Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Bachelorarbeit Nr. 190

Monte Carlo Tree Search Algorithmen für das Brettspiel "Scotland Yard"

Manuel Gräber

Studiengang:	Informatik
Prüfer/in:	Prof. Dr.-Ing. Andrés Bruhn
Betreuer/in:	Prof. Dr.-Ing. Andrés Bruhn
Beginn am:	1. Dezember 2014
Beendet am:	13. Mai 2015
CR-Nummer:	I.2.1, I.2.8, G.3

ZUSAMMENFASSUNG

Monte Carlo Algorithmen haben in letzter Zeit immer mehr an Bedeutung gewonnen. Vor allem das Einsetzen von Monte Carlo Algorithmen zum Erstellen und randomisierten Absuchen eines Suchbaums hat neue Wege im Bereich der Künstlichen Intelligenz geschaffen. In der vorangegangenen Arbeit von Minorics [1] wurden für das Brettspiel Scotland Yard KIs für die Steuerung von Mister X entwickelt. Diese KI-Algorithmen haben jedoch keine Planung der Züge im klassischen Sinn vorgenommen. Eine Steuerung der Detektive wurde zudem nicht implementiert. Diese Arbeit erweitert die Ergebnisse der vorangegangenen Arbeit durch das Umsetzen von KIs zur Steuerung der Detektive und durch das Einsetzen von Monte-Carlo-Tree-Search-Algorithmen für die Zugplanung. Neben der Implementierung der einzelnen KIs steht auch deren ausführliche Evaluation im Mittelpunkt der Arbeit. Diese wurde anhand von umfassenden Testspielen durchgeführt, bei den jeweils verschiedene KIs für Mister X und die Detektive gegeneinander evaluiert werden.

INHALTSVERZEICHNIS

1. EINLEITUNG

1.1 MOTIVATION

"Scotland Yard", das 1983 von Ravensburger Spiele veröffentlicht wurde, ist ein beliebtes Brettspiel. Es spielen zwei Teams gegeneinander: Zum einen die Detektive, die versuchen müssen, den Verbrecher Mister X zu fangen, und zum anderen Mister X, dessen einziges Ziel es ist, den Detektiven zu entkommen. Das Spiel wird auf einem Stadtplan von London gespielt. Um sich fortzubewegen, müssen die Spieler auf die vorhandenen Verkehrsmittel (Bus, Taxi und U-Bahn, Mister X steht zusätzlich noch das Boot zur Verfügung) zurückgreifen. Das Problem der Detektive ist, dass sie nicht wissen, wo genau sich Mister X befindet. Allerdings muss dieser sich in regelmäßigen Abständen den Detektiven zeigen. Gefangen wird Mister X, indem ein Detektiv auf das Feld zieht, auf dem sich Mister X befindet.
Da der Spielplan einen hohen Verzweigungsgrad aufweist, und da die Tickets, über die die Spieler verfügen, begrenzt sind, stellt die Entwicklung von Algorithmen sowohl für die Detektive als auch für Mister X eine interessante Aufgabe in Bezug auf die Entwicklung von künstlichen Intelligenzen dar. In der vorangegangenen Arbeit von Minorics (Studienarbeit 2013) [1] wurden ausschließlich worst-case-basierte-Algorithmen für Mister X entworfen: Einen einfachen distanzbasierten Algorithmus und einen Algorithmus, der nur sichere Pfade geht. Es wurde allerdings kein Algorithmus für die Detektive entworfen. Ebenso wenig findet eine Planung statt, die üblicherweise durch die Analyse eines Suchbaums realisiert wird. Da es aufgrund des hohen Verzweigungsgrads nicht möglich ist, einen vollständigen Suchbaum zu erstellen, legen andere Arbeiten (Nijssen and Winands, IEEE T-CIAIG 2012 [2]) nahe, hierfür stochastische Verfahren, wie Monte-Carlo-Tree-Search-Algorithmen zu verwenden. Diese bauen nach einem bestimmten Verfahren einen Teilsuchbaum auf und bewerten die einzelnen Züge indem das Spiel von dem resultierenden Punkt aus zu Ende simuliert wird. Durch eine Balance zwischen dem Suchen neuer Zugmöglichkeiten und dem genaueren Untersuchen eines bereits vielversprechenden Zweiges lassen sich vielversprechendere Züge finden.

1.2 AUFGABENSTELLUNG

Ziel dieser Arbeit ist es für die beiden Parteien MiniMax und Monte Carlo Tree Search (MCTS) basierte Algorithmen zu entwickeln. Um dies zu realisieren, müssen zunächst nichtplanende Algorithmen entworfen werden, die später von den MCTS Algorithmen verwendet werden können, um die Spiele zu simulieren (Playout). Der einfachste

dieser Algorithmen ist das zufällige Auswählen eines möglichen Zuges. Um ein möglichst realistisches Playout zu erhalten, soll im Rahmen dieser Arbeit auch ein Greedy-Algorithmus für die Detektive entwickelt werden, der anhand einer Wahrscheinlichkeitsverteilung (Belief State) versuchen soll, ihren Abstand zu Mister X zu minimieren. Durch den Greedy-Detektiv-Algorithmus ist es dann möglich, die Spielstärke der MCTS-Algorithmen mit den von Minorics (Studienarbeit 2013) [1] erstellten Algorithmen zu vergleichen. Anschließend soll noch die Spielstärke der MCTS-Detektive mit den Greedy-Detektiven verglichen werden.

1.3 VERWANDTE ARBEITEN

Seit der Entwicklung des Monte Carlo Tree Search Algorithmus wurden dazu viele Arbeiten veröffentlicht. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search [3] stellte als erstes diesen Algorithmus vor, um eine künstliche Intelligenz für Go zu erstellen. Erst durch MCTS nahm die zu diesem Zeitpunkt noch schwache künstliche Intelligenz für Go an Spielstärke deutlich zu.
A Survey of Monte Carlo Tree Search Methods [4] beschreibt die Grundlagen des Algorithmus und gibt einen Überblick über viele mögliche Varianten und Modifikationen, um den Algorithmus für bestimmte Aufgaben anzupassen. Die vorangegangene Arbeit "Digital Escape – Ein KI-basierter Algorithmus zur Steuerung von Mr. X im Brettspiel Scotland Yard" [1] stellt künstliche Intelligenzen für Mister X vor, die in dieser Arbeit als Grundlage dienen. Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard [2] hat MCTS schon eingesetzt um KIs für das Spiel zu verbessern und hat damit schon einige Erkenntnisse zu diesem Thema erbracht.

1.4 GLIEDERUNG

In Kapitel 2 werden die Spielregeln des Spiels erklärt. In Kapitel 3 werden die grundlegenden Techniken erläutert, die dann in Kapitel 4 evaluiert werden. Im folgenden Kapitel 5 werden die Grundlagen zur Monte Carlo Tree Search beschrieben. Die Ergebnisse der Monte-Carlo-Tree-Search-KIs von Mister X werden dann in Kapitel 6 evaluiert. Im 7. Kapitel werden anschließend noch die Ergebnisse der Detekiv-Monte-Carlo-Tree-Search betrachtet. Die Arbeit endet mit einer Zusammenfassung in Kapitel 8, das zudem noch Vorschläge macht, wie die KIs noch weiter verbessert werden könnten.

2. SCOTLAND YARD

Scotland Yard ist ein Hide-and-Seek Brettspiel für drei bis sechs Spieler. Wenn nur drei Spieler am Spiel teilnehmen, spielen die beiden Detektivspieler je zwei Spielfiguren. Es gibt zwei Fraktionen, die Detektive und Mister X. Während Mister X versuchen muss, vor den Detektiven zu entkommen, müssen diese zusammenarbeiten, den Aufenthaltsort von Mister X erraten und ihn fangen. Sollte es einem Detektiv gelingen, Mister X zu fassen, gewinnen alle Detektive zusammen. Der Sieg der Detektive ist gleichzeitig die Niederlage von Mister X und umgekehrt. [5]

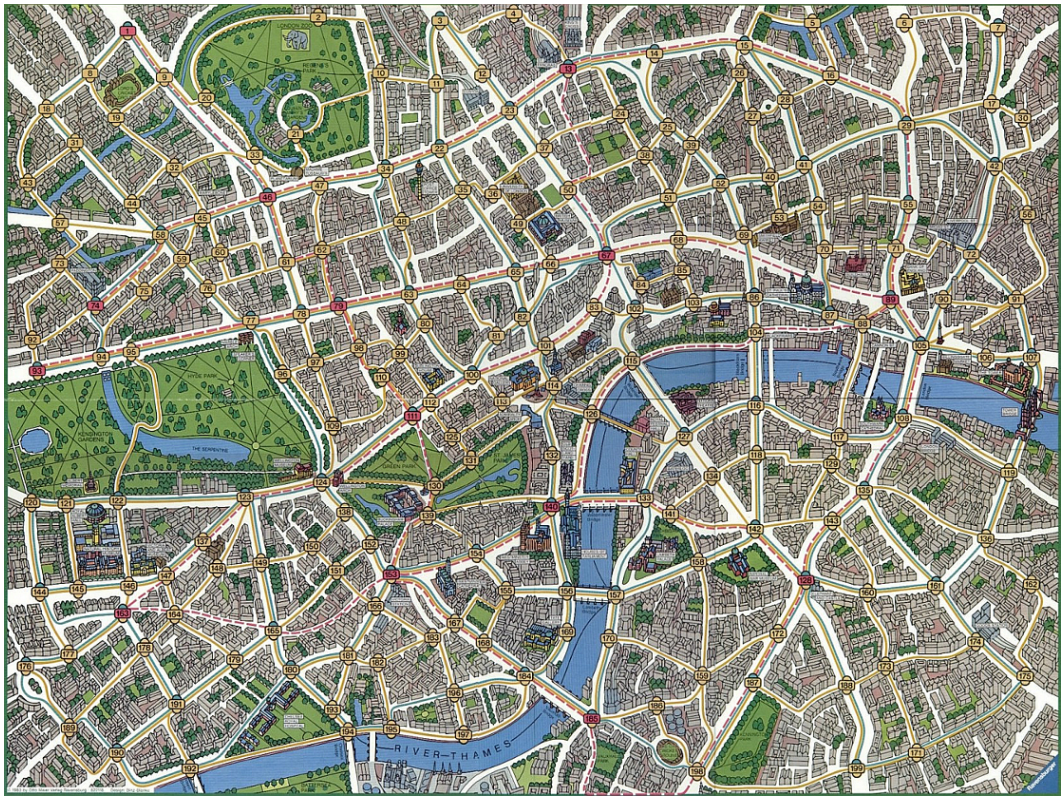

Abbildung 2.1: Spielplan von Scotland Yard. Die Felder sind von 1 bis 199 nummeriert. Die verschiedenen Linien zeigen welchen Verkehrsmittel zwischen den Feldern verwendet werden können. [5]

2.1 SPIELREGELN

2.1.1 *Beginn des Spiels*

Je nach Spieleranzahl treten drei bis fünf Detektive gegen Mister X an. Jeder Detektiv erhält zu Beginn des Spiels 10 Taxi-Tickets, 8 Bus-Tickets und 4 U-Bahn Tickets. Mister X erhält 4 Taxi-Tickets, 3 Bus-Tickets, 3 U-Bahn-Tickets und 2 Doppelzugkarten, außerdem für jeden, am Spiel teilnehmenden Detektiv, ein Black-Ticket. Die Startpositionen werden zufällig vergeben und die übriggebliebenen geheim gehalten. Die 18 möglichen Startpositionen sind die Felder 13, 26, 29, 34, 50, 53, 91, 94, 103, 112, 117, 132, 138, 141, 155, 174, 197, 198.

2.1.2 *Zugregeln*

Mister X zieht als erster. Anschließend ziehen die Detektive in fester Reihenfolge. Um einen Zug von Feld A nach Feld B auszuführen, muss es eine direkte Verbindung zwischen den Feldern geben, das Feld darf nicht von einem Detektiv belegt sein und der Spieler muss das entsprechende Ticket besitzen. Direkte Verbindung bedeutet hier, dass es zwischen den beiden Feldern keine Zwischenhaltestelle des verwendeten Verkehrsmittels geben darf. Nicht alle Verkehrsmittel sind auf allen Feldern verfügbar. Mister X hat zusätzlich zu Taxi, Bus und U-Bahn noch die Möglichkeit, mit dem Boot zu fahren.

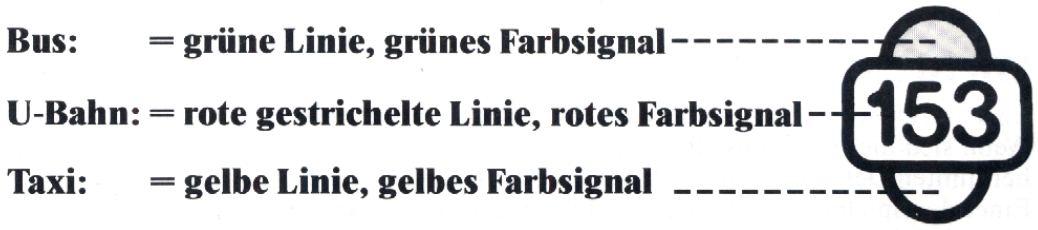

Abbildung 2.2: Ist der obere Halbkreis grün, handelt es sich um eine Bushaltestelle. Ist die Mitte rot, handelt es sich um eine U-Bahnhaltestelle. Das Taxi steht überall zur Verfügung. [5]

Um als Detektiv einen Zug auszuführen, wird das verwendete Ticket an Mister X abgegeben und die Spielfigur auf das entsprechende Feld gezogen. Zieht Mister X, notiert er das Zielfeld verdeckt und legt das verwendete Ticket offen ab. Ein Black-Ticket kann für jedes Verkehrsmittel benutzt werden.
Mister X hat außerdem die Möglichkeit, einen Doppelzug zu machen. Hierzu legt er eine seiner Doppelzukarten ab und darf anschließend erneut ziehen.

2.1.3 *Mister X zeigt sich*

Mister X muss sich nach den Zügen 3, 8, 13, 18 und am Ende des Spiels zeigen. Die Detektive wissen dann sicher, auf welchem Feld er sich gerade befindet. Für die Zugzahl wird von der Anzahl der von Mister X ausgeführten Zügen ausgegangen. Ein Doppelzug zählt als zwei Züge.

2.1.4 *Ende des Spiels*

Das Spiel ist beendet, sobald Mister X gefasst wurde oder alle Detektive nicht mehr ziehen können. Im ersten Fall haben die Detektive gewonnen, andernfalls gewinnt Mister X.

2.2 INTERPRETATION DER REGELN

Von den Regeln werden einige Spezialfälle nicht abgedeckt. Diese Fälle treten sehr selten ein, müssen aber trotzdem definiert sein.

2.2.1 *Mister X kann nicht ziehen*

Sollte Mister X keinen möglichen Zug haben, verliert er das Spiel. Das kann entweder eintreten, wenn er kein passendes Ticket übrig hat, was sehr selten eintritt, da Mister X pro Zug mindestens drei Tickets von den Detektiven bekommt, oder von den Detektiven eingekreist wurde und deshalb nicht mehr ziehen kann.

2.2.2 *Ein Detektiv kann nicht ziehen*

Wenn ein Detektiv von den anderen Detektiven blockiert wird und deshalb nicht ziehen kann, wird sein Zug übersprungen und er kommt in seiner nächsten Runde wieder normal an die Reihe. So ist es möglich, dass das Spiel länger als 24 Züge dauert. (Jeder Detektiv hat zu Beginn je 22 Tickets, und Mister X kann 2 Doppelzüge machen). Ein Spiel, das Mister X gewinnt und in dem er beide Doppelzüge einsetzt, dauert in der Regel 25 Züge, da das Spiel erst beendet wird, wenn alle Detektive keinen Zug gemacht haben, da sie nicht mehr ziehen können.

3. GRUNDLEGENDE KIS

In diesem Kapitel werden die grundlegenden KIs von Mister X und den Detektiven beschrieben. Die hier beschriebenen Methoden werten nur den momentanen Spielstand aus und entscheiden anhand dieser Informationen welchen Zug sie ausführen.

3.1 KI VON MISTER X

Die distanzbasierte und die Safepath-KI von Mister X wurden in der vorangegangenen Arbeit von Minorics erstellt [1]. Auf sie wird hier nicht tiefer eingegangen.

3.1.1 *Random-KI*

Die Random KI von Mister X wählt aus allen möglichen Zügen zufällig einen aus, wobei die Wahrscheinlichkeiten gleichverteilt sind. Er macht keine Doppelzüge und setzt Blacktickets nur ein, um mit dem Boot zu fahren.

3.1.2 *Distanzbasierte-KI*

Mister X versucht, die Distanz zu dem Detektiv, der ihm an nächsten ist, zu maximieren. Er macht dabei keine Doppelzüge. Hierfür berechnet er zu jedem Feld, das er mit einem Zug erreichen kann, aus, wie viele Züge der nächste Detektiv benötigt, um auf dieses Feld zu kommen. Dann zieht er auf das Feld mit dem höchsten Wert. [1]

3.1.3 *Safepath-KI*

Hier versucht Mister X den Weg zu gehen, auf dem er für die meisten Züge sicher ist. Wenn er keinen Möglichkeit findet, in einem Zug in eine sichere Distanz zu den Detektiven zu kommen, setzt er einen Doppelzug ein. Blacktickets verwendet er nur, um mit dem Boot zu fahren. Für diesen Algorithmus wird für jedes Feld berechnet, wie viele Züge vor den Detektiven Mister X dort ankommen kann. Gewählt wird der Weg, der den kürzesten Weg zu dem Feld mit dem höchsten Abstand zu den Detektiven beschreibt. Anschließend wird der erste Zug des Weges gemacht. Der Weg und das Zielfeld werden für jeden Zug neu berechnet. [1]

3.2 KI DER DETEKTIVE

Auch wenn die Detektive als Team auftreten, machen sie ihre Züge individuell. Grund hierfür ist, dass es so einfacher und schneller ist, aus der sehr beschränkten Anzahl an Zügen den Besten zu berechnen. Wenn jeder der Detektive 5 mögliche Züge hat, und mit 5 Detektiven gespielt wird, ergäben sich sonst 3125 mögliche Züge, die alle evaluiert werden müssen. Außerdem ist es so einfacher, einzelne Detektive durch menschliche Spieler zu ersetzen.

3.2.1 *Random*

Jeder der Detektive wählt aus seinen möglichen Zügen zufällig einen aus. Wie bei Mister X sind ist die Wahrscheinlichkeiten für jeden Zug gleich.

3.2.2 *Greedy*

Bei der Greedy-KI verfügen die Detektive über einen Speicher für einen Belief-State in dem sie alle möglichen Standorte von Mister X abspeichern. Mister X zeigt sich während des Spiels alle 5 Züge; sobald er das tut werden alle Felder, mit Ausnahme des Feldes auf dem Mister X steht, auf "false" bzw. 0,0 gesetzt. Sobald Mister X zieht, werden, von jedem möglichen Standort ausgehend, alle möglichen neuen Standorte berechnet. Die Greedy-KI versucht nun, den Abstand der Detektive zu allen möglichen Standorten von Mister X zu minimieren. Sobald ein Detektiv auf ein möglichen Standort von Mister X zieht, wird dieser entweder gefasst und das Spiel ist vorbei oder das Feld wird aus dem Belief-State entfernt. Zu Beginn wird der Belief-State mit allen möglichen Startpositionen, auf denen sich kein Detektiv befindet, initialisiert.

3.2.2.1 *Bool-Belief*

Im Bool-Belief-State werden alle Felder, auf denen sich Mister X befinden kann, mit einem Wahrheitswert versehen. Jedes Feld, auf dem sich Mister X befinden kann wird gleich bewertet. Sobald Mister X einen Zug macht werden alle Felder, auf die Mister X theoretisch ziehen könnte, auf "wahr" gesetzt.

3.2.2.2 *Float-Belief*

Hier wird anstatt einem Wahrheitswert eine Gleitkommazahl verwendet. 1,0 bedeutet Mister X befindet sich sicher auf diesem Feld, 0,0 markiert die Unmöglichkeit. Sobald Mister X zieht, wird von jedem Feld mit einem Belief Wert von > 0, dieser genommen und durch die Anzahl der möglichen Verbindungen, die von diesem Feld ausgehen und vom verwendeten Ticket verwendet werden können, geteilt und auf das entsprechende Zielfeld einem mit 0,0 initialisierten temporären Array addiert. Sollte eins oder

mehrere Felder keinen möglichen Zug mit dem verwendeten Ticket haben, werden die einzelnen Werte am Ende durch die Summe aller Werte geteilt. Anschließend wird der temporäre Array in den ursprünglichen kopiert. Somit entspricht der berechnete Wert für jedes Feld der tatsächlichen Wahrscheinlichkeit, falls Mister X zufällig zieht.

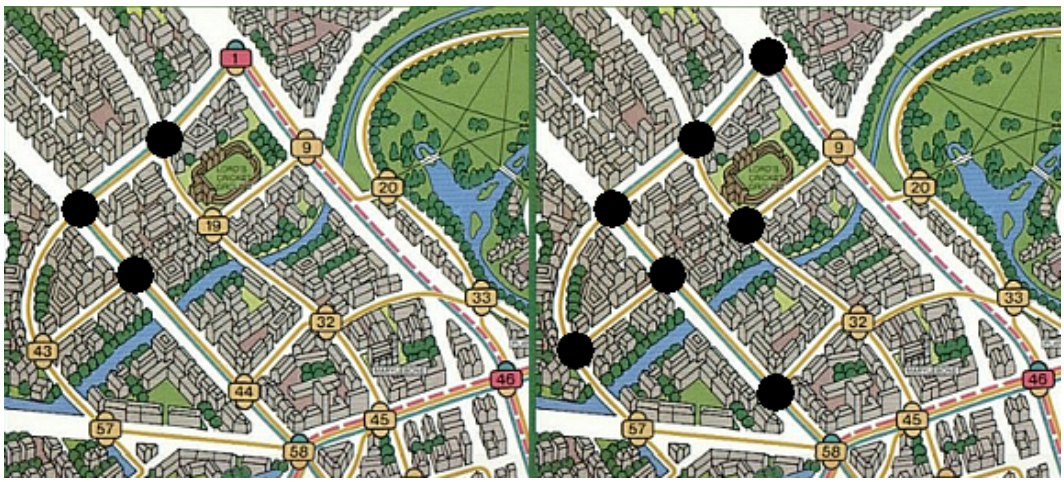

Abbildung 3.1: Visuelle Darstellung des Bool-Belief-State: Mister X könnte auf den Feldern 8, 18 und 31 stehen. Er zieht mit einem Taxiticket, jetzt sind seine möglichen Felder 1, 8, 18, 19, 31, 43 und 44.

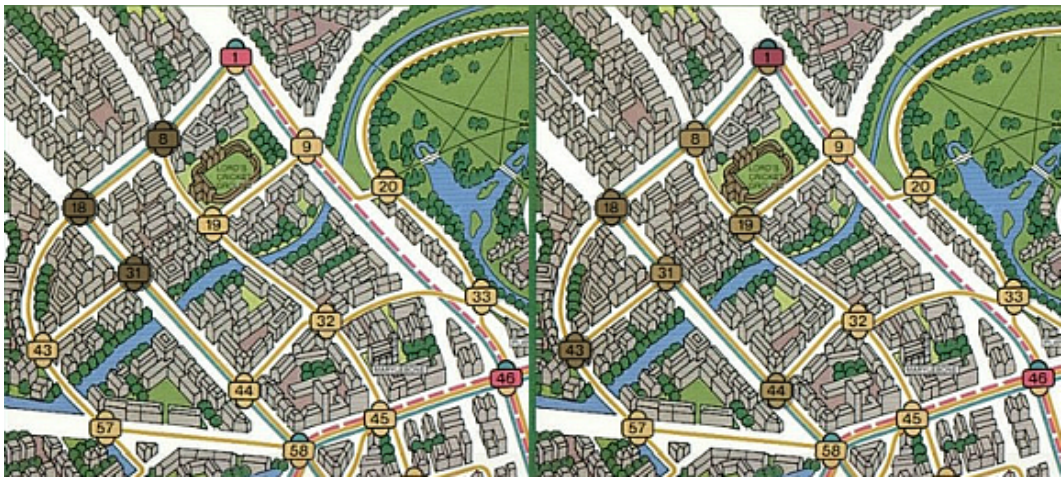

Abbildung 3.2: Visuelle Darstellung des Float-Belief-State: Mister X könnte auf den Feldern 8, 18 und 31 stehen, deren Wert ist jeweils 0,33. Er zieht mit einem Taxiticket. Jedes der Felder hat 3 Taxi-Verbindungen, daher wird der Wert für jedes Feld durch 3 geteilt und auf die Zielfelder verteilt. Somit sind die Werte der möglichen Felder 1 (0,11), 8 (0,11), 18 (0,22), 19 (0,11), 31 (0,11), 43 (0,22) und 44 (0,11).

3.2.2.3 *Zugwahlkriterien*

Der Greedy-Algorithmus versucht den Abstand zu allen möglichen Positionen zu minimieren. Hierfür werden mehrere Kriterien verwendet. Nur wenn das primäre Kriterium bei zwei Zügen gleich ist, wird das Sekundäre betrachtet. Wenn auch dieses gleich ist, das Tertiäre und so weiter.
Die betrachteten Kriterien sind:

A. Der minimale Abstand der Detektivgruppe zu allen möglichen Positionen von Mister X (wobei diese im Float-Belief gewichtet werden)

B. Der minimale Abstand des ziehenden Detektivs zu diesen Positionen

C. Die maximale Anzahl der Verbindungen des Zielfelds

D. Die maximale Anzahl der vorhandenen Tickets des verwendeten Verkehrsmittels

E. Ein Zufallswert

Es werden zwei Zugauswahlkriterien verwendet, zum einen die Kriterien A, B, C, E in der aufgeführten Reihenfolge (Greedy 1), zum anderen alle Kriterien in der angegebenen Reihenfolge (Greedy 2). Daraus ergeben sich für die Detektive insgesamt vier Greedy-KIs, zwei mit Bool-Belief und zwei mit Float-Belief.

3.2.3 *Zusammenfassung*

In dieser Arbeit werden drei KIs für Mister X und fünf für die Detektive verwendet.

Random	Mister X macht aus allen möglichen Zügen einen Zufälligen
Distanz	Mister X maximiert die Distanz zu den Detektiven
Savepath	Mister X geht den Weg, auf dem er möglichst lange sicher ist

Tabelle 3.1: KIs von Mister X, Distanz und Savepath sind Teil der vorangegangenen Arbeit

Random	Detektiv macht aus allen möglichen Zügen einen zufälligen
Greedy Bool 1	Detektiv verwendet Bool-Belief und Zugwahlkriterium 1
Greedy Bool 2	Detektiv verwendet Bool-Belief und Zugwahlkriterium 2
Greedy Float 1	Detektiv verwendet Float-Belief und Zugwahlkriterium 1
Greedy Float 2	Detektiv verwendet Float-Belief und Zugwahlkriterium 1

Tabelle 3.2: KIs der Detektive

4. EVALUATION UND BEWERTUNG DER STRATEGIEN

In der vorangegangenen Arbeit wurde keine KI für die Detektive implementiert, und somit war es nicht möglich, die KIs von Mister X einem Test gegen eine andere KI zu unterziehen. Deshalb hat ein menschlicher Spieler die Aufgabe der Detektiv-KI übernommen. Dabei war es nicht möglich eine große Anzahl an Spielen durchzuführen. Zudem war die Spielstärke des menschlichen Spielers oft nicht konstant, da die Konzentration mit der Zeit nachlässt. Da im Laufe dieser Arbeit KIs für die Detektive erstellt wurden, ist es nun möglich, die Stärken der einzelnen KIs gegeneinander zu testen.

4.1 TESTKRITERIEN

Für die Tests stehen drei Mister X-KIs und fünf Detektiv-KIs zur Verfügung. Daraus ergeben sich 15 unterschiedliche Kombinationen. Für die Tests werden insgesamt eine Million Spiele gespielt. Jede der verschiedenen Kombinationen hat hier die selben Ausgangsstellungen, damit die Ergebnisse besser zu vergleichen sind. Betrachtet werden hier die Gewinnrate der einzelnen KIs (wobei ein Sieg für Mister X ein gewonnenes Spiel darstellt), die Zeit, die die einzelnen Spiele dauern, und die durchschnittliche Anzahl der Züge, die die Spiele benötigen, wobei die Anzahl für gewonnene Spiele und für verlorene Spiele auch einzeln betrachtet werden. Ein Spiel mit n Spielern bedeutet, dass n-1 Detektive spielen.

4.2 GEWINNRATE

Die Tabellen 4.1 - 4.3 zeigen die Gewinnrate der einzelnen Paarungen der KIs, wobei die ersten drei Spalten die absolute Siegzahl von Mister X angeben und die anderen drei die prozentuale Siegzahl. In den Abblidungen 4.1 - 4.3 werden die Ergebnisse noch grafisch visualisiert. Mehrfaches ausführen des selben Programms auf unterschiedlichen Computern ergibt stets ein sehr ähnliches Ergebnis, wobei die Abweichung von diesen Werten bei allen Ausführungen kleiner als 0,1% sind.

6 Spieler	Random	Distanz	Savepath	Random	Distanz	Savepath
Random	698853	995173	999957	69,89%	99,52%	100,00%
Greedy Bool 1	464	140629	440579	0,05%	14,06%	44,06%
Greedy Bool 2	522	138400	426676	0,05%	13,84%	42,67%
Greedy Float 1	508	148374	487215	0,05%	14,84%	48,72%
Greedy Float 2	484	145579	469593	0,05%	14,56%	46,96%

Tabelle 4.1: 1 Million Spiele mit 6 Spielern, Siege von Mister X und prozentuale Gewinnrate

5 Spieler	Random	Distanz	Savepath	Random	Distanz	Savepath
Random	750596	996949	999952	75,06%	99,69%	100,00%
Greedy Bool 1	1773	255214	674146	0,18%	25,52%	67,41%
Greedy Bool 2	1821	251501	659742	0,18%	25,15%	65,97%
Greedy Float 1	1617	266241	707352	0,16%	26,62%	70,74%
Greedy Float 2	1692	264099	690105	0,17%	26,41%	69,01%

Tabelle 4.2: 1 Million Spiele mit 5 Spielern, Siege von Mister X und prozentuale Gewinnrate

4 Spieler	Random	Distanz	Savepath	Random	Distanz	Savepath
Random	806911	998368	999926	80,69%	99,84%	99,99%
Greedy Bool 1	6992	427906	884880	0,70%	42,79%	88,49%
Greedy Bool 2	7375	421034	886725	0,74%	42,10%	88,67%
Greedy Float 1	6527	442088	895967	0,65%	44,21%	89,60%
Greedy Float 2	6659	442202	888263	0,67%	44,22%	88,83%

Tabelle 4.3: 1 Million Spiele mit 4 Spielern, Siege von Mister X und prozentuale Gewinnrate

Die Ergebnisse zeigen deutlich, dass - zumindest mit den hier verwendeten KIs - die Siegchancen von Mister X mit abnehmender Spielerzahl deutlich zunehmen. Die Tatsache, dass beide Random-KIs nie eine Siegchance von über einem Prozent haben, zeigt auch, wie wichtig eine gute KI für den Sieg ist.
Aus den Ergebnissen lässt sich auch sehr leicht ablesen, dass "Savepath" die stärkste KI von Mister X ist. Bei den KIs der Detektive lässt sich das nicht so eindeutig sagen. Beim Spiel mit 6 Spielern ist die stärkste KI der Detektive die "Greedy Bool 2-KI", während beim Spiel mit 4 Spielern gegen die Savepath-KI die "Greedy Bool 1-KI" minimal besser abschneidet. Die "Greedy Float 1/2-KIs" sind im Spiel gegen die Distanz-KI und die Savepath-KI der "Greedy Bool 1/2-KI" unterlegen. Nur bei 4 Spielern gegen "Random Mister X" schneidet sie etwas besser ab als "Greedy Bool 1/2". Es lässt sich

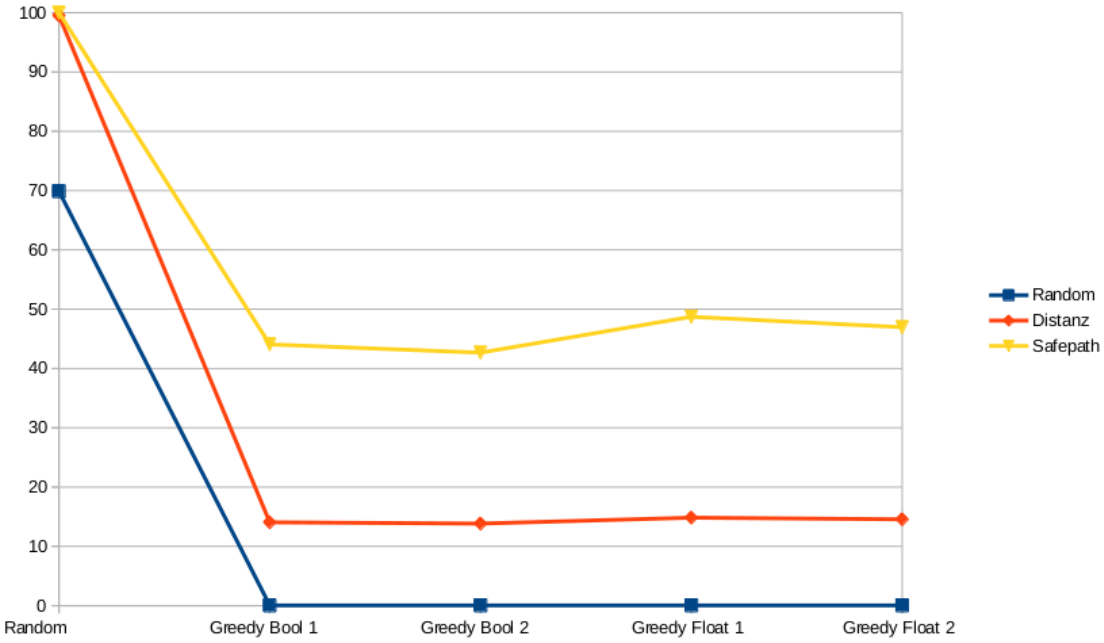

Abbildung 4.1: Prozentuale Gewinnrate von Mister X bei 6 Spielern

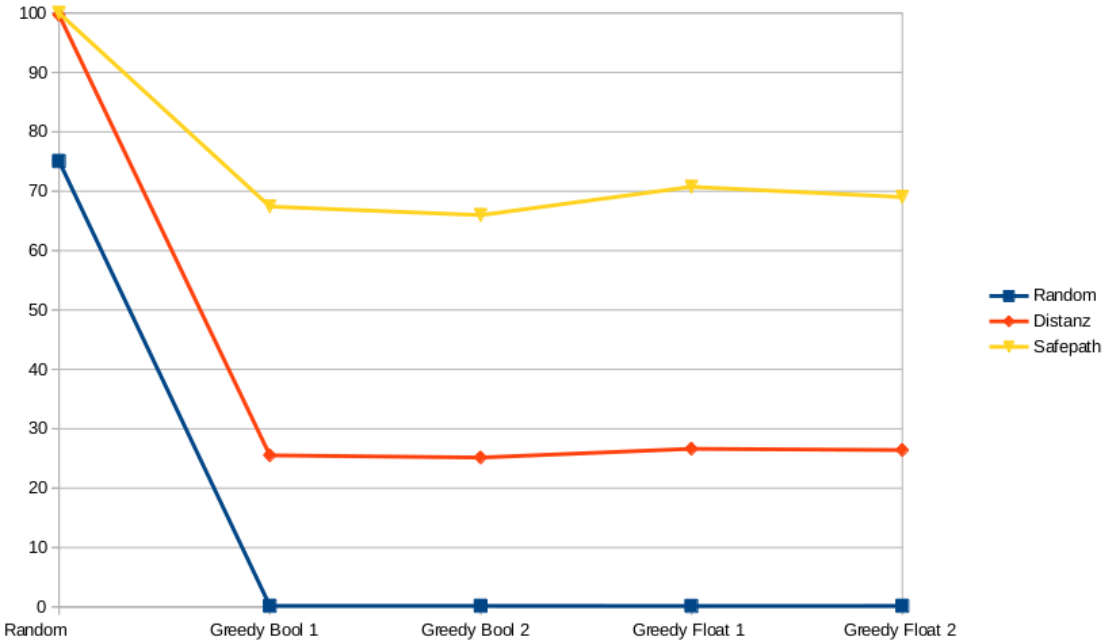

Abbildung 4.2: Prozentuale Gewinnrate von Mister X bei 5 Spielern

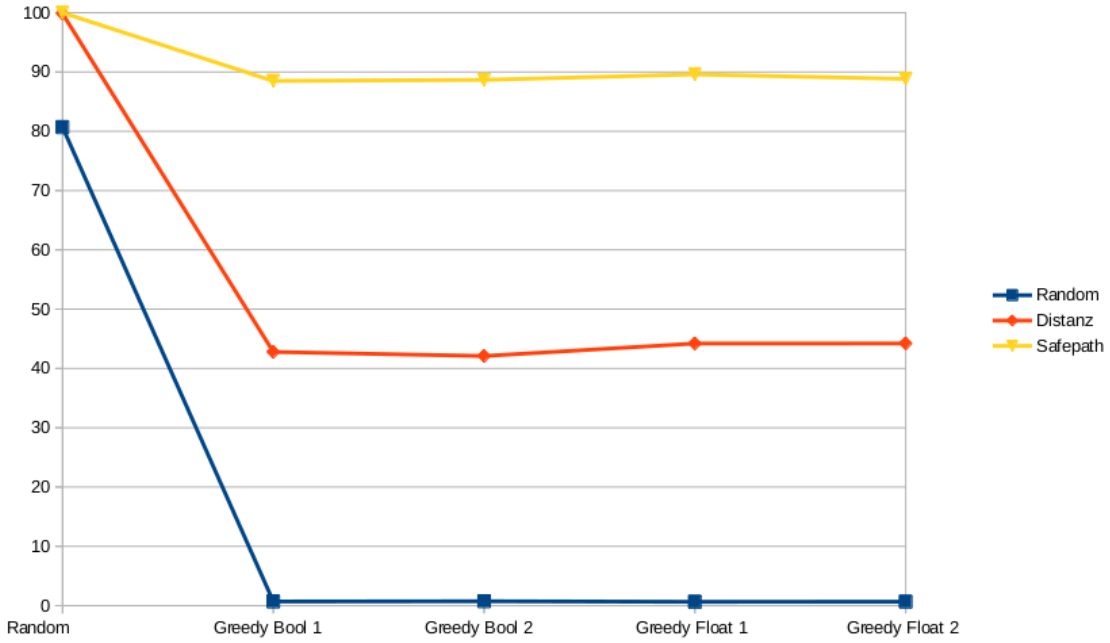

Abbildung 4.3: Prozentuale Gewinnrate von Mister X bei 4 Spielern

somit sagen, dass die genauen Aufenthaltswahrscheinlichkeiten eines zufällig ziehenden Mister X, der "Greedy Float 1/2" mehr schadet als hilft, da Mister X oft nicht zufällig zieht.
Zu beachten ist hier, dass "Greedy Bool 2" vorhersehbarer als "Greedy Bool 1" ist, da bei Greedy Bool 2 weitere Kriterien betrachtet werden, und somit der Zufall seltener relevant ist. Da hier die KIs keine Informationen über die gegnerische KI verfügen, ist das hier kein Nachteil. Sollte das allerdings nicht der Fall sein, könnte Mister X versuchen die Züge der Detektive vorherzusehen und auszunutzen.

4.3 RUNDENZAHL

Hier wird die Anzahl der gespielten Runden. Es wird sowohl die gesamte Anzahl aller Spiele sowie der Durchschnitt der Spiele, die von Mister X gewonnen bzw. verloren werden, berechnet.

4.3.1 *Gesamte Rundenzahl*

Die Tabellen 4.4 - 4.6 zeigen in den ersten drei Spalten die Gesamtzahl der Runden, die anderen drei Spalten zeigen die durchschnittliche Rundenzahl.

6 Spieler	Random	Distanz	Savepath	Random	Distanz	Savepath
Random	17537715	21092757	21827640	17,54	21,09	21,83
Greedy Bool 1	5426186	11668044	19043009	5,43	11,67	19,04
Greedy Bool 2	5420843	11636436	18966637	5,42	11,64	18,97
Greedy Float 1	5325052	11608702	19363462	5,33	11,61	19,36
Greedy Float 2	5320935	11567506	19249698	5,32	11,57	19,25

Tabelle 4.4: 1 Million Spiele mit 6 Spielern, die ersten 3 Spalten zeigen die Gesamtzahl der Runden, die anderen 3 die Durchschnittliche

5 Spieler	Random	Distanz	Savepath	Random	Distanz	Savepath
Random	17800042	20629308	21169856	17,80	20,63	21,17
Greedy Bool 1	5923746	13666427	21550436	5,92	13,67	21,55
Greedy Bool 2	5914387	13630587	21484781	5,91	13,63	21,48
Greedy Float 1	5795897	13598381	21746440	5,80	13,60	21,75
Greedy Float 2	5786619	13563753	21631587	5,79	13,56	21,63

Tabelle 4.5: 1 Million Spiele mit 5 Spielern, die ersten 3 Spalten zeigen die Gesamtzahl der Runden, die anderen 3 die Durchschnittliche

4 Spieler	Random	Distanz	Savepath	Random	Distanz	Savepath
Random	17842107	19925785	20268431	17,84	19,93	20,27
Greedy Bool 1	6633909	16023440	23388813	6,63	16,02	23,39
Greedy Bool 2	6635379	15995018	23475967	6,64	16,00	23,48
Greedy Float 1	6485775	16121169	23420630	6,49	16,12	23,42
Greedy Float 2	6486885	16107946	23399293	6,49	16,11	23,40

Tabelle 4.6: 1 Million Spiele mit 4 Spielern, die ersten 3 Spalten zeigen die Gesamtzahl der Runden, die anderen 3 die Durchschnittliche

4.3.2 *Gewonnene Spiele für Mister X*

Die durchschnittliche Anzahl der Runden bei den Spielen, die Mister X gewonnen hat, sind in den Tabellen 4.7 - 4.9 gegeben.
Die Rundenzahl bei der Distanz-KI und der Random-KI ist meist etwas kleiner als 23. Das ist die Anzahl der Tickets, die die Detektive haben plus 1. Bei Safepath dauert es 2 Züge länger, da Mister X hier seine Doppelzüge einsetzt.

6 Spieler	Random	Distanz	Savepath
Random	21,05	21,15	21,83
Greedy Bool 1	23,05	23,05	24,89
Greedy Bool 2	23,04	23,04	24,90
Greedy Float 1	23,05	23,05	24,88
Greedy Float 2	23,04	23,04	24,88

Tabelle 4.7: 1 Million Spiele mit 6 Spielern, Rundenzahl der von Mister X gewonnenen Spiele

5 Spieler	Random	Distanz	Savepath
Random	20,55	20,66	21,17
Greedy Bool 1	22,99	23,02	24,76
Greedy Bool 2	22,99	23,01	24,77
Greedy Float 1	23,00	23,01	24,75
Greedy Float 2	22,98	23,01	24,76

Tabelle 4.8: 1 Million Spiele mit 5 Spielern, Rundenzahl der von Mister X gewonnenen Spiele

4 Spieler	Random	Distanz	Savepath
Random	19,80	19,94	20,27
Greedy Bool 1	22,91	22,98	24,53
Greedy Bool 2	22,90	22,98	24,55
Greedy Float 1	22,89	22,98	24,50
Greedy Float 2	22,88	22,97	24,52

Tabelle 4.9: 1 Million Spiele mit 4 Spielern, Rundenzahl der von Mister X gewonnenen Spiele

4.3.3 *Verlorene Spiele für Mister X*

Die durchschnittliche Anzahl der Runden bei den Spielen, die die Detektive gewonnen haben, sind in den Tabellen 4.10 - 4.12 gegeben.
Je größer die Rundenzahl ist, desto länger hat es Mister X geschafft den Detektiven zu entkommen. Anhand der geringen Zahl, die bei der Random-Mister X-Spalte steht, sieht man, wie effektiv die Greedy-Detektiv-KIs dabei sind, Mister X zu fangen, wenn er nicht versucht zu entkommen.

6 Spieler	Random	Distanz	Savepath
Random	9,40	9,11	14,95
Greedy Bool 1	5,42	9,81	14,44
Greedy Bool 2	5,41	9,80	14,55
Greedy Float 1	5,32	9,62	14,12
Greedy Float 2	5,31	9,61	14,26

Tabelle 4.10: 1 Million Spiele mit 6 Spielern, Rundenzahl der von den Detektiven gewonnenen Spiele

5 Spieler	Random	Distanz	Savepath
Random	9,52	9,42	16,79
Greedy Bool 1	5,89	10,46	14,91
Greedy Bool 2	5,88	10,48	15,11
Greedy Float 1	5,77	10,18	14,50
Greedy Float 2	5,76	10,18	14,67

Tabelle 4.11: 1 Million Spiele mit 5 Spielern, Rundenzahl der von den Detektiven gewonnenen Spiele

4 Spieler	Random	Distanz	Savepath
Random	9,64	9,60	17,22
Greedy Bool 1	6,52	10,82	14,65
Greedy Bool 2	6,51	10,92	15,03
Greedy Float 1	6,38	10,69	14,14
Greedy Float 2	6,38	10,67	14,51

Tabelle 4.12: 1 Million Spiele mit 4 Spielern, Rundenzahl der von den Detektiven gewonnenen Spiele

4.3.4 *Auswertung*

Anhand der Rundenzahl lässt sich leicht sehen, wie eindeutig die Spiele sind. Vor allem an den Spielen, die für Mister X mit einer Niederlage enden, lässt sich auch die Spielstärke der einzelnen KIs abschätzen. Ein Random Mister X wird von Greedy-Detektiven im Durchschnitt in 6 Zügen gefangen. Bei der Distanz-KI dauert es etwa 10 Züge, bei der Savepath-KI 14 Züge. Auffällig ist, dass bei den gewonnenen Spielen, die Rundenzahl bei Random KI und Distanz KI in der Regel unter 23 liegen, bei Safepath

unter 25. Das liegt daran, dass nur die Safepath-KI Doppelzüge einsetzt und somit das Spiel zwei Runden länger dauern kann.
Interessant ist die Rundenzahl, vor allem für die Zeit, die benötigt wird, ein Spiel zu spielen. Je weniger Runden gespielt werden, desto schneller ist das Spiel vorbei und desto wahrscheinlicher gewinnen die Detektive.

4.4 BENÖTIGTE ZEIT

Die KIs benötigen Zeit für die Berechnung der Züge. Im vorangegangenen Abschnitt wurde die durchschnittliche Zugzahl betrachtet. In diesem Abschnitt wird die Zeit, die für das Ausspielen der einzelnen Spiele benötigt wird, untersucht. Die Zeitmessung wurde auf einem Intel I5 4570 mit 3.2 GHz ausgeführt, der Quelltext wird mit clang++ (Version 5.3.2) und der Optimierungsstufe 2 für ein 64 bit Linux kompiliert. Die Zeitmessungen wurden bei einem anderen Testdurchlauf erstellt. Dieser hatte zwar nur 100000 Durchläufe, trotzdem sind die Ergebnisse aufgrund der immer noch sehr großen Anzahl an Spielen repräsentativ. Die Ergebnisse werden in den Tabellen 4.13 - 4.15 aufgelistet.

6 Spieler	Random	Distanz	Savepath
Random	0m 42,30s	1m 30,30s	1m 37,13s
Greedy Bool 1	2m 41,80s	5m 59,34s	8m 44,41s
Greedy Bool 2	2m 47,56s	6m 12,76s	8m 46,01s
Greedy Float 1	2m 49,40s	6m 16,12s	9m 01,86s
Greedy Float 2	2m 48,31s	6m 11,49s	8m 58,46s

Tabelle 4.13: Dauer von 100 000 Spielen, bei 6 Spielern

5 Spieler	Random	Distanz	Savepath
Random	0m 37,43s	1m 23,54s	1m 29,90s
Greedy Bool 1	2m 23,73s	5m 23,36s	7m 26,22s
Greedy Bool 2	2m 14,11s	5m 11,50s	7m 27,32s
Greedy Float 1	2m 19,47s	5m 31,24s	7m 52,11s
Greedy Float 2	2m 14,86s	5m 22,01s	7m 39,64s

Tabelle 4.14: Dauer von 100 000 Spielen, bei 5 Spielern

Die Zeit, die für die einzelnen Spiele benötigt wird, soll nicht die Performance der einzelnen Algorithmen messen, sondern ist später für Monte-Carlo-Tree-Search relevant, da dort mehrere Spiele simuliert werden. Greedy Float 1 gegen Savepath dauerte

4 Spieler	Random	Distanz	Savepath
Random	0m 31,45s	1m 13,08s	1m 19,78s
Greedy Bool 1	1m 51,78s	4m 30,08s	6m 09,74s
Greedy Bool 2	1m 51,37s	4m 29,02s	6m 13,81s
Greedy Float 1	1m 46,68s	4m 38,44s	6m 14,85s
Greedy Float 2	1m 50,18s	4m 29,90s	6m 24,08s

Tabelle 4.15: Dauer von 100 000 Spielen, bei 4 Spielern

für 6 Spieler und 100000 Spiele etwa 9 Minuten. Das entspricht etwa 185 Spielen pro Sekunde.

4.5 AUSWERTUNG

Sowohl die Safepath-KI für Mister X als auch die Greedy Bool-KI haben beim Spiel mit 6 Spielern eine gute Siegchance, und sind ausreichend schnell um auf einem aktuellen Computer etwa 200 Spiele pro Sekunde zu spielen. Die Random-KIs sind sehr schnell in der Berechnung ihrer Züge. Allerdings liefern sie ziemlich schlechte Ergebnisse.

5. MONTE CARLO TREE SEARCH

Monte-Carlo-Tree-Search (MCTS) ist ein Algorithmus, der von einem Agenten eingesetzt werden kann, um Entscheidungen zu treffen. Der Algorithmus baut einen Baum auf, in dem jeder Knoten einen Spielzustand darstellt und jede Kante einen Übergang. Die Anzahl der möglichen Zügen beschreibt somit die Breite b des Baums und die Gesamtanzahl der Züge die Tiefe n. Somit hätte ein vollständiger Baum insgesamt b^n Endknoten. Es ist somit für viele Situationen unmöglich einen vollständigen Baum zu erzeugen. MCTS baut nur einen Teil dieses Baumes auf und simuliert an jedem Knoten das Spiel bis zum Ende oder verwendet eine vorhandene Heuristik. Zur Simulation des Spiels kann ein beliebiger Algorithmus verwendet werden, wozu auch das zufällige Auswählen von Zügen zählt, allerdings liefert MCTS für realistische Simulationen bessere Ergebnisse. Der durch MCTS erstellte Baum ist in der Regel nicht balanciert, da der Algorithmus vielversprechende Knoten bevorzugt betrachtet. MCTS führte zu einer starken Verbesserung von KIs für das Spiel Go, die aufgrund der immensen Zahl an möglichen Zügen zuvor menschlichen Spielern weit unterlegen war.

5.1 ALLGEMEINER ALGORITHMUS

MCTS lässt sich in vier Teile aufteilen. Zum ersten die Selection, dann die Expansion, die Simulation und schließlich die Backpropargation. Die einzelnen Schritte werden im Folgenden erläutert. Diese Schritte werden solange durchgeführt, bis eine bestimmte Abbruchbedingung eintritt, häufig eine Zeitbeschränkung oder eine vorgegebene Anzahl an Iterationen.

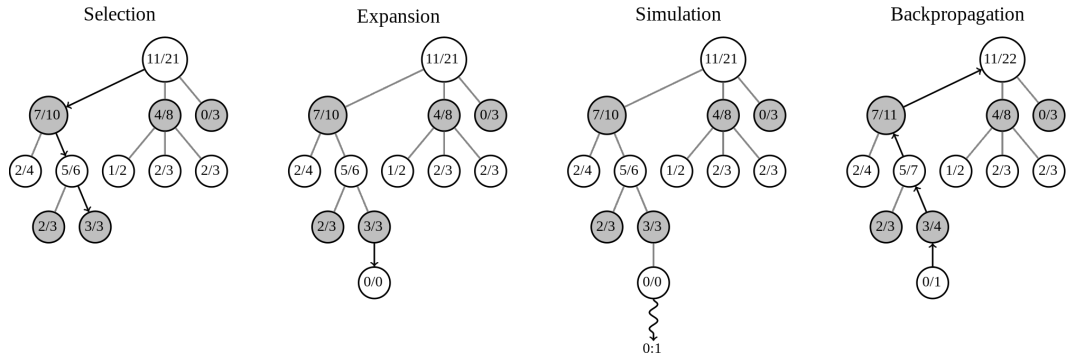

Abbildung 5.1: Die einzelnen Phasen von MCTS, die erste Zahl zeigt die Anzahl der gewonnenen Spiele, die Zweite die Gesamtzahl der Besuche des Knotens. [6]

5.1.1 *Selection*

Im der Selection-Phase wird der Knoten ausgewählt, der näher untersucht werden soll. Begonnen mit der Wurzel wird rekursiv ein Kindknoten ausgewählt, bis der ausgewählte Knoten keine Kinder hat. Sobald dieser erreicht ist, beginnt die Expansion-Phase.
Um einen Knoten auszuwählen, können je nach Selection-Strategie verschiedene Faktoren berücksichtigt werden. In der Regel wird versucht eine Balance zwischen dem erkunden neuer Optionen und dem genaueren Untersuchen erfolgversprechender Knoten herzustellen. Jedem Knoten wird ein Wert zugewiesen, der sich aus der Summe von bisherigem Ergebnis und einem Erkundungswert zusammensetzt. Ein unerkundeter Kindknoten hat hierbei immer den Erkundungswert ∞ und wird somit immer bevorzugt gewählt. Aus diesen wird der Knoten mit dem größten Wert ausgewählt; bei mehreren gleichen wird einer davon zufällig ausgewählt.
In Abblidung 5.1 wird in dieser Phase zuerst der Knoten (7/10), dann (5/6) und schließlich (3/3) ausgewählt. Da (3/3) keine Kindknoten hat, ist die Phase beendet.

5.1.2 *Expansion*

Hier wird ein weitere Knoten zum Baum hinzugefügt. Dieser stellt einen zufälligen Spielzug dar, der nach dieser Folge von Zügen möglich ist. Anschließend wird dieser Knoten ausgewählt.
In Abblidung 5.1 wird an dieser Stelle unter dem ausgewählten Knoten (3/3) ein neuer Kindknoten (0/0) angelegt.

5.1.3 *Simulation*

In dieser Phase wird vom ausgewählten Knoten das Spiel bis zum Ende simuliert. Zur Simulation kann ein beliebiger Algorithmus eingesetzt werden. Für gewöhnlich hat ein Spiel ein Ergebnis im Bereich [0, 1]. Hierbei bezeichnet die 1 den Sieg, die 0 eine Niederlage. Theoretisch sind auch höhere oder negative Werte möglich.
Das Spiel wird in Abblidung 5.1, von der resultierenden Stellung (0/0) aus, simuliert. Das Ergebnis ist eine 0 (Niederlage).

5.1.4 *Backpropargation*

Das Ergebnis der Simulation wird zum Ergebnis aller besuchten Knoten addiert, zudem wird ein Zähler, der die Anzahl der Besuche misst, inkrementiert.
In Abbildung 5.1 wird das Ergebnis auf den Ergebniswert aller besuchten Knoten addiert und die Anzahl der Besuche inkrementiert.

5.2 SELECTION STRATEGIE

Das Auswählen des nächsten Knotens ist ein sehr wichtiger Teil des Algorithmus. Es muss ein guter Kompromiss zwischen dem Erkunden neuer Knoten und dem genaueren Untersuchen bestehender Knoten gemacht werden. Auer [4] hat hierfür eine einfache Formel vorgeschlagen, sobald der Wert aller Kindknoten berechnet wurde, wird das Kind mit dem höchsten Wert gewählt. UCB1 steht für Upper-Confidance-Bound, UCT für Upper-Confidance-Tree, sie werden in der Arbeit von Auer näher erläutert.

$$UCB1 = X_j + \sqrt{\frac{2 \cdot \ln(n)}{n_j}}$$

X_j stellt hier das durchschnittliche Ergebnis des Zuges j dar. n die Gesamtzahl der Besuche des Elternknotens und n_j die Gesamtzahl der Besuche des untersuchten Knotens. Für Ergebniswerte im Bereich [0, 1] stellt diese Formel einen sinnvollen Kompromiss zwischen vielversprechenden, häufig besuchten und damit bereits gut erkundeten Knoten, und Knoten, die bisher schlechtere Ergebnisse geliefert haben, aber auch noch wenig erkundet sind, dar.
Um für andere Bereiche als [0, 1] sinnvolle Ergebnisse zu liefern wird der zweite Summand gewichtet. Er wird zudem verwendet, um die Balance zwischen Erkunden und Untersuchen zu modifizieren.

$$UCT = X_j + C \cdot \sqrt{\frac{2 \cdot \ln(n)}{n_j}}$$

In Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard [2] wird zudem noch eine Heuristik verwendet, um Erkenntnisse aus vorangegangenen Spielen einzubringen.
In der Arbeit wird als Selection Strategie die UCT-Formel mit $C = 1$ verwendet, da die Ergebnisse bereits im Bereich [0, 1] liegen.

5.3 MINIMAX

Der MiniMax-Algorithmus baut für einen bestehenden Spielstand einen vollständigen Baum auf, wobei in jedem Endknoten der Ergebniswert des Spiels steht. Dieser Wert wird bis zur Wurzel hochpropagiert, wobei je nach Knoten das größte oder das kleinste Kind seinen Wert an den Vater weitergibt. Das Problem mit MiniMax ist, dass es eine Zeitkomplexität von $O(b^m)$ hat, was der Anzahl der Endknoten entspricht, wobei b die Anzahl der Zugmöglichkeiten (die Breite) und m die Gesamtzahl der möglichen Züge (die Tiefe) ist. Wenn man annimmt, dass in Scoland Yard jeder Spieler 5 verschiedene Zugmöglichkeiten hat, ein Spiel 24 Runden dauert und mit 6 Spielern gespielt wird, ergeben sich daraus $5^{(24 \cdot 6)} \cdot \approx 4.5 \cdot 10^{100}$ Endknoten, was nicht in absehbarer Zeit zu berechnen ist. [7]

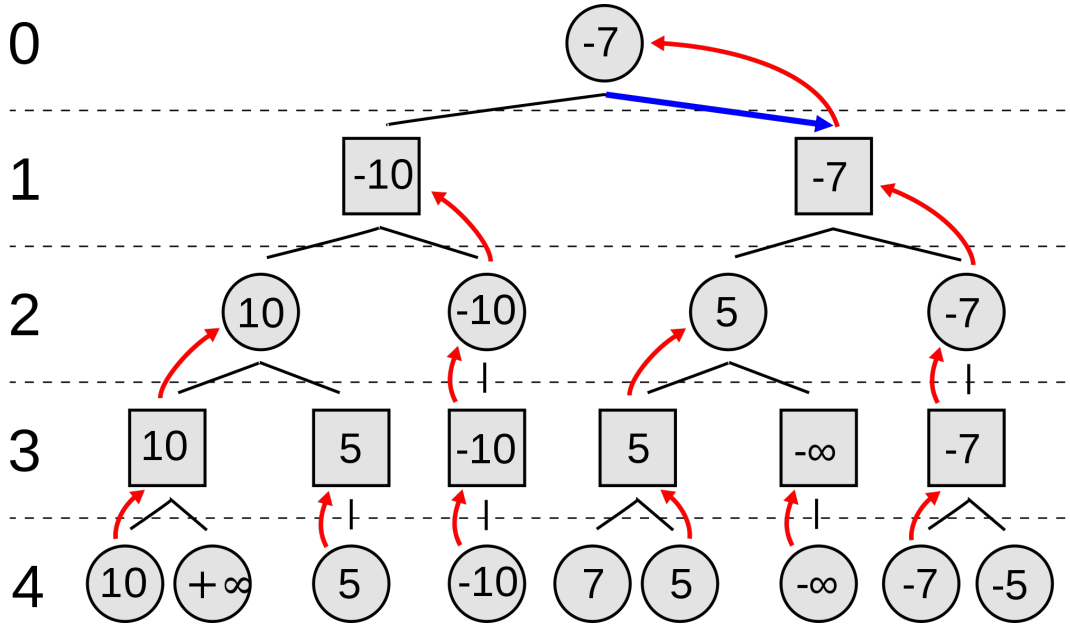

Abbildung 5.2: Klassischer MiniMax, die Kreise sind zu maximieren, die Quadrate zu minimieren, der blaue Pfeil stellt den besten Zug dar. [8]

5.3.1 *MiniMax für MCTS*

Der beschriebene Algorithmus ist in dieser Form nur für Ein-Spieler-Spiele geeignet. Da es sich bei Scotland Yard um ein Spiel mit zwei Teams handelt, muss der Algorithmus angepasst werden. Beim klassischen MiniMax Algorithmus wird ein Knoten nach dem Ergebnis bewertet, das den größten Nutzen bringt, sollte der Gegenspieler perfekt spielen. Bei MCTS wird allerdings in jedem Knoten das erwartete Ergebnis gespeichert, das sich aus den bisherigen Simulationen berechnet.
Um in der Selection-Phase einen Knoten auszuwählen, sollte berücksichtigt werden, dass der Gegner nicht die für den Spieler besten Züge macht. Eine Möglichkeit wäre nach jeder Backpropargation, jedem Knoten einen zusätzlichen MiniMax Wert zuzuweisen und anhand von diesem die Selection-Phase durchzuführen. [9] In A Survey of Monte Carlo Tree Search Methods [4] wird vorgeschlagen einen NegaMax-Algorithmus zu verwenden.

5.3.2 *NegaMax*

Beim NegaMax wird das Ergebnis nach jedem Schritt in der Backpropargation-Phase invertiert. Dadurch wird in der Selection-Phase während der gegnerischen Züge immer der für den Spieler schlechteste Zug gewählt. Somit werden realistischere Sze-

narien häufiger untersucht, als solche, bei denen der Gegner dem Spieler regelrecht hilft.

5.4 ERGEBNIS

Nach dem die Abbruchbedingung für die MCTS eingetreten ist, muss ein Ergebnis zurückgegeben werden. A Survey of Monte Carlo Tree Search Methods [4] listet 4 Möglichkeiten auf, den besten Knoten zu ermitteln:

- Max-Child: Der Knoten mit dem besten Ergebnis
- Robust-Child: Der Knoten, der am häufigsten besucht wurde
- Max-Robust-Child: Der Knoten, der sowohl das beste Ergebnis als auch am häufigsten besucht wurde. Sollte es keinen solchen geben, wird MCTS weiter ausgeführt bis es einen solchen gibt.
- Secure-Child: Eine Kombination aus Max-Child und Robust-Child

Verwendet wird hier Max-Child, also das Kind, dass das beste Ergebnis liefert. Robust-Child sollte meist das selbe Ergebnis wir Max-Child ergeben, allerdings werden dadurch gute Züge, die erst spät entdeckt werden, vernachlässigt. Max-Robust Child ist keine Option, da jetzt ein Ergebnis gefordert wird und nicht erst, sobald Max Child auch das robusteste ist, was mehr Rechenleistung beansprucht. Für Secure Child müsste erst eine Gewichtung für die beiden Teile evaluiert werden.

5.5 IMPLEMENTIERUNG

Bei meiner Implementierung des MCTS-Algorithmus habe ich einige kleine Modifikationen zu dem MCTS-Algorithmus gemacht, um sowohl die Leistung zu erhöhen als auch den Quelltext einfacher zu halten:

$$UCT = \frac{Erg_i}{n_i} + C \cdot \sqrt{\frac{2 \cdot \ln(n_i)}{n_p}}$$

wurde durch

$$UCT = \frac{Erg_i}{n_i + \epsilon} + C \cdot \sqrt{\frac{2 \cdot \ln(n_i + \epsilon)}{n_p}} + \epsilon \cdot Zufallswert$$

ersetzt, um bei Knoten die bisher noch nicht besucht wurden nicht durch 0 zu teilen oder den Logarithmus von 0 berechnen zu müssen. Erg_i ist die Summe aller bisherigen Ergebnisse, die der Knoten i bisher erzeugt hat. n_i ist die Anzahl der Besuche des Knotens i, n_p die Anzahl der Besuche des Elternknotens von i. ϵ ist definiert als

0.000001 und der Zufallswert stellt eine Zahl im Bereich [0, 1] dar und soll verhindern, dass bei der Auswahl nicht darauf geachtet werden muss, dass es zwei Knoten mit gleicher Bewertung gibt.
Die Implementierung verwendet eine Variante von NegaMax, bei der das Ergebnis nicht bei der Backpropargation invertiert wird, sondern bei der Selection Phase. Das hat den Grund, dass so im Baum leichter nachzuvollziehen ist, wie dieser sich aufbaut und so die Suche für den Spieler an jedem Knoten sein Ergebnis beinhaltet und nicht das Ergebnis der Gegner.
Der Einfachheit halber werden bei meiner Implementierung immer alle Kindknoten auf einmal angelegt, sobald in der Expansion-Phase ein neuer Knoten hinzugefügt wird. Der Grund hierfür ist, dass so einfacher zu erkennen ist, ob ein Knoten erweiterbar ist oder nicht, da ein erweiterbarer Knoten keine Kinder hat. So muss auch nur einmal berechnet werden, welche möglichen Kindknoten es gibt. Da der Knoten laut dem MCTS-Algorithmus noch nicht existiert, wird jeder Knoten der bisher noch nicht besucht wurde, nicht erweitert, da er faktisch erst durch das erste Besuchen zum Baum hinzugefügt wird.

5.6 VOR- UND NACHTEILE VON MCTS

5.6.1 *Vorteile*

MCTS ist ein einfacher Algorithmus und ist bei fast allen rundenbasierten Spielen einsetzbar. Es ist relativ einfach zu implementieren und benötigt zum Funktionieren nur eine Möglichkeit den aktuellen Spielstand zu bewerten. Die kann auch durch das zufällige Ausspielen des Spiels entstehen.
Der Algorithmus kann jederzeit abgebrochen werden und liefert auch dann ein akzeptables Ergebnis. Es ist nicht notwendig, dass erst ein kompletter Spielbaum aufgebaut werden muss, um den nächsten Zug zu ermitteln.

5.6.2 *Nachteile*

Ein offensichtlicher Nachteil von MCTS ist die benötigte Rechenleistung. Bei 1000 Iterationen muss das Spiel pro Zug 1000 mal simuliert werden.
Ebenso liefert MCTS nicht unbedingt den besten Zug, sondern den, der sich aus den Playouts ergibt. Bei einer schlechten Simulation-Strategie kann MCTS auch schlechte Ergebnisse liefern.

6. EVALUATION UND BEWERTUNG DER MCTS FÜR MISTER X

Im Folgenden werden die Ergebnisse der Testläufe analysiert und bewertet. Es gibt sehr viele Parameter um die MCTS Simulation zu konfigurieren, und da die Ausführung der Tests sehr lang dauert, können leider nicht alle Konfigurationen getestet werden. Für die Selection-Strategie werden keine Alternativen getestet.

6.1 PARAMETER DER SIMULATION

Für die Simulation müssen sowohl für Mister X als auch für die Detektive eine KI ausgewählt werden. Aus den vorgestellten Strategien in Kapitel 3 ergeben sich drei KIs für Mister X und fünf KIs für die Detektive. Daraus ergeben sich 15 mögliche Konfigurationen für MCTS. Zusätzlich kann noch die Anzahl der Iterationen für die MCTS variiert werden und schließlich muss noch die Detektiv-KI gewählt werden, gegen die Mister X wirklich spielt. Daraus ergeben sich für jede Spieleranzahl $5 \cdot 3 \cdot It \cdot 5 = 75 \cdot It$ mögliche Kombinationen von Match-Ups. Da bereits erwähnt wurde, dass die Tests sehr lange dauern können, ist es nicht möglich alle Kombinationen zu testen. Da vier der Detektiv-KIs sehr ähnlich sind wird nur eine davon für die Tests herangezogen, und da die Random-KIs, wie in Kapitel 4 festgestellt wurde, keinen ernstzunehmenden Gegner darstellen, wird diese als Gegner für MCTS nicht eingesetzt, als Simulationsgegner hingegen schon.
Als Notation wird hier folgendes verwendet: $MCTS_s(KI_X, KI_D, It)$ wobei s das simulierende Team ist, in diesem Fall also X für Mister X, KI_X die KI von Mister X, KI_D die Detektiv-KI und It die Anzahl der Iterationen, die die MCTS durchläuft.
Getestet werden für MCTS jede Kombination aus den Mister X-KIs und den Detektiv-KIs Random und Greedy Bool 1 mit 100, 500, 1000 und 2000 Iterationen jeweils mit Greedy Bool 1 als Gegner. Ebenso werden einige Testspiele eines menschlichen Spielers gegen die MCTS Konfiguration mit den besten Ergebnissen durchgeführt. Die Greedy Bool 1 wird gewählt, weil sie bessere Ergebnisse als die Greedy Float-KIs erbrachte und bei guten Ergebnissen zufälligere Ergebnisse als Greedy Bool 2 liefert, und somit mehr Zugmöglichkeiten der Detektive abdeckt.

6.2 ERGEBNISSE

Aufgrund der Dauer der einzelnen Spiele werden nur 1000 Spiele gespielt. Die Spiele werden immer gegen die Greedy Bool 1-KI der Detektive getestet. Hierbei gelten folgende Abkürzungen: R_X: Random-Mister X-KI, D_X: Distanz-Mister X-KI, S_X: Safepath-

Mister X-KI, R_D: Random-Detektive-KI, G_D: Greedy-Bool-1-Detektive-KI. Die Spalten stellen die Konfiguration der Simulation dar, die Zeilen die Anzahl der Iterationen.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	73	22	81	577	77	712
500	135	62	309	463	221	712
1000	158	106	368	451	268	721
2000	175	155	410	449	335	709

Tabelle 6.1: 1000 Spiele mit 6 Spielern, Detektiv-KI: Greedy Bool 1, Gewinne von Mister X

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	139	89	164	643	132	786
500	249	203	474	586	357	841
1000	269	267	558	548	448	799
2000	278	279	595	554	504	798

Tabelle 6.2: 1000 Spiele mit 5 Spielern, Detektiv-KI: Greedy Bool 1,Gewinne von Mister X

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	293	257	321	779	276	854
500	462	376	690	722	555	910
1000	502	406	728	721	633	899
2000	501	472	736	754	666	898

Tabelle 6.3: 1000 Spiele mit 4 Spielern, Detektiv-KI: Greedy Bool 1,Gewinne von Mister X

6.3 BEWERTUNG

Die Ergebnisse der Testläufe in den Tabellen 6.1 - 6.3 und Abbildungen 6.1 - 6.3 zeigen ein interessantes Bild. Zum einen zeigt sich, dass durch das verwenden von MCTS die Siegchancen gegenüber dem nichtplanenden Algorithmus deutlich bessere Ergebnisse liefern können. $MCTS_X(S_X, G_D, It)$ liefert bei 6 Spielern eine Gewinnrate von ca. 70% gegenüber dem Ergebnis der nichtplanenden Safepath-KI, die gegen die Greedy Bool 1-KI eine Gewinnrate von 44% hat, ist das eine deutliche Verbesserung. Auch die Distanz-KI kann so von 14% auf 45% verbessert werden.
Interessant ist, dass das Erhöhen der Iterationen bei den Konfigurationen $MCTS_X$ (S_X, G_D, It) und $MCTS_X(D_X, G_D, It)$ fast keinen Einfluss auf die Ergebnisses zur

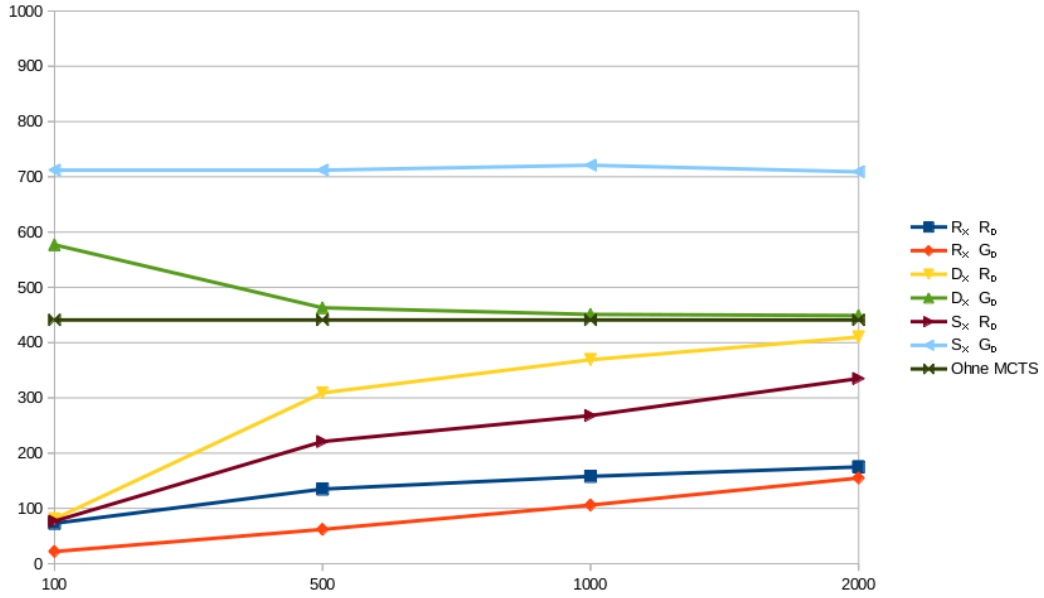

Abbildung 6.1: Anzahl der Siege von Mister X gegen 5 Greedy Bool 1 Detektive in Abhängigkeit von der Anzahl von MCTS Iterationen (6 Spieler), Greedy Bool 1 gegen Safepath als Vergleichswert

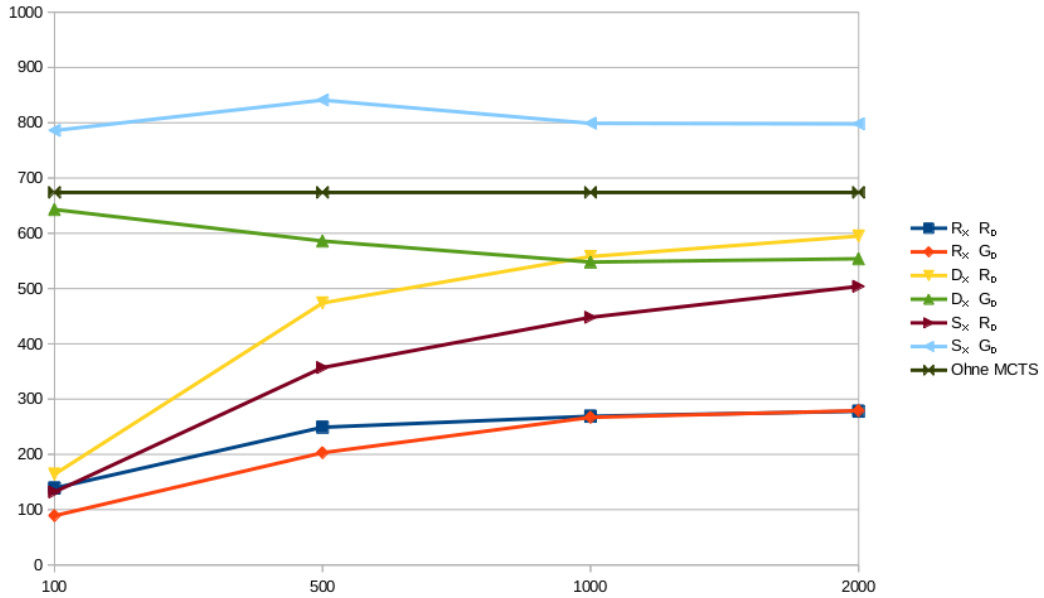

Abbildung 6.2: Anzahl der Siege von Mister X gegen 4 Greedy Bool 1 Detektive in Abhängigkeit von der Anzahl von MCTS Iterationen (5 Spieler), Greedy Bool 1 gegen Safepath als Vergleichswert

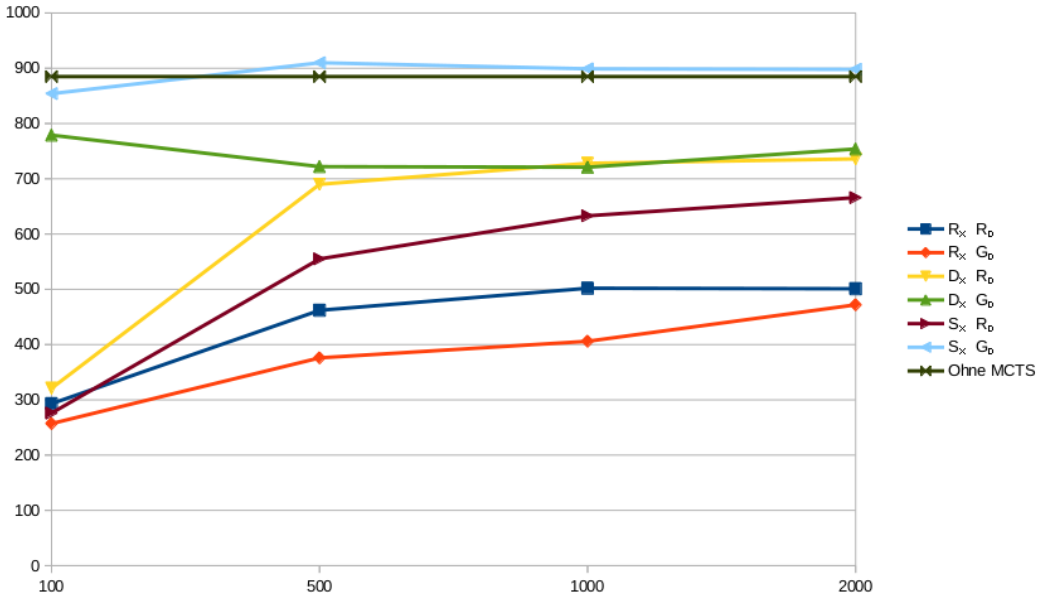

Abbildung 6.3: Anzahl der Siege von Mister X gegen 3 Greedy Bool 1 Detektive in Abhängigkeit von der Anzahl von MCTS Iterationen (4 Spieler), Greedy Bool 1 gegen Safepath als Vergleichswert

Folge hat. Bei $MCTS_X(D_X, G_D, It)$ nimmt die Gewinnrate von Mister X sogar deutlich ab: von 577 auf 449 bei 6, 643 auf 554 bei 5 und von 779 auf 721 bei 4 Spielern, was jeweils eine Reduktion von ca. 10% ist. Zu beachten ist auch, dass beim Spiel mit 4 Spielern das Einsetzen von MCTS fast keinen Vorteil gegenüber der nichtplanenden Safepath-KI bringt.
Die Ergebnisse der anderen Konfigurationen verbessern sich allerdings durch die Erhöhung deutlich. Bemerkenswert ist, dass alleine durch das Nutzen der Random-KIs durch MCTS im 6 Spieler Spiel eine Gewinnrate von 17,5% gegen die Greedy Bool 1-KI der Detektive, entstanden ist, während ohne MCTS die Random-KI die Siegchancen bei unter 0,2% liegen. Im Spiel mit 4 Spielern wird so sogar eine Gewinnrate von etwa 50% erzielt, gegenüber 0,7% ohne MCTS.
Interessanterweise sind die Ergebnisse von $MCTS_X(R_X, G_D, It)$ meist schlechter als die von $MCTS_X(R_X, R_D, It)$. Sie nähern sich bei zunehmender Zahl der Iterationen aneinander an. Möglicherweise schätzt $MCTS_X(R_X, G_D, It)$ die Ergebnisse der Simulationen zu schlecht ein, da das Ergebnis bei fast allen getesteten Zügen wegen der schwachen Spielstärke der Random-KI eine Niederlage ist, und macht somit eher einen zufälligen Zug.
Das letzte Interessante, das sich aus den Ergebnissen entnehmen lässt, ist, dass $MCTS_X(D_X, R_D, It)$ besser als $MCTS_X(S_X, G_D, It)$ abschneidet. Die Ergebnisse werden bei zunehmenden Iterationen besser, allerdings liegen die Ergebnisse der Distanz-KI im-

mer über der eigentlich spielstärkeren Safepath-KI. Erklären lässt sich das eventuell dadurch, dass die Safepath-Random-Simulation zu optimistische Ergebnisse liefert im Gegensatz zu den etwas schlechteren Ergebnissen der distanzbasierten Simulation. Somit überschätzt die Safepath-Simulation ihre Stellung und macht dementsprechend schlechtere Züge.

6.3.1 *Tiefe des Baums*

Anzumerken ist, dass die Tiefe des Baums häufig nicht bis zum nächsten Zug von Mister X reicht (natürlich abgesehen von dem Fall, dass Mister X einen Doppelzug macht). Die folgenden Berechnungen basieren darauf, dass der Baum balanciert sei. MCTS produziert in der Regel zwar einen unbalancierten Baum, da aber für diese Rechnung die Ergebnisse der Simulationen keine Rolle spielen, wird das an dieser Stelle vernachlässigt.
Die kleinste Zahl an ausgehenden Kanten im Graph von Scotland Yard ist 2, die größte 13. Der Durchschnitt liegt bei 4,7. Sollte Mister X auf einem Feld mit 4 Verbindungen stehen, und hat er sowohl noch ein Blackticket als auch einen Doppelzug übrig, resultieren daraus für ihn 4 normale Züge, 4 Züge mit einem Blackticket, 4 normale Züge mit Doppelzug und 4 Züge mit Blackticket und Doppelzug. Insgesamt hat er also 16 verschiedene Züge, aus denen einer ausgewählt werden muss. In einem 6 Spieler Spiel nehmen wir einmal an, dass alle Detektive auf Feldern stehen, die auch über 4 Verbindungen verfügen. Die Gesamtzahl aller möglichen Detektivzüge zwischen den Zügen von Mister X wäre also $4^5 = 1024$. Daraus ergeben sich $16 * 1024 = 16384$ Möglichkeiten, wie ein Zug ausgespielt werden kann. Bei 2000 Iterationen ist es also unwahrscheinlich, dass Mister X einen zweiten Zug berechnet (außer er verwendet einen Doppelzug). In diesem Fall wäre die Tiefe des Baums etwa 5 (ein balancierter Baum der Tiefe 4 hat $16 * 4^3 = 1024$ Knoten auf der untersten Ebene. 2000 weniger als 4 mal größer als 1024. Somit erreicht er eine Tiefe von 5).

6.4 DAUER

Wie bisher schon angedeutet dauern die Spiele durch das Simulieren vieler Testspiele bei MCTS-Algorithmen deutlich länger. Die Zeiten der verschiedenen Konfigurationnen sind in den Tabellen 6.4 - 6.6 aufgelistet.
Wie man sieht, unterscheidet sich die Dauer der Tests stark. Sie liegen zwischen ca. 4 Minuen für die $MCTS_X(R_X, R_D, 100)$ bis hinzu über 27 Stunden für $MCTS_X(S_X, G_D, 2000)$. Das entspricht 98 Sekunden pro Spiel. Als Vergleichswert: Ohne MCTS können in einer Sekunde 185 Spiele mit diesen KIs gespielt werden.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	5m 06,2s	10m 40,4s	10m 49,2s	57m 48,7s	10m 49,2s	76m 05,0s
500	31m 40,0s	66m 36,5s	66m 33,0s	289m 47,5s	65m 39,3s	392m 33,6s
1000	65m 21,5s	157m 12,0s	139m 41,7s	554m 44,9s	139m 04,8s	764m 41,9s
2000	131m 57,6s	348m 23,8s	295m 23,2s	1182m 29,8s	292m 54,4s	1633m 21,0s

Tabelle 6.4: 1000 Spiele mit 6 Spielern, Detektiv-KI: Greedy Bool 1, Dauer der Tests

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	4m 37,9s	11m 19,9s	9m 20,4s	53m 55,5s	10m 07,4s	53m 55,5s
500	28m 16,0s	77m 59,5s	61m 41,9s	263m 44,8s	62m 59,3s	344m 05,4s
1000	58m 02,0s	172m 46,6s	132m 26,8s	525m 33,1s	125m 18,9s	665m 21,7s
2000	116m 01,8s	351m 48,11s	275m 36,2s	1045m 07,0s	276m 51,0s	1355m 23,5s

Tabelle 6.5: 1000 Spiele mit 5 Spielern, Detektiv-KI: Greedy Bool 1, Dauer der Tests

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	4m 12,1s	13m 11,3s	9m 00,9s	47m 35,4s	9m 55,9s	52m 19,6s
500	24m 12,3s	72m 58,0s	56m 29,4s	221m 19,0s	55m 38,7s	269m 30,0s
1000	48m 40,2s	152m 37,4s	118m 06,7s	446m 35,1s	120m 22,4s	551m 21,3s
2000	97m 38,3s	323m 39,9s	238m 13,4s	891m 12,2s	244m 42,8s	1064m 55,0s

Tabelle 6.6: 1000 Spiele mit 4 Spielern, Detektiv-KI: Greedy Bool 1, Dauer der Tests

6.4.1 *Zeit pro Zug*

Wenn die in den Tabellen 6.4 - 6.6 angegebenen Zeiten durch die Anzahl der gespielten Züge geteilt werden, erhält man in etwa die durchschnittliche Zeit, bis ein Zug berechnet wird. Die Zeit, die die Detektive während dieser Spiele benötigen, ist vernachlässigbar. In Kapitel 4 wurde die Dauer eines Spiels zwischen Safepath und Greedy-Bool-1 als etwa 0,0054 Sekunden berechnet, was bei 1000 spielen 5,4 Sekunden entsprechen. Je weiter das Spiel voranschreitet, desto mehr nimmt die Zeit, die für die Berechnung der Züge benötigt wird, ab. Das hat den einfachen Grund, dass die Simulationen, die MCTS durchführt, schneller durchgeführt werden können. Wenn man davon ausgeht, dass die Zeit pro Zug linear abnimmt, und für den letzten Zug keine Zeit benötigt wird, entspricht die durchschnittliche Zugdauer der halben Zeit, die für die Berechnung des ersten Zuges benötigt wird. Die Anzahl der Runden für die Tests sind in den Tabellen 6.7 - 6.9 gegeben.
Die sich daraus ergebenden durchschnittliche Berechnungszeiten finden sich in den Tabellen 6.10 - 6.12.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	11430	6397	10281	18491	10031	20365
500	13684	7673	15956	16730	14083	20555
1000	14281	9187	17259	16798	15499	20742
2000	14503	10206	18315	16916	16717	20774

Tabelle 6.7: 1000 Spiele mit 6 Spielern, Detektiv-KI: Greedy Bool 1, Gesamtzahl gespielter Runden

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	12568	8412	11588	19390	10952	21575
500	15461	11301	17867	18761	15922	22396
1000	16055	12822	19702	18385	17684	22045
2000	16117	13203	20726	18653	18845	22036

Tabelle 6.8: 1000 Spiele mit 5 Spielern, Detektiv-KI: Greedy Bool 1, Gesamtzahl gespielter Runden

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	15016	12375	13972	21532	13132	22352
500	18408	14845	20266	20839	18282	23581
1000	18877	15283	21500	20932	19902	23553
2000	19131	16585	22166	21503	20934	23664

Tabelle 6.9: 1000 Spiele mit 4 Spielern, Detektiv-KI: Greedy Bool 1, Gesamtzahl gespielter Runden

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	0,03s	0,10s	0,06s	0,19s	0,06s	0,22s
500	0,14s	0,52s	0,25s	1,04s	0,28s	1,15s
1000	0,27s	1,03s	0,49s	1,98s	0,54s	2,21s
2000	0,55s	2,05s	0,97s	4,19s	1,05s	4,72s

Tabelle 6.10: Durchschnittliche Zeit, die zur Zugberechnung benötigt wird. (6 Spieler)

Wie zu erwarten, dauern die Berechnungen etwa entsprechend der Zahl der Iterationen länger, in der Regel also für 2000 Iterationen doppelt so lange wie für 1000. Hier zeigt sich auch wieder, dass die Random-KI im Gegensatz zu den Greedy-KIs deutlich schneller zu einem Ergebnis kommt.

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	0,02s	0,08s	0,05s	0,17s	0,06s	0,15s
500	0,11s	0,41s	0,21s	0,84s	0,24s	0,92s
1000	0,22s	0,81s	0,40s	1,72s	0,43s	1,81s
2000	0,43s	1,60s	0,80s	3,36s	0,88s	3,69s

Tabelle 6.11: Durchschnittliche Zeit, die zur Zugberechnung benötigt wird. (5 Spieler)

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, R_D)	(D_X, G_D)	(S_X, R_D)	(S_X, G_D)
100	0,02s	0,06s	0,04s	0,13s	0,05s	0,14s
500	0,08s	0,29s	0,17s	0,64s	0,18s	0,69s
1000	0,15s	0,60s	0,33s	1,28s	0,36s	1,40s
2000	0,31s	1,17s	0,64s	2,49s	0,70s	2,70s

Tabelle 6.12: Durchschnittliche Zeit, die zur Zugberechnung benötigt wird. (4 Spieler)

6.5 MCTS MIT 20000 ITERATIONEN

Da die Zeit, der nicht-Random-KIs, deutlich über der der Random-KIs liegt und die Ergebnisse sich bis 2000 Iterationen noch verbessern, wird ein Test mit $MCTS_X(R_X, R_D, 20000)$ ausgeführt.

	Siege	Runden	G Runden	V Runden	Dauer
6 Spieler	166 (175)	14,7	25,1	12,6	1241m 20,8s
5 Spieler	282 (278)	16,2	25,0	12,8	1057m 36,8s
4 Spieler	558 (501)	19,8	25,0	13,2	914m 31.9s

Tabelle 6.13: 1000 Spiele mit $MCTS_X(R_X, R_D, 20000)$ gegen Greedy Bool 1, die Spalten zeigen: Siege Mister X, durchschnittliche Rundenzahl, durchschnittliche Rundenzahl gewonnener Spiele, durchschnittliche Rundenzahl verlorener Spiele und die Testdauer. Die geklammerten Werte sind die Ergebnisse aus $MCTS_X(R_X, R_D, 2000)$

Die Ergebnisse, die in Tabelle 6.13 dargestellt sind, weichen nicht sehr von denen mit 2000 Iterationen ab. Bei 6 Spielern sind sie im Vergleich zu diesen sogar minimal schlechter, was allerdings auch Zufall sein kann. Im Spiel mit 4 Spielern haben sich die Ergebnisse um etwa 6% verbessert, von 50% auf 56%. Warum nur im 4 Spielermodus eine merkliche Verbesserung zu sehen ist, ist jedoch schwer zu erklären.

6.6 MCTS GEGEN MENSCHLICHEN SPIELER

Hier wurden 7 Spiele mit 6 Spielern gegen $MCTS_X(S_X, G_D, 500)$ gespielt, gewonnen haben die menschlichen Spieler davon 6. Beim Spielen zeigten sich einige Schwächen von Mister X: Er verwendet in Situationen, in denen es ihm keinen Vorteil bringt Doppelzüge und Blacktickets, zum Beispiel während der ersten 3 Züge, wo seine Position noch gänzlich unbekannt ist. Oder er verwendet ein Blackticket um das verwendete Ticket nicht preisgeben zu müssen, zeigt sich allerdings während des nächsten Zuges. Erklären lassen sich diese Züge dadurch, dass Mister X diese Züge zunächst wie jeden anderen Zug sieht, und diesen auch genauso behandelt. Wenn das fehlen eines Blacktickets oder Doppelzuges zu Beginn des Spiels zu wenig Einfluss auf die Siegchancen von Mister X haben, kann es sein, dass das Einsetzen eines solchen Tickets das beste Ergebnis liefert und dementsprechend ausgeführt wird. Insofern verhindert die relativ geringe Simulationstiefe ein sinnvolles Einsetzen der Blacktickets und Doppelzüge.

6.7 MCTS GEGEN DIE ANDEREN GREEDY-KIS

Mister X verfügt bei den MCTS Testläufen zur Simulation über die selbe KI die auch von den Detektiven zur Zugberechnung verwendet wird. Das könnte unter Umständen ausgenutzt werden, da so Züge gemacht werden, die von der KI der Detektive nicht erwartet werden. Dadurch könnte Mister X einen Vorteil ziehen. Aus diesem Grund werden auch gegen die anderen Greedy-KIs der Detektive getestet. $MCTS_X(S_X, G_D, It)$ ist nach den Ergebnissen die stärkste KI dementsprechend wird diese KI gegen die drei anderen Greedy-KIs getestet. Die Ergebnisse werden in den Tabellen 6.14 - 6.16 aufgelistet.

6 Spieler	Greedy Bool 2	Greedy Float 1	Greedy Float 2	Greedy Bool 1
100	692	530	544	712
500	687	606	599	712
1000	710	633	633	721
Ohne MCTS	427	487	470	441

Tabelle 6.14: 1000 Spiele mit 6 Spielern, $MCTS_X(S_X, G_D, It)$ gegen andere Detektiv-KIs. Ohne MCTS stellt die erwartete Siegzahl bei 1000 Spielen mit der Safepath-KI als Vergleichswert dar.

Man sieht, dass die Ergebnisse gegen die Float-KIs bei niederer Zahl an Iterationen deutlich schlechter ausfallen, als gegen die Bool-KI. Mit zunehmender Zahl an Iterationen nimmt der Unterschied allerdings wieder ab. Da die beiden Greedy Bool-KIs sehr ähnliche Züge machen, ist es nicht verwunderlich, dass die Ergebnisse hier nur wenig unterschiedlich ausfallen.

5 Spieler	Greedy Bool 2	Greedy Float 1	Greedy Float 2	Greedy Bool 1
100	782	604	583	786
500	808	712	726	841
1000	805	749	750	799
Ohne MCTS	660	707	690	674

Tabelle 6.15: 1000 Spiele mit 5 Spielern, $MCTS_X(S_X, G_D, It)$ gegen andere Detektiv-KIs. Ohne MCTS stellt die erwartete Siegzahl bei 1000 Spielen mit der Safepath-KI als Vergleichswert dar.

4 Spieler	Greedy Bool 2	Greedy Float 1	Greedy Float 2	Greedy Bool 1
100	865	682	674	910
500	907	838	797	899
1000	916	857	853	898
Ohne MCTS	887	896	888	885

Tabelle 6.16: 1000 Spiele mit 4 Spielern, $MCTS_X(S_X, G_D, It)$ gegen andere Detektiv-KIs. Ohne MCTS stellt die erwartete Siegzahl bei 1000 Spielen mit der Safepath-KI als Vergleichswert dar.

Allerdings ist im Spiel mit 4 Spielern die nichtplanende Safepath-KI der MCTS gegen einen unbekannten Gegner (Greedy-Float-1/2) überlegen.

6.8 ZUSAMMENFASSUNG

Gegen 6 Spieler liefert die $MCTS_X(S_X, G_D, It)$-KI gegen die von ihr selbst zur Simulation verwendete KI gute Ergebnisse. Gegen einen menschlichen Spiele hat MCTS allerdings sehr schlecht abgeschnitten. Da hier nur wenige Spiele gespielt werden, kann allerdings nicht sicher gesagt werden, dass es sich nicht um Zufall gehandelt hat. Zu beachten sind auch die Ergebnisse der $MCTS_X(R_X, R_D, It)$ KI, die ohne eine sinnvolle Simulation im Spiel mit 4 Spielern eine Gewinnrate von ca. 50% hat. Durch das Ausschließen von einigen Zügen aus der Planung, könnten die MCTS-KIs möglicherweise noch weiter verbessert werden.

Für die Detektive muss MCTS etwas anders verwendet werden. Da die Detektive nicht wissen, wo sich Mister X befindet, sie also keine perfekten Informationen haben, müssen sie die Position raten. Durch den Belief-State haben die Detektive eine Liste alle möglichen Positionen von Mister X. Den Detektiven steht eine bestimmte Anzahl an MCTS-Iterationen zu. Für jede mögliche Position von Mister X wird nun eine MCTS gestartet. Sobald diese Berechnungen beendet sind, werden die Ergebnisse der einzelnen Züge des aktiven Detektivs addiert und der Zug mit dem besten Ergebnis ausgeführt. Da die Anzahl der Iterationen nicht immer durch die Anzahl der möglichen Positionen von Mister X teilbar ist, werden, um einzelne Positionen nicht bevorzugt zu behandeln, Teilungsreste verworfen. Somit kann die tatsächliche Anzahl der Iterationen etwas geringer ausfallen.

7.1 ERGEBNISSE

Getestet wird gegen die Safepath-KI. Da die Tests sehr lange dauern, wird MCTS nur mit 100 und 500 Iterationen durchgeführt. Die Tabellen 7.1 - 7.3 geben die Anzahl der Siege von Mister X an.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)	Greedy Bool 1
100	992	999	967	664	44,0%
500	964	992	863	370	44,0%

Tabelle 7.1: 1000 Spiele mit 6 Spielern, Gewinne von Mister X, gespielt gegen Safepath, die prozentuale Gewinnrate ohne die Verwendung von MCTS als Vergleichswert

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)	Greedy Bool 1
100	995	999	984	734	67,4%
500	983	994	921	518	67,4%

Tabelle 7.2: 1000 Spiele mit 5 Spielern, Gewinne von Mister X, gespielt gegen Safepath, die prozentuale Gewinnrate ohne die Verwendung von MCTS als Vergleichswert

Es ist deutlich zu erkennen, dass die Detektive mithilfe von MCTS gegen die Safepath-KI meist schlecht abschneiden. Es werden allerdings auch nur höchstens 500 Iterationen für die MCTS durchgeführt. Wie auch im vorangegangenen Kapitel sieht man, dass durch das Nutzen der exakten KI der Gegenspielers relativ gute Ergebnisse er-

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)	Greedy Bool 1
100	996	998	989	840	88,5%
500	993	994	973	736	88,5%

Tabelle 7.3: 1000 Spiele mit 4 Spielern, Gewinne von Mister X, gespielt gegen Safepath, die prozentuale Gewinnrate ohne die Verwendung von MCTS als Vergleichswert

zeugt werden können. Ansonsten sehen die Ergebnisse für die $MCTS_D$ sehr schlecht aus.
Um hierzu noch einen Vergleichswert zu haben, werden die selben Tests auch gegen die Distanz-KI von Mister X durchgeführt. Die Ergebnisse sind in den Tabellen 7.4 - 7.7 aufgeführt.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)	Greedy Bool 1
100	870	986	541	671	14,1%
500	629	867	226	433	14,1%

Tabelle 7.4: 1000 Spiele mit 6 Spielern, Gewinne von Mister X, gespielt gegen die Distanz-KI, die prozentuale Gewinnrate ohne die Verwendung von MCTS als Vergleichswert

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)	Greedy Bool 1
100	895	978	556	712	25,5%
500	755	875	276	494	25,5%

Tabelle 7.5: 1000 Spiele mit 5 Spielern, Gewinne von Mister X, gespielt gegen die Distanz-KI, die prozentuale Gewinnrate ohne die Verwendung von MCTS als Vergleichswert

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)	Greedy Bool 1
100	931	970	604	777	42,8%
500	806	875	376	646	42,8%

Tabelle 7.6: 1000 Spiele mit 4 Spielern, Gewinne von Mister X, gespielt gegen die Distanz-KI, die prozentuale Gewinnrate ohne die Verwendung von MCTS als Vergleichswert

Da die Spielstärke der Distanz-Mister X-KI deutlich unter der der Safepath-KI liegt, sehen hier die Resultate der Detektive auch besser aus, allerdings liegen sie hier fast alle unter der Leistung der nichtplanenden Greedy Bool 1. Wie bereits bei den anderen Tests auch, ist MCTS wieder am effizientesten gegen die KI, die sie selbst auch für die Simulation verwendet. Leider stehen keine Testergebnisse für 1000 und 2000 Iteratio-

nen zur Verfügung, da diese zu lang dauern würden. Somit ist es leider nicht möglich die weitere Abhängigkeit zwischen Iterationen und Gewinnrate zu analysieren.

7.2 DAUER

Da bei den Detektiven für jeden einzelnen Detektiv eine eigenständige MCTS durchgeführt wird, erhöht sich die Dauer der einzelnen Spiele deutlich. (Siehe Tabelle 7.7 - 7.9.)

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	28m 07,0s	134m 07,8s	277m 14,8s	326m 40,0s
500	149m 08,9s	811m 42,5s	1411m 12,9s	1653m 24,5s

Tabelle 7.7: 1000 Spiele mit 6 Spielern, Dauer der Tests, gespielt gegen Safepath

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	19m 34,6s	95m 22,5s	190m 52,0s	229m 53,2s
500	104m 15,3s	563m 35,4s	997m 06,3s	1110m 17,4s

Tabelle 7.8: 1000 Spiele mit 5 Spielern, Dauer der Tests, gespielt gegen Safepath

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	11m 34,8s	58m 50,3s	117m 49,2s	143m 37,8s
500	64m 30,9s	339m 11,2s	622m 43,6s	731m 54,2s

Tabelle 7.9: 1000 Spiele mit 4 Spielern, Dauer der Tests, gespielt gegen Safepath

Aufgrund der Dauer der Tests werden keine Tests mit 1000 MCTS Iterationen durchgeführt. Die Zeiten sind bei Spielen mit 5 Detektiven deutlich höher als mit nur 3. Das hat den einfachen Grund, dass jeder Detektiv in jeder Runde seinen Zug berechnen muss.

7.2.1 *Zeit pro Zug*

Die Berechnung der Zugdauer wird analog zu der in Kapitel 6 durchgeführt.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	23780	23473	23793	22722
500	24002	23890	23897	21259

Tabelle 7.10: 1000 Spiele mit 6 Spielern, Rundenzahl, gespielt gegen Safepath

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	23548	23284	23722	22895
500	24074	23807	24027	21979

Tabelle 7.11: 1000 Spiele mit 5 Spielern, Rundenzahl, gespielt gegen Safepath

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	23110	23139	23455	22903
500	23898	23782	23941	22853

Tabelle 7.12: 1000 Spiele mit 4 Spielern, Rundenzahl, gespielt gegen Safepath

Aus der Rundenzahl, die in den Tabellen 7.10 - 7.12 gelistet sind, ergeben sich die in den Tabellen 7.13 - 7.15 zusammengefassten durchschnittliche Berechnungsdauern für die einzelnen Züge. Da pro Runde Anzahl Spieler - 1 Detektive ziehen, wird die Zeit pro Runde noch durch die Anzahl der Spielenden Detektive geteilt.

6 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	0,01s	0,07s	0,14s	0,17s
500	0,07s	0,41s	0,71s	0,93s

Tabelle 7.13: 1000 Spiele mit 6 Spielern, Zugdauer, gespielt gegen Safepath

5 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	0,01s	0,06s	0,12s	0,15s
500	0,06s	0,36s	0,64s	0,76s

Tabelle 7.14: 1000 Spiele mit 5 Spielern, Zugdauer, gespielt gegen Safepath

Die einzelnen Züge sind relativ schnell berechnet, da auch nur 500 Iterationen pro Zug gemacht werden. Die einzelnen Tests dauern entsprechend länger, da pro Runde 3 bis 5 Detektivzüge berechnet werden müssen.

4 Spieler	(R_X, R_D)	(R_X, G_D)	(D_X, G_D)	(S_X, G_D)
100	0,01s	0,05s	0,10s	0,13s
500	0,05s	0,29s	0,52s	0,64s

Tabelle 7.15: 1000 Spiele mit 4 Spielern, Zugdauer, gespielt gegen Safepath

7.3 MCTS DETEKTIVE GEGEN MCTS MISTER X

Für MCTS Detektive gegen MCTS Mister X werden nur zwei Test vorgestellt. Bei beiden Tests sind $MCTS_X(S_X, G_D, 500)$ gegen $MCTS_D(S_X, G_D, 500)$ bei 4 und 6 Spielern. Diese Tests werden statt 1000 nur 100 mal durchgeführt. Die Ergebnisse finden sich in der Tabelle 7.16.

	Siege	Runden	G Runden	V Runden	Dauer
6 Spieler	33 (44%)	16,4 (19,0)	24,6 (24,8)	12,3 (14,4)	157m 27,2s
5 Spieler	47 (67%)	17,7 (21,6)	24,4 (24,8)	11,8 (14,9)	124m 55,9s
4 Spieler	75 (88%)	20,6 (23,4)	23,8 (24,5)	11,0 (14,7)	88m 01,5s

Tabelle 7.16: 100 Spiele mit 6 Spielern, die Spalten zeigen: Siege Mister X, Durchschnittliche Rundenzahl, durchschnittliche Rundenzahl gewonnener Spiele, durchschnittliche Rundenzahl verlorener Spiele und die Testdauer. Die Werte in Klammern zeigen die Werte ohne MCTS aus Kapitel 4.

Da nur 100 Durchläufe gemacht werden, kann das Ergebnis eine hohe Abweichung von den tatsächlichen Wahrscheinlichkeiten haben. Dennoch sind die Ergebnisse vor allem für die Detektive gut. Die beiden Ergebnisse zeigen jeweils bessere Werte als beim beiderseitigen Verzicht auf MCTS.

7.4 ZUSAMMENFASSUNG

Für die Detektive scheint die implementierte Version der MCTS nur sinnvoll gegen diejenigen KIs zu sein, die sie selbst verwendet. Das Problem ist einfach, dass für jeden potentiellen Standort von Mister X eine eigenständige MCTS durchgeführt wird. Das resultiert in einer geringen Tiefe des Baums und erklärt damit das schlechte Abschneiden. Um das zu verbessern, könnte statt wie beim Bool Belief wieder ein Float Belief eingesetzt werden (das hier nicht getestet wird). Anstatt die Zugwahrscheinlichkeit für jede Verbindung gleich anzusehen, könnte diese für Felder, die weiter von den Detektiven weg liegen und schwerer von ihnen zu erreichen sind, höher eingestuft werden.

8. ZUSAMMENFASSUNG UND AUSBLICK

8.1 ZUSAMMENFASSUNG

Die in der vorangegangenen Arbeit von Minorics entwickelte Safepath-KI für Mister X hat je nach Spieleranzahl eine Gewinnchance von ca. 40% bis hin zu 90%. Durch das Einsetzen von MCTS ist es möglich, die Spielstärke der einzelnen nichtplanenden KIs mitunter deutlich zu erhöhen. Voraussetzung hierfür ist allerdings, dass gute Simulations-KIs zur Verfügung stehen. Wenn für MCTS keinerlei KI vorhanden ist, sondern nur zufällige Züge gezogen werden, liefert MCTS allerdings immer noch Ergebnisse, die durchaus realistische Chancen auf einen Sieg haben.
Es ist allerdings auch möglich, die Spielstärke der KI durch das Einsetzen von MCTS zu reduzieren, wenn die Simulation-KI-Kombination ungünstig gewählt wird. Die besten Ergebnisse liefert MCTS, wenn die generische KI Teil der Simulation ist, da so mögliche Schwächen ausgenutzt werden können.

8.2 AUSBLICK

Es ist möglich, die Spielstärke der MCTS weiter zu verbessern, indem unsinnige Züge nicht zur Berechnung herangezogen werden. Zum Beispiel könnte in die MCTS ein Filter geschaltet werden, der sämtliche "Verschleierungs-Blacktickets" die einen Zug eingesetzt werden, bevor Mister X sich zeigt, aus der Liste der möglichen Knoten entfernt. Das würde die Anzahl der möglichen Züge reduzieren und somit mehr Ressourcen für sinnvollere Züge lassen.
Eine weitere Verbesserungsmöglichkeit wäre eine Kombination von verschiedenen KIs. Nach einem Zufallsverfahren wird für jede Simulation (oder sogar für jeden Zug) eine KI ausgewählt. Dadurch werden mehrere mögliche Strategien des Gegners berücksichtigt und somit Schwachstellen der verwendeten KI ausgeglichen.
Bisher wird, sobald ein Zug gefunden wird, der gesamte Baum verworfen und beim nächsten Zug ein neuer aufgebaut. Es ist möglich nach einem Zug nur den Teil des Baums zu verwerfen, der nicht eingetreten ist und den Rest zu behalten. Das würde entweder das Finden eines Zuges beschleunigen, oder könnte die Qualität des Zuges verbessern.
Im bestehenden Programm verwenden alle Detektive grundsätzlich die selbe KI. Möglicherweise könnte das Einsetzen von unterschiedlicher KIs für die einzelnen Detektive ihre kollektive Spielstärke erhöhen.
In Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard [2] wird vorgeschlagen, dass die Detektive für die Bewertung der Züge nicht nur 0 (für die Nieder-

lage) und 1 (für den Sieg) einsetzen, sondern auch einen Weiteren. Dieser soll genutzt werden, wenn ein anderer Detektiv Mister X fängt. Das erhöht das Interesse daran Mister X selbst zu fangen, wodurch sie versuchen mehr zum Spiel beizutragen. Hier könnten verschiedene Werte getestet und ausprobiert werden.
Aktuell wird eine feste Anzahl an Iterationen für die MCTS verwendet. Eine weitere Variante wäre das Vorgeben einer bestimmten Zeit, die MCTS pro Zug zur Verwendung steht. Insgesamt gibt es also noch zahlreiche Ideen, die Performance der im Rahmen der Arbeit entwickelten KIs weiterzuentwickeln.

LITERATUR

[1] G. R. Minorics, "Digital Escape – ein KI-basierter Algorithmus zur Steuerung von Mr. X im Brettspiel Scotland Yard", Studienarbeit, Institut für Visualisierung und Interaktive Systeme , Universität Stuttgart, 2013.

[2] P. Nijssen und M. Winands, "Monte Carlo tree search for the hide-and-seek game Scotland Yard", *IEEE Transactions on Computational Intelligence and AI in Games*, Bd. 4, Nr. 4, S. 282–294, 2012.

[3] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search", in *In: Proc. Computers and Games*, Springer-Verlag, 2006.

[4] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis und S. Colton, "A survey of Monte Carlo tree search methods", *IEEE Transactions on Computational Intelligence and AI in Games*, Bd. 4, Nr. 1, S. 1–43, 2012.

[5] Otto Maier Verlag Ravensburger, *Scotland Yard*, 1983.

[6] Mciura. (2013), Adresse: http://commons.wikimedia.org/wiki/File:MCTS_(English).svg.

[7] R. Basbous und B. Nagy, "Generalized game trees and their evaluation", in *proc. IEEE Conference on Cognitive Infocommunications*, 2014, S. 55–60.

[8] N. Nogueira. (2006), Adresse: http://en.wikipedia.org/wiki/File:Minimax.svg.

[9] H. Baier und M. Winands, "MCTS-MiniMax Hybrids", *IEEE Transactions on Computational Intelligence and AI in Games*, Nr. 99, S. 1–14, 2015.

A. ANHANG

A.1 DAS PROGRAMM

Das Programm ist in C++11 geschrieben. Für die graphische Ausgabe kann QT5 mit OpenGL verwendet werden, die eventuelle Eingabe von Zügen erfolgt aber in jedem Fall über eine Konsole.
Zum Kompilieren der grafischen Version kann einfach "qmake && make" aufgerufen werden. Für die beiden andere Versionen steht für Linux ein einfaches Skript zur Verfügung. (Diese Versionen benötigen den Header sys/time.h zur Initialisierung der Zufallsgenerators, der nicht überall verfügbar ist.) Die beiden Versionen SY_noQT und SY_noQT_comp unterscheiden sich nur darin, dass bei SY_noQT_comp die Startpositionen für das i-te Spiel immer gleich sind, während sie bei SY_noQT immer zufällig sind. Wenn es gewünscht ist, kann der Parameter -D AUSGABELEVEL=0 angepasst werden. 0 Bedeutet, dass das Programm während des Spiels keine Ausgaben macht. 1: das Programm macht wichtige Ausgaben, 2: Das Programm mach einige Ausgaben über offene Berechnungen, 4: eigentlich verborgene Informationen werden offenbart, 8: Berechnungen über verborgene Informationen werden ausgegeben. Die Werte von allen Ausgaben, die man sehen will, müssen aufaddiert werden und dem AUSGABELEVEL übergeben werden (15: Alle Informationen ausgeben). Idealerweise sollte das Programm mit dem Optimierungslevel 2 kompiliert werden, Optimierungslevel 3 hat bei Tests zu längeren Ausführungszeiten geführt.
Der Aufruf der grafischen Version benötigt die Übergabe einer Konfigurationsdatei. Die Konfigurationsdatei ist wie folgt aufgebaut: In jeder Zeile steht nur ein Parameter, eine Zahl oder ein Dateipfad, die in dieser Reihenfolge angegeben werden müssen:

1. <Anzahl Spieler>
2. <Graphdatei>
3. <Bilddatei>
4. <Koodrinatendatei>
5. <Mister X Modus>
6. <Mister X KI Modus>
7. <MC_X_It>
8. <MC_X_D_Modus>
9. <MC_X_X_Modus>
10. <MC_X_X_KI_Modus>
11. <MC_X_float_belief_state>
12. <MC_X_VergleichFunktion>
13. <Detektive Modus>

14. <float_belief_state>
15. <VergleichFunktion>
16. <MC_D_It>
17. <MC_D_D_Modus>
18. <MC_D_X_Modus>
19. <MC_D_X_KI_Modus>
20. <MC_D_float_belief_state>
21. <MC_D_VergleichFunktion>
22. <closed>

Die nichtgrafische Versionen erwarten die Übergabe vieler Parameter. Es sind die gleichen Parameter in der selben Reihenfolge wie bei der grafischen Version nur dass auf <Anzahl Spieler> <Anzahl Iterationen> folgt und <Bilddatei> <Koodrinatendatei> und <closed> wegfallen.

A.1.1 *Bedeutungen der Parameter*

- Anzahl Spieler: Anzahl der Spieler die am Spiel teilnehmen inklusive Mister X (erlaubte Werte 2 bis 6)
- Graphdatei: Datei mit allen Verbindungen des Graphen
- Bilddatei: Bild des Spielplans als ppm
- Koordinatendatei: Datei mit den Positionen aller Spielfelder
- Mister X Modus: Mister X spielt als 0: Mensch, 1: Random, 2: Greedy, 3: MCTS
- Mister X KI Modus: Art des Greedy Modus; 0: Distanzbasiert, 3: Safepath
- Detektive Modus: Detektive spielen als 0: Mensch, 1: Random, 2: Greedy, 3: MCTS
- float_belief_state: 0: Bool Belief, 1: Float Belief
- VergleichFunktion: 0: Greedy 1, 1: Greedy 2
- MC_X/D_It: Iterationen die die MCTS bei jedem Zug für Mister X bzw. die Detektive durchführt
- MC_X/D_*: Einstellungen für die MCTS-Simulation
- closed: 0: Mister X immer sichtbar, 1: Mister X nur zu den zu zeigenden Zügen sichtbar

ERKLÄRUNG

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift