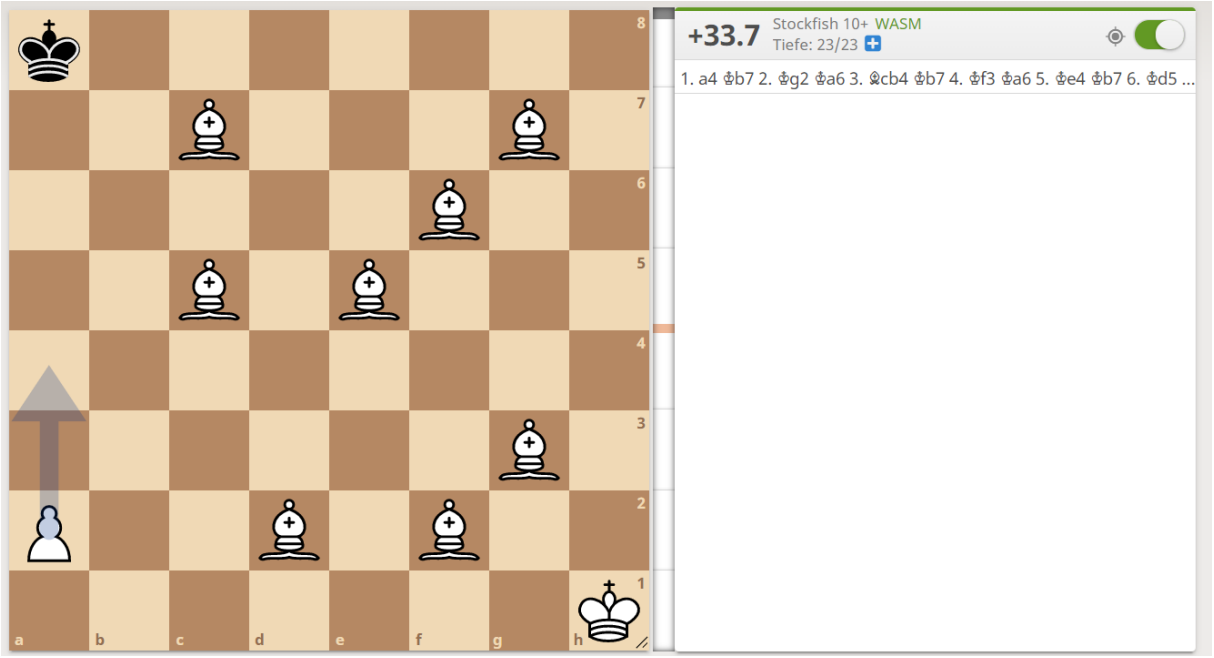

Abbildung 27: Beispiel für das Versagen des Schachcomputers

In Abbildung 27 sehen wir ein weiteres Endspiel, jedoch versagt hier das Programm. Hier steht Weiß um 33,7 Bauern besser da als Schwarz. An der Stelle von Schwarz würde das Programm aufgeben und auf der Stelle von Weiß würde das Programm ein Remis ablehnen. Diese Partie ist jedoch Remis. Weiß kann mit seinen vielen Läufern den schwarzen König Matt setzen, da alle Läufer auf schwarzen Feldern sind. Sollte Weiß mit seinem König bzw. mit seinem Bauern dem schwarzen König zu nahe kommen, droht ein Patt.

5 Fazit

Zusammenfassend kann man sagen, dass Schachprogramme die Schachwelt revolutioniert haben. Durch Schachprogramme konnten sehr viele neue Stellungen und Spielstile entwickelt werden. Mittlerweile ist auf Schachprogrammen gar nicht mehr verzichtbar, da sie einen so bedeutenden Wert in der Schachszene erlangt haben, auch wenn der Weg dorthin, aufgrund von mangelnder Hardware und der Komplexität bzw. der Vielfalt der Stellungen im Schach, schwer war.

In dieser Arbeit wurden die Grundkomponenten von Schachprogrammen gezeigt, die die Bewertungsfunktion verwenden. Es wurde auch gezeigt, dass Schachprogramme eben nicht jede Schachstellung und jede Situation richtig analysieren und bewerten können. Hinzufügend muss gesagt werden, dass die momentanen Schachprogramme, die auf der Bewertungsfunktion basieren, immer mehr durch Schachprogramme, die auf künstlicher Intelligenz basieren, wie z.B. AlphaZero, abgelöst werden.

Der Arbeitsprozess dieser Arbeit war im Großen und Ganzen sehr angenehm. Es gab kaum Schwierigkeiten beim Recherchieren und beim Ausarbeiten. Es fiel mir anfangs schwierig die Beschreibung zum Pseudocode (Kapitel 4.4) zu schreiben, aber diese Herausforderung lies sich relativ schnell überwältigen. Vor dem Ausarbeiten dieser vorwissenschaftlichen Arbeit hätte ich nie vermutet, dass Schachprogramme so komplex sein können.

Literaturverzeichnis

Alt, R., Deventer, K., Klünser, J., Strobl, T., & Wiedmann, T. (2018). *Die FIDE - Schachregeln.* Von https://www.schachbund.de/srk-downloads.html?file=files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf&cid=50517 abgerufen [letzter Zugriff: 02.02.2020]

Bauermeister, K. *SCHACHCOMPUTER GESCHICHTE*. Von http://www.schachcomputer.at/gesch1.htm , http://www.schachcomputer.at/gesch2.htm , http://www.schachcomputer.at/gesch3.htm , http://www.schachcomputer.at/gesch4.htm , http://www.schachcomputer.at/gesch5.htm , http://www.schachcomputer.at/gesch14.htm abgerufen [letzter Zugriff: 01.02.2020]

Bauermeister, K. *SCHACHCOMPUTER GESCHICHTE*. Von http://www.schachcomputer.at/gesch16.htm , http://www.schachcomputer.at/gesch18.htm , http://www.schachcomputer.at/gesch21.htm abgerufen [letzter Zugriff: 01.02.2020]

Isenberg, G. *Chess Programming Wiki*. Von https://www.chessprogramming.org/Fritz , https://www.chessprogramming.org/Stockfish , https://www.chessprogramming.org/Houdini , https://www.chessprogramming.org/Horizon_Effect abgerufen [letzter Zugriff: 19.02.2020]

Reimers, T., Tietjen, T., & Wehr, C. (2006/2007). *Schach-Computer Algorithmen und Architekturen.* Von http://www.weblearn.hs-bremen.de/risse/RST/WS06/Schachcomputer.pdf abgerufen [letzter Zugriff: 20.01.2020]

Abbildungsverzeichnis

Begleitprotokoll

Name des Schülers/der Schülerin: Mahmoud El-Shennawi

Thema der Arbeit: Die Funktionsweise von Schachcomputern

Name der Betreuungsperson: Mag. Hans-Jürgen Koller

Datum	**Vorgangsweise, ausgeführte Arbeiten, aufgesuchte Bibliotheken, …**	**Besprechungen mit der betreuenden Lehrperson, Fortschritte, offene Fragen, Probleme, nächste Schritte**
Oktober 2018	Auswahl des Themas	
November 2018	Erstellung des Konzepts	
14. Jänner 2019	Feedback des Konzeptes durch Betreuer	Mag. Hans-Jürgen Koller via E-Mail
26. Februar 2019	Genehmigung des Themas vom Landesschulinspektor / von der Landesschulinspektorin	
Juni 2019 – Juli 2019	Schreibphase der ersten Kapitel	
9. Juli 2019	Erste Rückmeldung	Mag. Hans-Jürgen Koller via E-Mail
Juli 2019 – August 2019	Schreibphase weiterer Kapitel	
18. September 2019	Treffen zur Besprechung der VWA und für Rückmeldungen	Mag. Hans-Jürgen Koller
September 2019 – Jänner 2020	Schreibphase	
28. Jänner 2020	Treffen zur Besprechung der VWA und für Rückmeldungen	Mag. Hans-Jürgen Koller
13. Februar 2020	Fertigstellung der VWA	
18. Februar 2020	Letzte Rückmeldung	Mag. Hans-Jürgen Koller via E-Mail
	Abgabe der BW	

Diese Vorwissenschaftliche Arbeit umfasst 34.585 Zeichen exklusive Vorwort, Inhalts-, Literatur- und Abbildungsverzeichnis.

Ort, Datum	Unterschrift:
_____	_____