HAW HAMBURG

Hausarbeit

Niklas Hoefflin

Vergleich von informierten Suchalgorithmen unter Anwendung von Heuristiken am Beispiel des 15-Puzzle

Modul: Intelligente Systeme

Betreuung durch: Prof. Dr. Peer Stelldinger
Eingereicht am: 10. März 2023

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung. Diese Hausarbeit beschäftigt sich mit dem 15-Puzzle und dem Finden optimaler Lösungen durch informierte Suchalgorithmen wie dem A*-Algorithmus und dem IDA*-Algorithmus. Zudem werden verschiedene Strategien, sogenannte Heuristiken, eingeführt, welche in den Suchalgorithmen angewendet werden, um die Suchzeit zum Lösen vom 15-Puzzle zu reduzieren. Abschließend werden die verwendeten Suchalgorithmen unter Anwendung der Heuristiken auf der Basis von Laufzeit- und Speichereffizienz untereinander verglichen. Die Ergebnisse der Analysen haben ergeben, dass der A* aufgrund des hohen Speicherverbrauchs ungeeignet ist und sich der IDA* für das konkrete Problem des 15-Puzzle besser eignet, wobei zu berücksichtigen ist, dass dieser eine deutlich höhere Laufzeit hat. Außerdem geht hervor, dass in beiden Algorithmen die Lineare-Konflikte-Heuristik gegenüber der Manhattan-Distanz und der Hamming-Distanz ab einer bestimmten Suchtiefe deutlich schneller eine Lösung berechnen konnte und dabei im Fall des A* weniger Speicherplatz benötigte.

Schlüsselwörter: Intelligente Systeme, Informierte Suchalgorithmen, Heuristiken, Künstliche Intelligenz, 15-Puzzle

1 Einleitung

Eines der am weitverbreitetsten Testprobleme für uninformierte und informierte Suchalgorithmen in der künstlichen Intelligenz ist das 15-Puzzle. Es ist der Kategorie der Geduldsspiele zuzuordnen und wurde zwischen den Jahren 1870-1880 vom US-Amerikaner Noyes Palmer Chapman erfunden[8]. Das 15-Puzzle besteht aus einem Spielfeld von 4×4 Feldern, wobei diesem Spielfeld 15 Spielsteine mit den Zahlen von $1 - 15$ zugeordnet sind. Ein Feld enthält keinen Spielstein und bleibt somit leer. Durch das Verschieben der Spielsteine in das leere Feld kann ein Startzustand in einen Zielzustand überführt werden. Ziel des Spiels ist es, die vorher gemischten Spielsteine wieder an ihren ursprünglich festgelegten Zielzustand zu bringen. Für jeden gültigen Startzustand kann man die optimale Lösung, also die Mindestanzahl an Zügen zu ihrem gültigen Zielzustand, in maximal 80 Zügen, berechnen[1].

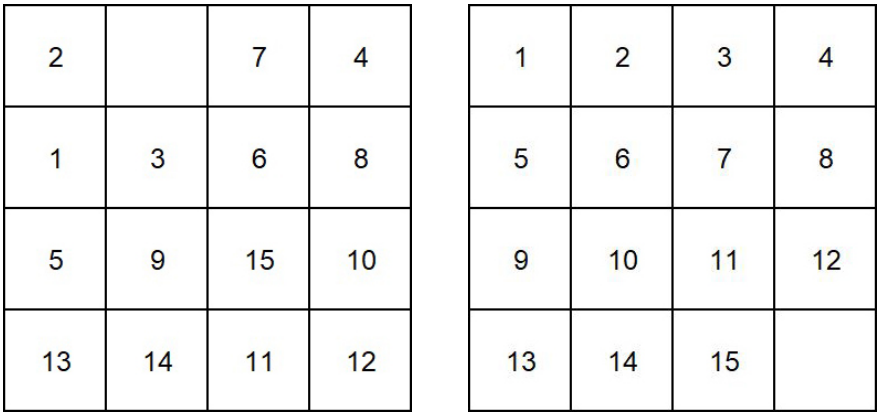

Abbildung 1: Mögliches ungelöstes 15-Puzzle(links), Zielzustand(rechts).

Diese Hausarbeit wird sich zuerst mit der generellen Thematik des 15-Puzzles befassen, bevor die in dieser Arbeit verwendeten informierten Suchalgorithmen wie der A*-Algorithmus (A Stern) und der IDA*-Algorithmus (iterative deepening A*), vorgestellt werden. Im Anschluss an die Suchalgorithmen werden auf die in dieser Arbeit verwendeten Heuristiken Bezug genommen. Es werden insgesamt drei verschiedene Heuristiken verwendet: Die Manhattan-Distanz, die Hamming-Distanz und die Lineare-Konflikte-Heuristik. Abschließend werden die Suchalgorithmen unter Anwendung der Heuristiken beim Lösen von 15-Puzzles auf ihr Verhalten untersucht und untereinander auf die Laufzeit- (Zeit zum Lösen) und Speichereffizienz (Menge erweiterter Spielfelder) verglichen.

1.1 Komplexitätsklasse

Das 15-Puzzle und alle anderen Varianten der n-Puzzle gehören zu der Klasse der NP-vollständigen Probleme, dass heißt, sie sind mindestens so schwer zu lösen wie jedes andere Problem das in NP liegt[7]. Unter der Annahme, dass $P \neq NP$ gilt, so existiert kein Algorithmus, der die kürzeste Lösung für ein beliebiges n-Puzzle in Polynomialzeit lösen kann. Das Überprüfen der Lösbarkeit eines n-Puzzle liegt in der Komplexitätsklasse P[6], wohingegen das Finden einer kürzesten Lösung für ein beliebiges n-Puzzle NP-schwer ist[7].

1.2 Lösbarkeit

Um ein beliebiges 15-Puzzle von einem Startzustand in den Zielzustand überführen zu können, muss die Permutation (beliebige Anordnung von Spielsteinen) des Puzzles lösbar sein. Um zu überprüfen, ob ein Puzzle lösbar ist, muss die Parität der Gesamtsumme $N_{sum} = N_1 + N_2$ gerade sein[9], andernfalls ist ein Puzzle unlösbar. Die Gesamtsumme setzt sich aus der Summe der ungeordneten Zahlenpaare der Permutation N_1 und der Zeilennummer, an der sich das leere Feld befindet N_2 zusammen. Um die Anzahl der ungeordneten Zahlenpaare N_1 in einer Permutation zu bestimmen, werden ausgehend von der aktuellen Zahl alle nachfolgenden Zahlenpaare gezählt, bei denen die nachfolgende Zahl kleiner als die aktuelle ist. Dies wird für alle Zahlen in Leserichtung wiederholt, bis das letzte Feld unten rechts erreicht ist, wobei das leere Feld ignoriert wird. Um die Zeilennummer, in der sich das leere Feld befindet N_2 zu ermitteln, wird von der untersten Zeile von 0 angefangen zu zählen. Der Wert wird in jeder Zeile um 1 erhöht, bis die Zeile erreicht wird, welche das leere Feld enthält.
Im Fall von Abbildung[1] links lauten die Zahlenpaare (2|1), (7|4), (7|1), (7|3), (7|6), (7|5), (4|1), (4|3), (6|5), (8|5), (15|10), (15|13), (15|14), (15|11), (15|12), (13|11), (13|12), (14|11) und (14|12). Somit beträgt die Anzahl der ungeordneten Zahlenpaare $N_1 = 19$ und der Wert des leeren Feldes beträgt $N_2 = 3$. Da die Summe $N_{sum} = N_1 + N_2 = 19 + 3 = 22$ gerade ist, folgt daraus, dass die Anordnung der Abbildung[1] links lösbar ist.

2 Algorithmen

Die in dieser Hausarbeit verwendeten informierten Suchalgorithmen sind Algorithmen, die während der Suche nach einer Lösung für ein beliebiges 15-Puzzle Informationen über den aktuellen Spielzustand berücksichtigen, um den Suchraum begrenzen zu können, um dann ein Puzzle schneller zu lösen. Beim 15-Puzzle werden von den Algorithmen Suchbäume generiert, welche Knoten enthalten, die eine beliebige Permutation von Spielsteinen eines Spielzustands repräsentieren. Diese werden dann während der Laufzeit von den Algorithmen erweitert, um daraus weitere zielführende Permutationen zu generieren, die am Ende dann mit dem Zielzustand übereinstimmen.

2.1 A*-Algorithmus

Der A*-Algorithmus (A*) ist ein informierter Suchalgorithmus, der den kürzesten Pfad zwischen einem Start- und einem Zielknoten findet. Er erreicht dies, indem er einen Suchbaum vom Startknoten aus aufbaut und sich immer um einen Knoten erweitert, bis der Zielknoten gefunden wurde. Durch die Eigenschaft, dass der A* alle bekannten Knoten speichert, ist der Speicherplatzverbauch sehr hoch und für die meisten Probleme ungeeignet. Um den Knoten mit den niedrigsten Kosten zu finden, der als Nächstes untersucht werden soll, werden alle bereits untersuchten Knoten mit einem f-Wert gekennzeichnet. Dieser steht für die Gesamtkosten des Knotens und dieser schätzt ab, wie lang der Pfad vom Start- zum Zielknoten unter der Berücksichtigung des betrachteten Knotens im günstigsten Fall ist. Der f-Wert eines Knotens wird durch die Funktion $f(n) = g(n) + h(n)$ berechnet, wobei $g(n)$ die Kosten vom Startknoten bis zum Knoten n und $h(n)$ die geschätzten Kosten der Heuristik von n bis zum Zielknoten sind. Der A*-Algorithmus garantiert[4] den kürzesten Pfad zum Zielknoten, wenn die Heuristik zulässig ist (siehe Abschnitt 3.1).

2.2 IDA*-Algorithmus

Der IDA*-Algorithmus ist wie der A*-Algorithmus ein informierter Suchalgorithmus, der den kürzesten Pfad zwischen einem Start- und Zielknoten findet. Der Algorithmus ist eine Kombination aus einer iterativen Tiefensuche, die sich durch einen geringen Speicherverbrauch auszeichnet und einer Variante der Breitensuche wie bei dem A*. Er führt bei jeder Iteration eine Tiefensuche durch, welche für einen Pfad abgebrochen wird, wenn die Gesamtkosten eines Knotens $f(n) = g(n)+h(n)$ die obere Grenze überschreitet. Die obere Grenze wird am Anfang durch den h-Wert der Heuristik festgelegt. Nach jeder Iteration wird die obere Grenze angepasst. Sie erhält den minimalen f-Wert von allen Knoten, welche die Grenze überschritten haben. Der Algorithmus terminiert, wenn der Zielzustand gefunden wurde, andernfalls kommt es zu einer Endlosschleife, welche man durch das Setzen einer maximalen Grenze verhindern kann. Der IDA*-Algorithmus garantiert[5] den kürzesten Pfad zum Zielknoten, wenn die Heuristik wie bei dem A*-Algorithmus zulässig ist (siehe Abschnitt 3.1).

3 Heuristiken

Die Heuristiken, die in dieser Arbeit verwendet werden, haben die Aufgabe, Informationen darüber zu liefern, wie gut ein momentaner Spielzustand eines 15-Puzzles ist. Dies wird durch einen h-Wert ermittelt, welcher durch die Heuristik berechnet wird und angibt, wie nah ein Spielzustand dem Zielzustand ist. Durch diese Informationen wird der Suchraum begrenzt, wodurch eine Lösung schneller und speichereffizienter gefunden wird, solange die Heuristik zulässig ist. Alle in dieser Hausarbeit verwendeten Heuristiken sind zulässig (siehe Abschnitt 3.1).

3.1 Zulässigkeit

Eine Heuristik h ist zulässig, wenn für jeden Knoten n gilt, dass $h(n) \leq h^*(n)$ erfüllt ist, wobei $h(n)$ die aktuellen Kosten vom Knoten n zum Zielknoten und $h^*(n)$ die optimalen Kosten vom Knoten n zum Zielknoten sind. Das heißt, die Kosten um das Ziel zu erreichen, werden niemals überschätzt, welches wiederum bedeutet, dass die Kosten in jeden Knoten niemals höher sein dürfen als die niedrigsten möglichen Kosten vom aktuellen Knoten zum Ziel.

3.2 Manhattan-Distanz

Die Manhattan-Distanz (MD) ist eine Heuristik, bei der von allen Spielsteinen die Summe aus der vertikalen und der horizontalen Distanz zu ihrer Zielposition berechnet und anschließend für alle Spielsteine aufsummiert wird. Wobei das leere Feld ohne Spielstein ignoriert wird. Die Distanz von einem Spielstein n zu der Zielposition wird durch die Funktion $d(n) = |x_1 - x_2| + |y_1 - y_2|$ berechnet, wobei x_1 und y_1 für die aktuelle Position und x_2 und y_2 für die Zielposition steht.
Am Beispiel von Abbildung[1] links beträgt die Gesamtsumme MD_{sum} für alle Spielsteine im aktuellen Spielzustand $MD_{sum} = n_1(2) + n_2(0) + n_3(7) + n_4(4) + n_5(1) + n_6(3) + n_7(6) + n_8(8) + n_9(5) + n_{10}(9) + n_{11}(15) + n_{12}(10) + n_{13}(13) + n_{14}(14) + n_{15}(11) + n_{16}(12) = 1 + 0 + 1 + 0 + 1 + 2 + 1 + 0 + 1 + 1 + 1 + 2 + 0 + 0 + 1 + 1 = 13$.

3.3 Hamming-Distanz

Die Hamming-Distanz (HD) oder auch bekannt unter dem Namen Misplaced Tiles ist eine Heuristik, bei der von allen Spielsteinen die Summe der Spielsteine berechnet wird, bei denen die aktuelle Spielposition von der Zielposition abweicht, wobei wie bei der MD das leere Feld ignoriert wird. Wenn ein Spielstein von seiner Zielposition abweicht, wird die Gesamtsumme um 1 erhöht.
Am Beispiel von Abbildung[1] links beträgt die Gesamtsumme HD_{sum} für alle Spielsteine im aktuellen Spielzustand $HD_{sum} = n_1(2) + n_2(0) + n_3(7) + n_4(4) + n_5(1) + n_6(3) + n_7(6) + n_8(8) + n_9(5) + n_{10}(9) + n_{11}(15) + n_{12}(10) + n_{13}(13) + n_{14}(14) + n_{15}(11) + n_{16}(12) = 1 + 0 + 1 + 0 + 1 + 1 + 1 + 0 + 1 + 1 + 1 + 1 + 0 + 0 + 1 + 1 = 11$.

3.4 Lineare-Konflikte

Die Lineare-Konflikte-Heuristik (LC) ist eine Verbesserung der MD und wurde erstmals 1992 beschrieben[3]. Die LC ist eine Kombination aus der MD und der Anzahl von linearen Konflikten in einem Spielzustand. Ein linearer Konflikt tritt auf, wenn zwei Spielsteine n_i und n_j in der gleichen Zeile oder Spalte ihre Zielposition besitzen, wobei der Spielstein n_i rechts von n_j positioniert ist und bei dem die Zielposition von n_i links von n_j und die Zielposition von n_j rechts von n_i ist. Die MD dient im Fall der LC als untere Grenze, welche angibt, wie viele Züge nötig sind, um einen beliebigen Spielstein an ihre Zielposition zu bewegen. Sie berücksichtigt aber nicht, dass in einem n-Puzzle im Falle eines linearen Konflikts die Spielsteine nicht übereinander geschoben werden können, wodurch pro auftretenden linearen Konflikt zwei Züge auf die MD addiert werden müssen, da ein Spielstein zunächst zur Seite und danach wieder an die ursprüngliche Stelle geschoben werden muss, damit der andere Spielstein passieren kann.
Am Beispiel von Abbildung[1] links beträgt die Gesamtsumme der Heuristik $LC_{sum} = C_{sum} + MD_{sum} = 2 + 13 = 15$. Die Spielsteine $n_{11}(15)$ und $n_{15}(11)$ stehen in einem linearen Konflikt zueinander und somit beträgt die Summe der linearen Konflikte $C_{sum} = 2$, wobei die Summe der $MD_{sum} = 13$ unverändert bleibt.

4 Analyse

Im Vergleich der Suchalgorithmen geht es darum zu ermitteln, welcher Suchalgorithmus unter Anwendung der Heuristiken im Bezug auf die Zeit zum Lösen eines Puzzles und der Anzahl von erweiterten Spielfeldern effizienter ist. Darauf basierend können Rückschlüsse gezogen werden, welcher Suchalgorithmus besser zum Lösen der 15-Puzzle geeignet ist. Zur Messung der Laufzeit wurde die Java-Programmbibliothek Guava[2] in der Version 31.1 verwendet, da es unter Verwendung der von Java standardmäßig bereitgestellten Messmethoden zu Ungenauigkeiten während der Messvorgänge kam.

4.1 Aufbau

Bei dem Aufbau der Analysen war es wichtig, aussagekräftige Ergebnisse im Bezug auf den Vergleich der Suchalgorithmen zu erhalten. Aus diesem Grund wurden pro Suchtiefe jeweils zehn unterschiedliche 15-Puzzle generiert, die bei jeder Analyse gleich waren und bei der die Suchtiefe, also die minimale Anzahl benötigter Züge im Voraus berechnet wurde. Die Analysen wurden bei beiden Algorithmen unter Anwendung von jeder Heuristik jeweils 100 Mal wiederholt und anschließend das arithmetische Mittel von den Laufzeiten und den erweiterten Spielfeldern berechnet, um präzisere Ergebnisse zu erhalten. Hierbei ist zu beachten, dass abhängig von dem Algorithmus und der Heuristik die Analyse bei dem Erreichen des auf dem Testsystems maximal verfügbaren Speicherplatzes oder eines bestimmten Zeitintervalls, in der keine Lösung gefunden wurde, die aktuelle Messung manuell abgebrochen wurde.

4.2 A* - Vergleich Heuristiken

Im Laufzeitvergleich (siehe Abbildung[2]) der Heuristiken des A* ist zu beobachten, dass bis zu einer Tiefe von 13 sich die Heuristiken relativ gleich verhalten, wobei der MD hier am effizientesten ist. Erst ab einer Tiefe von 14 sieht man, dass der A* mit der HD deutlich mehr Zeit benötigt als mit den anderen Heuristiken. Der A* in Kombination mit der HD findet für die in der Analyse verwendeten 15-Puzzle nur eine Lösung bis zu einer Suchtiefe von 25, bevor der Algorithmus durch den Speicherplatzverbrauch in der Testumgebung abbricht. Bei einer Tiefe von 25 zeigt sich, dass der A* mit der HD ca. das 80-Fache mehr an Zeit benötigt, um die 15-Puzzle zu lösen, als mit den beiden anderen Heuristiken. Im Vergleich zwischen dem A* mit der MD und mit der LC ist zu

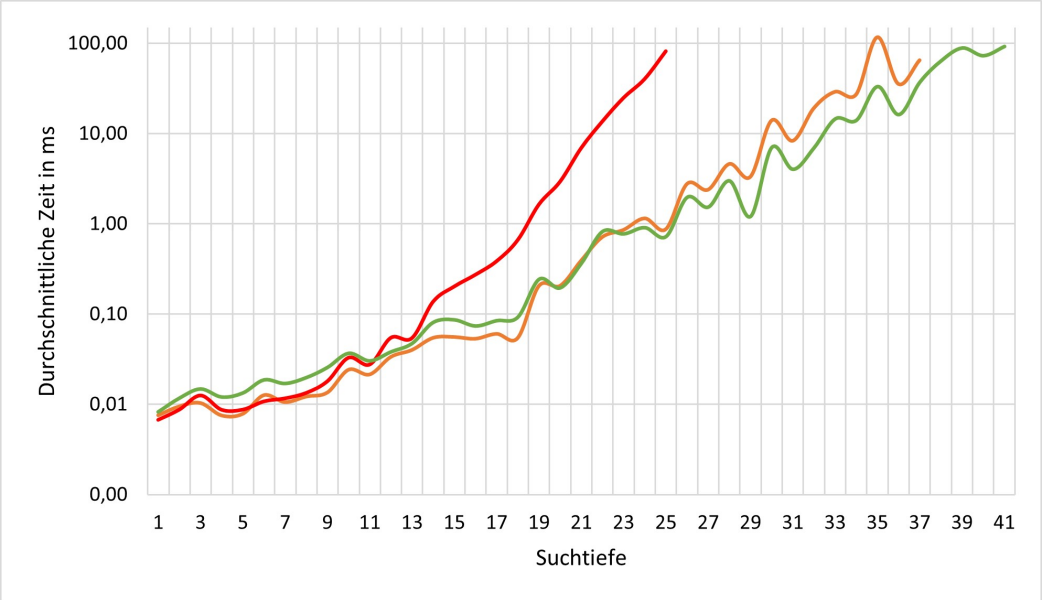

Abbildung 2: Vergleich der Laufzeiten pro Suchtiefe der drei Heuristiken im A*-Algorithmus. HD(rot), MD(orange), LC(grün)

beobachten, dass ab einer Tiefe von 23 der A* mit der LC schneller wird und der A* mit der MD aufgrund des Speicherplatzverbrauchs nur eine Lösung bis zu einer Tiefe von 37 findet. Den Unterschied der benötigten Zeit zum Lösen eines Puzzles zwischen den beiden Heuristiken kann man klar erkennen, wobei der Faktor von den verwendeten 15-Puzzles abhängt. Der A* mit der LC findet im Gegensatz zum A* mit der MD noch bis zu einer Tiefe von 41 Lösungen für die in der Analyse verwendeten 15-Puzzles. Im erweiterten Spielfeldervergleich pro Suchtiefe (siehe Abbildung[3]) ist zu beobachten, dass der A* mit der HD ab einer Tiefe von 9 deutlich mehr Spielfelder erweitert. Wie bei dem Laufzeitvergleich bricht der A* mit der HD bei einer Tiefe von 25 ab. Hier wurden bei einer Tiefe von 25 bereits über 159.000 Spielfelder erweitert, wobei das bei dem A* mit der MD ca. dem 55-Fachen und bei dem A* mit der LC ca. dem 102-Fachen entspricht. Darüber hinaus ist zu beobachten, dass der A* mit der LC generell immer weniger Spielfelder als der A* mit der MD erweitert, besonders deutlich zu erkennen ist dies ab einer Tiefe von 19. Dies ist darauf zurückzuführen, dass die LC auf der MD basiert und diese als untere Grenze benutzt und zusätzlich die linearen Konflikte hinzufügt, was zu einer besseren Schätzung des momentanen Spielzustands führt. Im Vergleich der Heuristiken im A* ist zu beobachten, dass durch die geringere Laufzeitkomplexität die MD in den niedrigeren Suchtiefen der LC überlegen ist, was sich aber mit steigender Tiefe und Komplexität der Puzzles ändert.

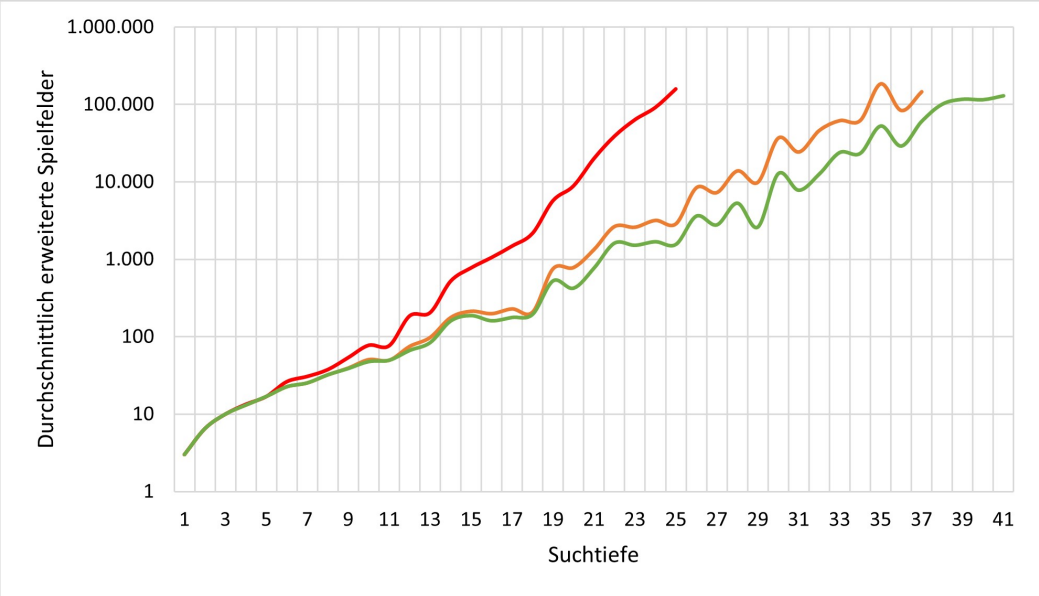

Abbildung 3: Vergleich der erweiterten Spielfelder pro Suchtiefe der drei Heuristiken im A*-Algorithmus. HD(rot), MD(orange), LC(grün)

4.3 IDA* - Vergleich Heuristiken

Im Laufzeitvergleich (siehe Abbildung[4]) der Heuristiken im IDA* ist zu beobachten, dass sich der IDA* mit den Heuristiken bis zu einer Tiefe von 11 relativ gleich verhält. Ab einer Tiefe von 12 braucht der IDA* mit der HD deutlich mehr Zeit zum Lösen der Puzzles. Da der IDA* nicht Speicherplatz beschränkt ist, findet der IDA* auch über Tiefen von 25 Lösungen für die in der Analyse verwendeten Puzzles. Bei einer Tiefe von 36 dauerte es im Durchschnitt 2,8 Minuten, um eine Lösung mit der HD zu berechnen, weshalb hier der Algorithmus manuell abgebrochen wurde. Im Vergleich zu den anderen Heuristiken beträgt die Zeit zum Lösen der Puzzles bei einer Tiefe von 36 mit der HD ca. dem 2.400-Fachen der MD und dem ca. 5.200-Fachen der LC. Zwischen dem IDA* mit der MD und mit der LC ist ab einer Tiefe von 25 zu beobachten, dass der IDA* mit der LC schneller wird. Der Unterschied der Laufzeiten ist deutlich erkennbar, wobei bei beiden Heuristiken der Algorithmus bei einer Tiefe von 54 manuell abgebrochen wurde, da nach über 5 Minuten immer noch keine Lösungen für die in der Analyse verwendeten Puzzles gefunden wurde.

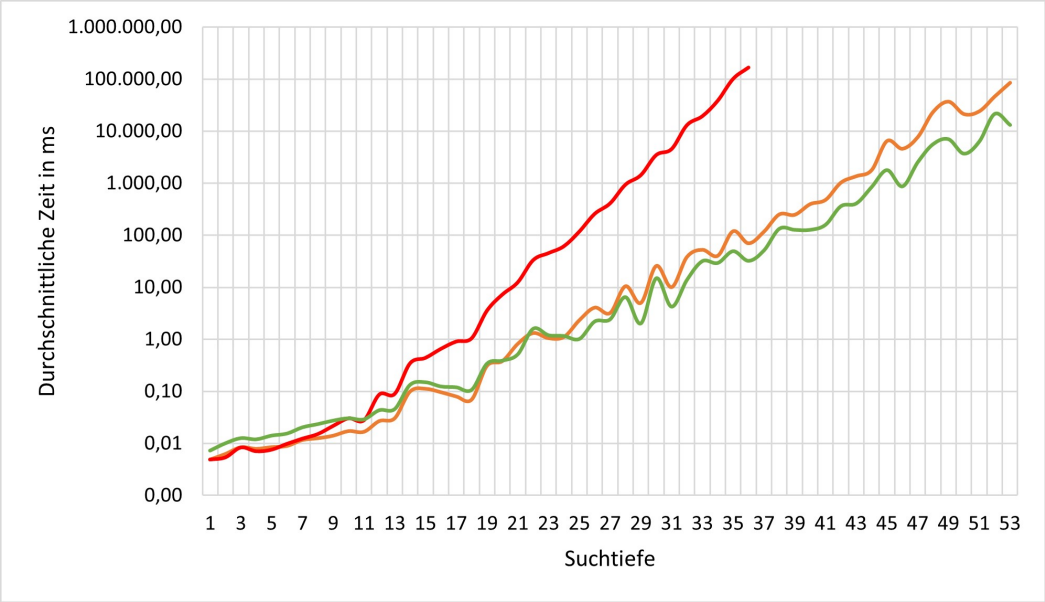

Abbildung 4: Vergleich der Laufzeiten pro Suchtiefe der drei Heuristiken im IDA*-Algorithmus. HD(rot), MD(orange), LC(grün)

Im erweiterten Spielfeldervergleich (siehe Abbildung[5]) pro Suchtiefe im IDA* ist zu beobachten, dass der IDA* mit der HD ab einer Suchtiefe von 6 deutlich mehr Spielfelder erweitert als der IDA* mit den anderen Heuristiken. Wie im Vergleich der Laufzeit im IDA* mit der HD wurde der Algorithmus bei einer Tiefe von 36 abgebrochen und hat bei einer Tiefe von 36 bereits über 502 Millionen Spielfelder erweitert. Im Vergleich zu den beiden anderen Heuristiken sind das bei dem IDA* mit der MD ca. das 2.900-Fache und bei dem IDA* mit der LC ca. das 11.500-Fache. Wie beim A* lässt sich im Vergleich der erweiterten Spielfelder beim IDA* mit der MD und der LC deutlich beobachten, dass der IDA* mit der LC mit steigender Tiefe immer deutlich weniger Spielfelder erweitert als der IDA* mit der MD. Wie beim A* ist der IDA* mit der MD in den niedrigeren Suchtiefen effizienter, wobei in den höheren Tiefen der IDA* mit der LC wieder die beiden anderen Heuristiken dominiert.

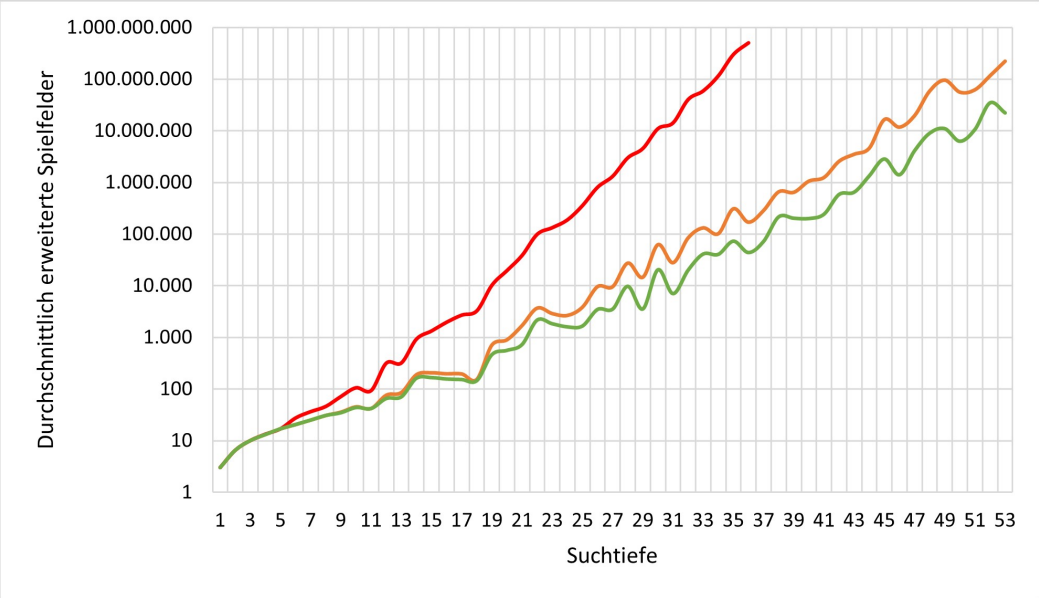

Abbildung 5: Vergleich der erweiterten Spielfelder pro Suchtiefe der drei Heuristiken im IDA*-Algorithmus. HD(rot), MD(orange), LC(grün)

4.4 A* vs. IDA* - Vergleich Algorithmen

Im Vergleich der beiden Suchalgorithmen ist zu beobachten, dass der A* im Laufzeitvergleich bis zu seinem Abbruch aufgrund des Speicherverbrauchs, Puzzles bis zu den Suchtiefen von 41 in ca. 100ms lösen kann. Der IDA* braucht im Gegensatz für Puzzles ab einer Suchtiefe von 38, abhängig von der verwendeten Heuristik, schon deutlich über 100ms. Bei der Anzahl von erweiterten Spielfeldern ist zu erkennen, dass der A* gegenüber dem IDA* durch das Speichern der bekannten Knoten deutlich weniger Spielfelder erweitern muss. Als deutlicher Vergleich dient hier die Anzahl der erweiterten Spielfelder der HD, welche im A* bei einer Tiefe von 25 nur um die 160.000 Spielfelder erweitert hat, wobei der IDA* mit der HD bei einer Tiefe von 25 mehr als 355 Millionen Spielfelder erweitert hat. Auch im Vergleich der anderen Heuristiken setzt sich dieses Muster fort. Bei allen verwendeten Heuristiken hat der A* nie über 200.000 Spielfelder erweitert, wohingegen der IDA* bis zu einer Tiefe von 41 bei der MD schon über 1.2 Milliarden Spielfelder erweitert hat.

5 Zusammenfassung

Die Ergebnisse der Analysen haben gezeigt, dass in beiden Suchalgorithmen die Lineare-Konflikte-Heuristik aufgrund der besseren Schätzung in den höheren Suchtiefen am effizientesten war. Jedoch hat sich auch gezeigt, dass durch die hohe Laufzeitkomplexität der Lineare-Konflikte-Heuristik die Manhattan-Distanz in den niedrigeren Suchtiefen für die in dieser Analyse verwendeten 15-Puzzle effizienter war. Die Hamming-Distanz eignet sich aufgrund der ungenauen Schätzung in beiden Suchalgorithmen am wenigsten. Der Vergleich von den verwendeten Suchalgorithmen hat gezeigt, dass der A* zwar schneller und weniger Spielfelder während der Laufzeit erweitern muss als der IDA*, der A* aber aufgrund des hohen Speicherverbrauchs und des zu großem Suchraums im 15-Puzzle ungeeignet ist. Auf der Basis dieser Beobachtungen kann man daraus schließen, dass der IDA* in Kombination mit der Lineare-Konflikte-Heuristik besser zum Lösen von 15-Puzzles geeignet ist.

6 Ausblick

Um weitere Erkenntnisse im Bezug auf den Vergleich von informierten Suchalgorithmen unter Verwendung von Heuristiken zu bekommen, könnte man aufbauend auf dieser Arbeit weitere informierte Suchalgorithmen einführen, wie z. B. den D* der eine dynamische Variante des A* ist. Darüber hinaus könnte man weitere Heuristiken einführen, wie z. B. die Konzepte der Invert-Distance und der Walking-Distance welche von Ken'ichiro Takahashi erfunden wurden[10].

Literatur

[1] Brüngger, A. ; Marzetta, A. ; Fukuda, K. ; Nievergelt, J.: The parallel search bench ZRAM and its applications. In: *Annals of Operations Research* 90 (1999), Jan, Nr. 0, S. 45–63. – URL https://doi.org/10.1023/A:1018972901171. – ISSN 1572-9338

[2] Google: *Google Core Libraries - Guava*. Feb 2022. – URL https://github.com/google/guava. – visited on 18.11.2022

[3] Hansson, Othar ; Meyer, Andrew E. ; Yung, Mordechai M.: *Generating admissible heuristics by criticizing solutions to relaxed models*. Department of Computer Science, Columbia University, 1985

[4] Hart, Peter E. ; Nilsson, Nils J. ; Raphael, Bertram: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In: *IEEE Transactions on Systems Science and Cybernetics* 4 (1968), Nr. 2, S. 100–107. – URL https://ieeexplore.ieee.org/document/4082128

[5] Korf, Richard E.: Depth-first iterative-deepening: An optimal admissible tree search. In: *Artificial Intelligence* 27 (1985), Nr. 1, S. 97–109. – URL https://www.sciencedirect.com/science/article/pii/0004370285900840. – ISSN 0004-3702

[6] Kornhauser, D. ; Miller, G. ; Spirakis, P.: Coordinating Pebble Motion On Graphs, The Diameter Of Permutation Groups, And Applications. In: *25th Annual Symposium on Foundations of Computer Science, 1984.*, URL https://ieeexplore.ieee.org/document/715921, 1984, S. 241–250

[7] Ratner, Daniel ; Warmuth, Manfred: Finding a Shortest Solution for the NxN Extension of the 15-Puzzle is Intractable. In: *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, AAAI Press, 1986 (AAAI'86), S. 168–172. – URL https://aaai.org/Papers/AAAI/1986/AAAI86-027.pdf

[8] Slocum, Jerry ; Sonneveld, Dic: *The 15 Puzzle Book: How it Drove the World Crazy*. Slocum Puzzle Foundation, 2006

[9] Slocum, Jerry ; Weisstein, Eric W.: *15 puzzle*. Feb 2000. – URL https://mathworld.wolfram.com/15Puzzle.html. – visited on 21.11.2022

[10] Takahashi, Ken'ichiro: *How to make a 15-puzzle automatic solution program(trans. jap.)*. 2001. – URL http://www.ic-net.or.jp/home/takaken/nt/slide/solve15.html. – visited on 10.11.2022

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original