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: 03. März 2023

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung. Das 15-Puzzle ist ein bekanntes Puzzlespiel, bei dem das Ziel darin besteht, durch Verschieben der Spielsteine auf einem 4×4 Spielfeld das Puzzle in eine bestimmte Reihenfolge zu bringen. Aufgrund seiner Einfachheit und der Größe des möglichen Suchraums ist es ein häufig verwendetes Testproblem in der Informatik. Eine Möglichkeit, das 15-Puzzle zu lösen, besteht in der Anwendung von informierten Suchalgorithmen, die mithilfe von Heuristiken, die als Orientierungshilfe dienen, eine optimale Lösung für ein beliebiges Puzzle berechnen können. Diese Hausarbeit beschäftigt sich mit dem Vergleich zwischen den informierten Suchalgorithmen A* und IDA* unter der Verwendung von Heuristiken mit der Fragestellung, welcher Algorithmus mit welcher Heuristik hinsichtlich der Laufzeit- und Speichereffizienz besser zum Lösen des 15-Puzzles geeignet ist. Um Aussagen darüber treffen zu können, ob ein Algorithmus besser geeignet ist, werden Experimente durchgeführt, bei denen beide Algorithmen unter Verwendung der Heuristiken eine gleiche Anzahl von identischen Puzzles lösen müssen. Abschließend werden die Ergebnisse ausgewertet und der geeignetere Algorithmus bestimmt. Die Ergebnisse der Experimente haben ergeben, dass der A* aufgrund der Eigenschaft des Speicherplatzverbrauchs, der abhängig von der Suchtiefe sehr groß werden kann, für das Lösen des 15-Puzzles ungeeignet ist. Der IDA* hatte im Vergleich zu dem A* immer eine durchschnittlich höhere Laufzeit, aber durch die Eigenschaft nicht Speicherplatz beschränkt zu sein, könnte der IDA* theoretisch jedes beliebige 15-Puzzle lösen. In beiden Algorithmen erwies sich zudem, dass mit der Linear-Konflikt-Heuristik die Puzzles schneller gelöst werden konnten als mit der Manhattan- und Hamming-Distanz. Somit ist der IDA* in Kombination mit der Linear-Konflikt-Heuristik für das konkrete Problem des 15-Puzzles am besten geeignet.

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

1 Einleitung

Eines der am weitverbreitetsten Testprobleme für uninformierte und informierte Suchalgorithmen in der Informatik 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[9]. Das 15-Puzzle besteht aus einem 4×4 Spielfeld, wobei diesem Spielfeld 15 Spielsteine mit den Zahlen von $1 - 15$ zugeordnet sind und einem Feld, welches unbesetzt bleibt. Durch das Verschieben der Spielsteine in das unbesetzte Feld kann ein Startzustand in den Zielzustand überführt werden. Ziel des Spiels ist es, die beliebige Anordnung der Spielsteine wieder in die Reihenfolge des Zielzustands (siehe Abbildung rechts 1) zu bringen. Für jeden gültigen Startzustand kann man die optimale Lösung, also die Mindestanzahl an Zügen zu ihrem Zielzustand, in maximal 80 Zügen, berechnen[1]. Diese Hausarbeit befasst sich mit dem Vergleich zwischen den beiden in-

2		7	4
1	3	6	8
5	9	15	10
13	14	11	12

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

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

formierten Suchalgorithmen A* und IDA* mit der Fragestellung, welcher Algorithmus besser zum Lösen des 15-Puzzles geeignet ist. Um Aussagen über die Eignung eines Algorithmus unter Verwendung von Heuristiken treffen zu können, werden Experimente durchgeführt, um die Laufzeit(Zeit zum Lösen)- und Speichereffizienz(Menge erweiterter Spielfelder) der Algorithmen messen und anschließend vergleichen zu können. Die in dieser Arbeit verwendeten Heuristiken sind die Manhattan- und die Hamming-Distanz sowie die Linear-Konflikt-Heuristik. Der Rest dieser Hausarbeit ist in folgende Abschnitte und Kapitel unterteilt: In den folgenden Abschnitten wird auf die Komplexitätsklasse und die Lösbarkeit des 15-Puzzles eingegangen. In Kapitel 2 werden die verwendeten Algorithmen eingeführt und die Funktionsweise im Bezug auf das 15-Puzzle beschrieben. Kapitel 3 erläutert die Anforderungen und die Funktionsweisen der verwendeten Heuristiken in den Suchalgorithmen. Der Aufbau der durchgeführten Experimente und die erhobenen Daten werden in Kapitel 4 beschrieben und die Ergebnisse in Kapitel 5 präsentiert. In Kapitel 6 werden die Ergebnisse diskutiert und analysiert. Ein Fazit und einen Ausblick werden in den Kapiteln 7 und 8 gegeben.

1.1 Komplexitätsklasse

Das 15-Puzzle sowie alle anderen Varianten der n-Puzzle werden zur Klasse der NP-vollständigen Probleme[7] zugeordnet. Dies bedeutet, dass eine Lösung zwar schnell auf Korrektheit überprüft werden kann, aber unter der Annahme, dass $P \neq NP$ gilt, kein Algorithmus existiert, der die kürzeste Lösung für ein beliebiges n-Puzzle in Polynomialzeit berechnen kann.

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[10], 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* 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 Speicherplatzverbrauch 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* ist wie der A* 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* garantiert[6] den kürzesten Pfad zum Zielknoten, wenn die Heuristik wie bei dem A* zulässig ist (siehe Abschnitt 3.1).

3 Heuristiken

Die Heuristiken, welche 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 eine 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 eine Heuristik zulässig (3.1) und konsistent (3.2) ist. Alle Heuristiken, die in dieser Hausarbeit verwendet werden, sind zulässig und konsistent.

3.1 Zulässigkeit

Eine Heuristik h ist zulässig[8], 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 sind als die niedrigsten möglichen Kosten vom aktuellen Knoten zum Ziel.

3.2 Konsistenz

Eine Heuristik h ist konsistent[8], wenn für jeden Knoten n und deren Nachfolger n' durch eine Aktion a, bei der eine Permutation durch den A* oder IDA* erstellt wird, gilt, dass $h(n) \leq c(n, a, n') + h(n')$ erfüllt ist. Das heißt, die geschätzten Kosten vom Knoten n zum Zielknoten sind niemals höher als die Kosten von Knoten n zum Knoten n' plus die geschätzten Kosten, um von n' den Zielknoten zu erreichen.

3.3 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 mit 0 gewertet wird. Die Distanz von einem Spielstein n zu der Zielposition wird folgende Gleichung berechnet:

$$d_1(n) = |x_1 - x_2| + |y_1 - y_2| \tag{1}$$

wobei x_1, y_1 für die aktuelle Position und x_2, y_2 für die Zielposition von n steht.
Am Beispiel von Abbildung[1] links beträgt die Gesamtsumme MD_{sum} für alle Spielsteine im aktuellen Spielzustand:

$$MD_{sum} = \begin{array}{c|cccccccccccccccc} d_1(n) & 2 & 0 & 7 & 4 & 1 & 3 & 6 & 8 & 5 & 9 & 15 & 10 & 13 & 14 & 11 & 12 \\ \hline & 1 & 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 1 \end{array} = 13$$

3.4 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 mit 0 gewertet wird. Wenn ein Spielstein von seiner Zielposition abweicht, wird die Gesamtsumme um 1 erhöht. Der Wert für einen Spielstein wird durch folgende Gleichung bestimmt:

$$d_2(n) = \begin{cases} 1, & \text{if } d_1(n) \neq 0, \\ 0, & \text{else.} \end{cases} \tag{2}$$

wobei $d_1(n)$ hier die Gleichung ist, welche zum berechnen der Distanz eines Spielsteins zu seiner Zielposition ist.
Am Beispiel von Abbildung[1] links beträgt die Gesamtsumme HD_{sum} für alle Spielsteine im aktuellen Spielzustand:

$$HD_{sum} = \begin{array}{c|cccccccccccccccc} d_2(n) & 2 & 0 & 7 & 4 & 1 & 3 & 6 & 8 & 5 & 9 & 15 & 10 & 13 & 14 & 11 & 12 \\ \hline & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{array} = 11$$

3.5 Linear-Konflikt-Heuristik

Die Linear-Konflikt-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 Summe der linearen Konflikte $C_{sum} = 2$, da die Spielsteine $n_{11}(15)$ und $n_{15}(11)$ in einem linearen Konflikt zueinanderstehen. Damit beträgt die Gesamtsumme von $LC_{sum} = C_{sum} + MD_{sum} = 2 + 13 = 15$.

4 Methoden

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 des 15-Puzzles 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 Experimente war es wichtig, aussagekräftige Ergebnisse im Bezug auf den Vergleich der Suchalgorithmen zu erhalten. Aus diesem Grund wurden für die Experimente Puzzles erstellt[5], bei denen die optimale Anzahl von Zügen berechnet wurde, um diese so in Klassen der verschiedenen Suchtiefen unterteilen zu können. Für jede

Suchtiefe wurden zehn unterschiedliche Puzzles erstellt, wobei es für die Suchtiefen eins und zwei nur jeweils zwei und vier mögliche Zustände gibt, weshalb diese entsprechend anteilig wiederholt und auf zehn Puzzles ergänzt wurden. Die unterteilten Puzzles wurden dann durch beide Suchalgorithmen unter Verwendung der einzelnen Heuristiken gelöst. Die Experimente wurden jeweils 100 Mal wiederholt und anschließend das arithmetische Mittel von den Laufzeiten und den erweiterten Spielfeldern berechnet, um präzisere Ergebnisse zu erhalten.

5 Ergebnisse

In den Experimenten wurden insgesamt 530 Puzzles[5] durch den A* und IDA* gelöst und Messdaten gesammelt. In den Tabellen 1 und 2 befindet sich ein Auszug der Messdaten der jeweiligen Algorithmen. Dabei wurden jeweils die Messdaten der Grenze der maximal erreichten Suchtiefe der jeweiligen Heuristik im A* und IDA* genommen, um eine bessere Übersicht über den Vergleich zu erhalten. Der A* konnte mit der Hamming-Distanz Puzzles bis zu einer Suchtiefe von 29 lösen und benötigte bei dieser Tiefe im Durchschnitt für die einzelnen Puzzles etwa 115 ms, erweitert wurden im Schnitt fast 80.000 Spielfelder. Mit der Manhattan-Distanz waren Puzzles bis zu einer Tiefe von 41 lösbar, dabei brauchte der A* bei der Tiefe von 41 in etwa 50 ms pro Puzzle, erweitert wurden hier im Schnitt fast 39.000 Spielfelder. Die Linear-Konflikt-Heuristik konnte Puzzles bis zu der Tiefe von 44 lösen, wobei die Laufzeiten im Schnitt bei dieser in etwa 45 ms betrug und um die 27.000 Spielfelder erweitert wurden.

		HD	**MD**	**LC**
Tiefe 29	Laufzeit in ms:	115,304	0,988	0,539
	Erweiterte Spielfelder:	79.974	1.252	542
Tiefe 41	Laufzeit in ms:	-	50,493	16,279
	Erweiterte Spielfelder:	-	38.512	11.145
Tiefe 44	Laufzeit in ms:	-	-	45,729
	Erweiterte Spielfelder:	-	-	27.327

Tabelle 1: Durchschnitt der Laufzeit und der erweiterten Spielfelder im A* bei 100 Wiederholungen der Experimente.

Der IDA* konnte mit der Hamming-Distanz Puzzles bis zu einer Tiefe von 36 lösen und benötigte dabei in einer Tiefe von 36 im Durchschnitt 9,8 s pro Puzzle, hierbei wurden im Durchschnitt um die 29,2 Mio. Spielfelder erweitert. Mit der Manhattan-Distanz und

der Linear-Konflikt-Heuristik konnte Puzzles bis zu einer Tiefe von 53 gelöst werden. Dabei betrug die durchschnittliche Laufzeit bei der Manhattan-Distanz in etwa 4,9 s und bei der Linear-Konflikt-Heuristik in etwa 1,2 s. Die Anzahl der erweiterten Spielfelder betrugen jeweils 12,7 Mio. und 1,9 Mio. in den beiden Heuristiken.

		HD	**MD**	**LC**
Tiefe 29	Laufzeit in ms:	150,676	1,272	0,773
	Erweiterte Spielfelder:	361.205	2.712	1.043
Tiefe 36	Laufzeit in ms:	9.825,371	13,586	6,349
	Erweiterte Spielfelder:	29.231.623	26.749	7.615
Tiefe 41	Laufzeit in ms:	-	58,914	23,101
	Erweiterte Spielfelder:	-	118.142	29.253
Tiefe 44	Laufzeit in ms:	-	158,511	66,035
	Erweiterte Spielfelder:	-	352.116	85.271
Tiefe 53	Laufzeit in ms:	-	4.941,943	1.248,228
	Erweiterte Spielfelder:	-	12.704.646	1.981.919

Tabelle 2: Durchschnitt der Laufzeit und der erweiterten Spielfelder im IDA* bei 100 Wiederholungen der Experimente.

Um einen Gesamtüberblick über den Verlauf der Laufzeit- und Speichereffizienz in den Algorithmen zu erhalten, wurden alle gemessenen Daten in zwei Liniendiagramme aufgeteilt und anschaulich im Anhang A und B visualisiert.

6 Diskussion

Im Anhang A der Laufzeiteffizienz der beiden Algorithmen kann man beobachten, dass das Lösen der Puzzles mit der Hamming-Distanz in beiden Algorithmen am längsten Zeit benötigt hat, dabei kann man ab einer Suchtiefe von 14 einen deutlichen Anstieg der Laufzeit feststellen, welcher sich in den darauf folgenden Tiefen deutlich fortführt. Mit der Linear-Konflikt-Heuristik konnten beide Algorithmen die Puzzles am schnellsten lösen, da diese Heuristik auf der Manhattan-Distanz aufbaut und diese um die linearen Konflikte erweitert und deshalb den Spielzustand besser abschätzen konnte. Besonders deutlich kann man den Unterschied in den höheren Suchtiefen und in den Tabellen 1 und 2 beobachten. Die Linear-Konflikt-Heuristik ist hier bei einer Tiefe von 41, in dem A* um ca. das Dreifache und im IDA* um ca. das 2,5-Fache schneller als mit der Manhattan-Distanz. In dem IDA* war die Linear-Konflikt-Heuristik bei einer Tiefe von 53 sogar um das ca. 6,4-Fache schneller als die Manhattan-Distanz. Trotzdem war die

Manhattan-Distanz in beiden Algorithmen in den niedrigen Suchtiefen schneller als die Linear-Konflikt-Heuristik, da die Laufzeitkomplexität der Manhattan-Distanz deutlich niedriger ist als die von der Linear-Konflikt-Heuristik, dieser Vorteil wurde aber in den höheren Tiefen durch die bessere Schätzung von der Linear-Konflikt-Heuristik aufgehoben.
Im Anhang B der Speichereffizienz der beiden Algorithmen kann man erkennen, dass wie in Anhang A die Hamming-Distanz auch hier am schlechtesten im Bezug auf die erweiterten Spielfelder ist und die Linear-Konflikt-Heuristik gegenüber den anderen Heuristiken erneut am besten abschneidet und in allen Tiefen am wenigsten Spielfelder erweitert hat. Aus den Experimenten geht außerdem hervor, dass der A* nicht in der Lage war, alle in den Experimenten verwendeten Puzzles zu lösen. Dabei war es abhängig von der verwendeten Heuristik. Unter der Linear-Konflikt-Heuristik konnte der A* nur bis zu einer Tiefe von 44 Puzzles lösen, da der auf dem Testsystem maximal verfügbare Speicherplatz erschöpft war. Das hängt mit der Eigenschaft zusammen, dass der A* während der Laufzeit alle bereits bekannten Spielfelder im Speicher behält, um nicht mehrfach ein bereits erweitertes Spielfeld untersuchen zu müssen. Der IDA* im Gegensatz hatte dieses Problem nicht. In den Experimenten konnte der IDA* Puzzles bis zu einer Tiefe von 53 mit der Manhattan-Distanz und der Linear-Konflikt-Heuristik lösen, bevor dieser manuell durch die hohe Laufzeit, um ein Puzzle zu lösen abgebrochen wurde. Aufgrund der Eigenschaft, dass der IDA* keine bereits bekannten Spielfelder im Speicher hält und nur anhand der momentanen niedrigsten Kosten, die durch die Heuristik berechnet wurde, Spielfelder erweitert, kommt es vor, dass dieser Spielfelder mehrmals erweitert, was sich in der Laufzeit bemerkbar macht. Aus diesem Grund kann der IDA* theoretisch den gesamten Suchraum des 15-Puzzles durchsuchen, benötigt dafür aber viel Zeit.

7 Fazit

Die Ergebnisse der Experimente konnten zeigen, dass von allen verwendeten Heuristiken die Linear-Konflikt-Heuristik von der Laufzeit und vom Speicherplatzverbrauch am effizientesten in beiden Algorithmen war. Außerdem konnte gezeigt werden, dass der A* in fast allen Suchtiefen schneller als der IDA* war und dieser durch seine Eigenschaft optimal effizient zu sein, immer weniger oder gleich viele Spielfelder wie der IDA* erweiterte, aufgrund dessen konnte bestätigt werden, dass die Experimente richtig und unter idealen Bedingungen durchgeführt wurden. Aufgrund der Eigenschaft des enormen Speicherplatzverbrauchs ist der A* für das Lösen des 15-Puzzles ungeeignet, da der Suchraum

zu groß ist und der Speicherplatz auf den allermeisten Testsystemen nicht ausreicht, um alle bekannten Spielfelder im Speicher zu halten. Der IDA* ist aufgrund der Eigenschaft, dass der Speicherplatzverbrauch proportional zu der Anzahl der Spielfelder von dem Start- zum Zielzustand ist, deutlich besser für das Lösen des 15-Puzzles geeignet, da dieser den gesamten Suchraum des 15-Puzzles betrachten kann.

8 Ausblick

Um weitere Erkenntnisse über die Eignung von Algorithmen für das 15-Puzzle gewinnen zu können, könnten aufbauend auf dieser Arbeit heuristische Optimierungsverfahren wie der Bergsteigeralgorithmus eingesetzt und die Ergebnisse mit denen dieser Arbeit verglichen werden. Darüber hinaus könnten weitere Heuristiken für den A* und IDA* eingeführt werden, um die Laufzeit- und Speichereffizienz zu verbessern. Eine mögliche Heuristik ist die Walking-Distance, welche von Ken'ichiro Takahashi erfunden wurde[11].

A Laufzeiteffizienz A* und IDA*

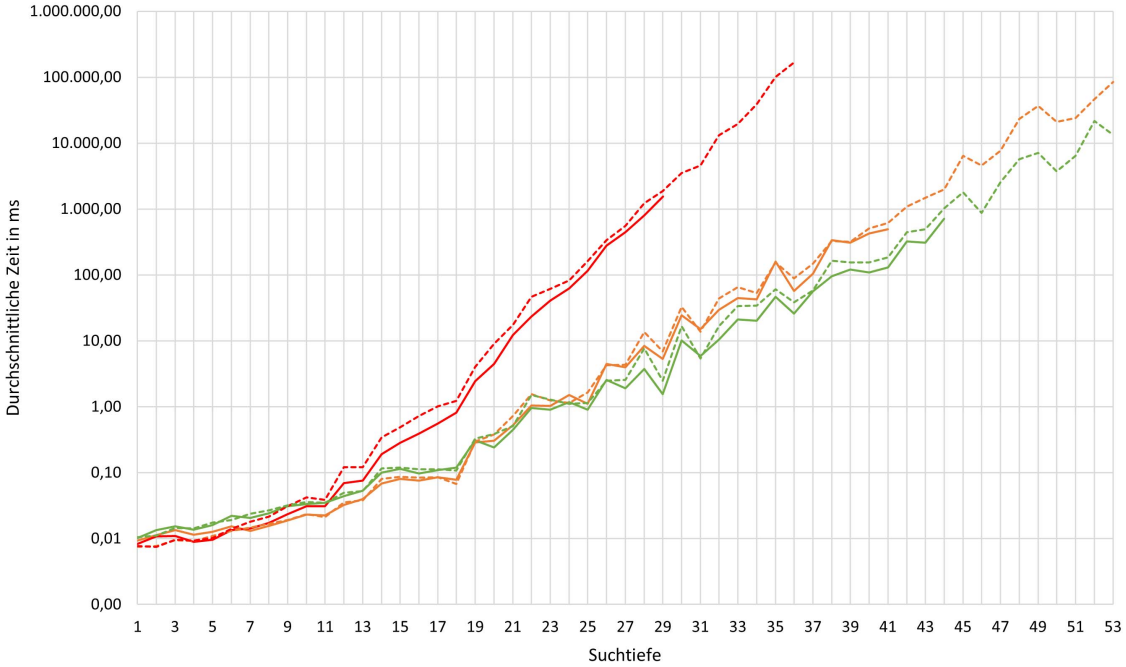

Abbildung 2: Vergleich der Laufzeiten pro Suchtiefe der drei Heuristiken im A* (durchgezogene Linie) und IDA* (gestrichelte Linie).
HD (rot), MD (orange), LC (grün)

B Speichereffizienz A* und IDA*

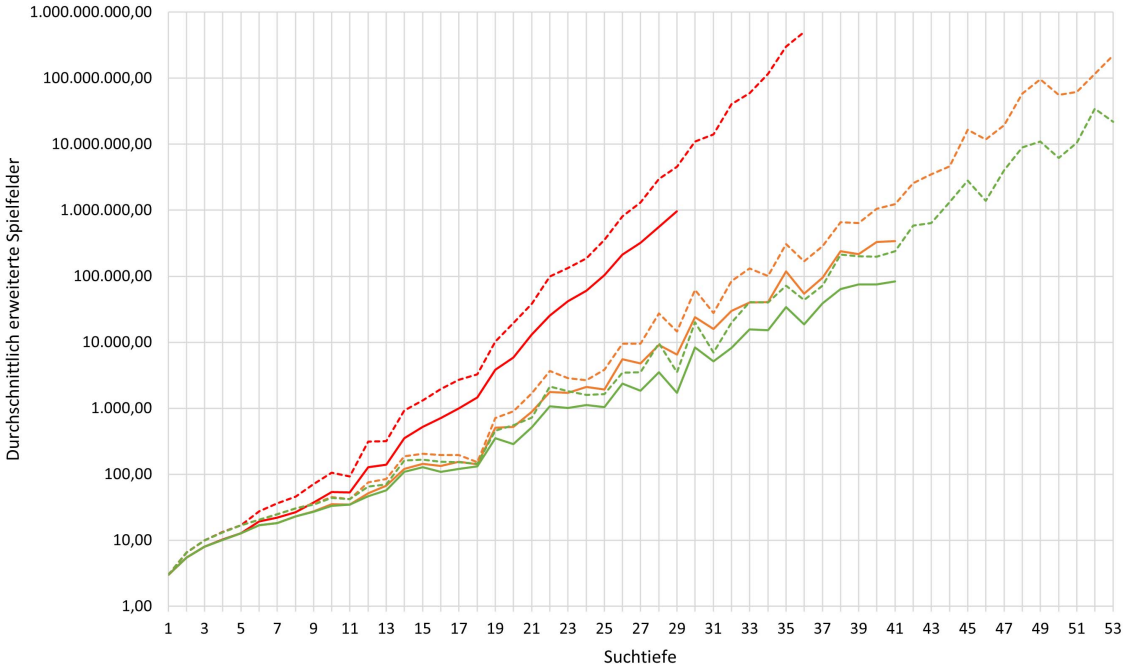

Abbildung 3: Vergleich der erweiterten Spielfelder pro Suchtiefe der drei Heuristiken im A* (durchgezogene Linie) und IDA* (gestrichelte Linie).
HD (rot), MD (orange), LC (grün)

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. – URL https://doi.org/10.7916/D89Z9CW3

[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] Hoefflin, Niklas: *HAW-IS-15-Puzzle-Solver*. 3 2023. – URL https://github.com/itakurah/HAW-IS-15-Puzzle-Solver/blob/fd0da04fa9dd92ef801dfbc863dfc5445aef4e3d/src/main/resources/puzzles/

[6] 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

[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] Russell, Stuart J. ; Norvig, Peter: *Artificial Intelligence: A modern approach*. 3. Pearson, 2016

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

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

[11] 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