

Funktionsprinzipien und Anwendungen von Algorithmen zur Pfadplanung

Bearbeiter 1: Mohammed Salih Mezraoui

Bearbeiter 2: David Gruber

Bearbeiter 3: Marius Müller

Gruppe: WissArb22/Thema 2/Gruppe-7

Ausarbeitung zur Vorlesung Wissenschaftliches Arbeiten

Trier, 15.07.2022

Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

The same in english.

Inhaltsverzeichnis

1 Einleitung und Problemstellung	1
2 Algorithmen zur Pfadplanung	2
2.1 Pfadplanung	2
2.2 Dijkstra Algorithmus	2
2.2.1 Definition	2
2.2.2 Prinzip	2
2.2.3 Dijkstras Vorteile	3
2.2.4 Dijkstras Nachteile	3
2.2.5 Dijkstras Pseudo-Code	3
2.3 Bellman-Ford-Algorithmus	5
2.3.1 Definition	5
2.3.2 Prinzip	5
2.3.3 Bellman-Ford Pseudo-Code	5
3 Optimierungsstrategien	7
3.1 Warum existieren unterschiedliche Konsistenzmodelle?	7
4 Anwendungen	8
4.1 GIS(Geoinformationssystem)	8
4.1.1 Definition	8
4.1.2 Geoinformationssystem mit Dijkstra-Algorithmus	8
4.2 Mobile Roboter mit verbessertem Dijkstra-Algorithmus	10
4.2.1 traditioneller Dijkstra vs besserer Dijkstra	10
4.2.2 Verbesserter Dijkstra	11
5 Zusammenfassung und Ausblick	13
Literaturverzeichnis	14
Glossar	15

1

Einleitung und Problemstellung

Begonnen werden soll mit einer Einleitung zum Thema, also Hintergrund und Ziel erläutert werden.

Weiterhin wird das vorliegende Problem diskutiert: Was ist zu lösen, warum ist es wichtig, dass man dieses Problem löst und welche Lösungsansätze gibt es bereits. Der Bezug auf vorhandene oder eben bisher fehlende Lösungen begründet auch die Intention und Bedeutung dieser Arbeit. Dies können allgemeine Gesichtspunkte sein: Man liefert einen Beitrag für ein generelles Problem oder man hat eine spezielle Systemumgebung oder ein spezielles Produkt (z.B. in einem Unternehmen), woraus sich dieses noch zu lösende Problem ergibt.

Im weiteren Verlauf wird die Problemstellung konkret dargestellt: Was ist spezifisch zu lösen? Welche Randbedingungen sind gegeben und was ist die Zielsetzung? Letztere soll das beschreiben, was man mit dieser Arbeit (mindestens) erreichen möchte.

2

Algorithmen zur Pfadplanung

In diesem Kapitel werden zwei der am häufigsten verwendeten Algorithmen (Dijkstra und Bellman) für die Planung von Pfaden beschrieben.

2.1 Pfadplanung

Karthik Karur, Nitin Sharma, Chinmay Dharmatti, und Joshua E. Siegel haben Pfadplanung definiert als „ein nicht-deterministisches Polynomialzeit ("NP") schweres Problem definiert, dessen Aufgabe es ist, einen kontinuierlichen Pfad zu finden, der ein System von einer Anfangs- zu einer Endkonfiguration verbindet“¹.

Die Komplexität des Problems wächst mit der Zunahme der Freiheitsgrade des Systems. Die zu wählende Route (die beste Route) wird durch Einschränkungen und Begrenzungen bestimmt, wie z. B. die kürzeste Entfernung zwischen den Endpunkten oder die kürzeste Zeit für eine kollisionsfreie Fahrt. Gelegentlich werden Beschränkungen und Ziele kombiniert, z. B. der Versuch, den Energieverbrauch zu begrenzen und gleichzeitig einen bestimmten Schwellenwert für die Fahrzeit nicht zu überschreiten.[KSDS21].

2.2 Dijkstra Algorithmus

2.2.1 Definition

Adeel Javaid beschrieb, dass „Der Dijkstra-Algorithmus (benannt nach seinem Entdecker E.W. Dijkstra) das Problem löst, den kürzesten Weg von einem Punkt in einem Diagramm (der Quelle) zu einem Ziel zu finden“².

2.2.2 Prinzip

Da nachgewiesen wurde, dass die kürzesten Wege von einem Quellknoten zu allen Orten in einem Diagramm tatsächlich in derselben Zeit gefunden werden können,

¹ Definition der Pfadplanung [KSDS21]

² Dijkstra Algorithmus Definition [Jav19]

wird dieses Problem manchmal als Problem der kürzesten Wege für eine einzige Quelle bezeichnet[Jav19].

Der Dijkstra-Algorithmus kann die beste Route wählen, die die Voraussetzungen für die Topologie-Roadmap erfüllt. Der klassische Dijkstra-Algorithmus unterteilt die Knoten im topologischen System in drei Gruppen: zunächst alle vorläufigen Knoten im System des Algorithmus, die nicht gekennzeichnet sind, und die Knoten, die sich während der Routenauswahl und der effizienten Routenauswahl kreuzen und verbinden sollen. Jede Screening-Schleife im optimalen Routenauswahlverfahren wählt den Knoten mit der kürzesten Pfadlänge vom momentanen Indikator knoten als permanenten Markierung-knoten, und der Dijkstra-Algorithmus iteriert, bis der aktuelle und der geplante oder alle Knoten erreicht sind. Der Knoten würde dann so lange bestehen bleiben, bis er durch einen permanenten Markierung-knoten ersetzt wird[ZG13].

2.2.3 Dijkstras Vorteile

- Minhang Zhou und Nina Gao haben zitiert „Der Dijkstra-Algorithmus kann alle optimalen Pfade finden, und die Trefferquote dieser optimalen Pfade liegt bei 100 %“³.
- Wenn der geplante Zielknoten erreicht ist, besucht der Dijkstra-Algorithmus die restlichen unerwünschten Knoten nicht[AIS⁺20].

2.2.4 Dijkstras Nachteile

Der Hauptnachteil des Algorithmus besteht darin, dass er eine blinde Suche durchführt, wodurch viel Zeit und relevante Ressourcen verschwendet werden, oder anders ausgedrückt, er ist zeitintensiv. Ein weiterer Nachteil ist, dass er keine negativen Seiten verwalten kann, was zu azyklischen Graphen führt, und dass er häufig nicht die beste Route findet[MS20].

2.2.5 Dijkstras Pseudo-Code

Der Dijkstra-Algorithmus arbeitet mit der Zuweisung einiger vorläufiger Entfernungswerte und versucht, diese schrittweise zu verbessern. Der Pseudocode des Algorithmus ist in der folgenden Abbildung dargestellt[HG12].

³ Erster Vorteil von Dijkstra [ZG13]

Algorithm 1 Dijkstra Algorithm

```

1: function DIJKSTRA (Graph, source)
2:   create vertex set D
3:   for each vertex v in Graph:
4:     distance[v]  $\leftarrow$  INFINITY
5:     previous[v]  $\leftarrow$  UNDEFINED
6:     add v to D
7:     distance[source]  $\leftarrow$  0
8:
9:   while D is not empty do:
10:    u  $\leftarrow$  in D with min distance[u]
11:    remove u from D
12:    for each neighbour v of u:
13:      alt  $\leftarrow$  distance[u] + length (u, v)
14:      if alt < distance[v]
15:        distance[v]  $\leftarrow$  alt
16:        previous[v]  $\leftarrow$  u
17:   end while
18:   return distance [], previous []
19: end function

```

Abb. 2.1. Dijkstra Algorithmus Pseudo-Code [AIS⁺20].

Abusalim Samah, Ibrahim Rosziati, Saringat Mohd, Jamel Sapiee und Wahab Jahari haben diesen Pseudocode wie folgt beschrieben:

„Beim Dijkstra-Algorithmus ist der Pfad nicht bekannt. Die Knoten werden in zwei Gruppen unterteilt: temporäre (t) und permanente (p).

- Zunächst wird die Entfernung des Quellknotens mit dem Wert Null initialisiert [$\text{distance}(a) = 0$], und die Entfernung der anderen Knoten wird mit dem Wert Unendlich belegt [$\text{distance}(x) = \text{infinity}$].
- Schritt 2: Suche nach dem Knoten x mit dem kleinsten Wert von $d(x)$. Wenn es keine temporären Knoten gibt oder der Wert von $d(x)$ gleich unendlich ist, wurde der Knoten x als permanent eingestuft, was bedeutet, dass sich $d(x)$ und der übergeordnete Wert von $d(x)$ nicht mehr ändern werden.

- Schritt 3: Wenden Sie den folgenden Vergleich für jeden temporären Knoten mit der Bezeichnung vertex y an, der an x angrenzt⁴.

2.3 Bellman-Ford-Algorithmus

2.3.1 Definition

Vaibhavi Patel und Prof. Chitra Baggar haben benotet dass, „Der Bellman-Ford-Algorithmus verwendet Entspannung, um kürzeste Pfade auf gerichteten Graphen zu finden, die nur eine Quelle haben“⁵.

2.3.2 Prinzip

Wenn es negative Gewichtsnusszyklen gibt, wird der Algorithmus sie erkennen. Eine negative Länge gibt es nicht, wenn es sich um Bereiche auf einer Karte handelt. Der Ballmann-Ford-Algorithmus ist im Allgemeinen mit dem Dijkstra-Algorithmus vergleichbar. Er entspannt alle Kanten $|V|$ mal, wobei $|V|$ die Menge der Scheitelpunkte ist[VP14].

2.3.3 Bellman-Ford Pseudo-Code

Der Bellman-Ford-Algorithmus wird wie in der Abbildung unten dargestellt ausgeführt.

⁴ Ausführliche Beschreibung von Dijkstra Pseudo Code [AIS⁺20]

⁵ Bellman-Ford-Algorithmus [VP14]

Algorithm 2 Bellman-Ford Algorithm

```

1: function bellmanFord (G, S)
2:   for each vertex V in G
3:      $distance[v] \leftarrow \text{INFINITY}$ 
4:      $previous[v] \leftarrow \text{NULL}$ 
5:      $distance[s] \leftarrow 0$ 
6:   for each vertex V in G
7:     for each edge (u, v) in G
8:        $alt \leftarrow distance[u] + length(u, v)$ 
9:       if  $alt < distance[v]$ 
10:         $distance[v] \leftarrow alt$ 
11:         $previous[v] \leftarrow u$ 
12:
13:   for each edge (u, v) in G
14:     if  $distance[u] + length(u, v) < distance(v)$ 
15:       Error: Negative Cycle Exists
16: return  $distance[], previous[]$ 

```

Abb. 2.2. Bellman-Ford Pseudo-Code[AIS⁺20].

Abusalim Samah, Ibrahim Rosziati, Saringat Mohd, Jamel Sapiee und Wahab Jahari haben sie diesen Pseudocode wie folgt beschrieben:

- „Schritt 1: Setzen Sie den Abstand des Quellknotens s auf den Wert Null ($distance[s] = 0$) und weisen Sie den anderen Knoten einen Abstand von INFINITY zu.
- Schritt 2: entspannt jede Kante ($n - 1$) Mal, wenn n die Anzahl der Knoten ist. Das Entspannen einer Kante bedeutet zu prüfen ob es möglich ist, den Weg zu dem Knoten, auf den die Kante zeigt, zu verkürzen, und, wenn ja, den Weg zu den Knoten durch die gefundene Route.
- Schritt 3: Prüfen, ob der Graph einen negativen Zyklus hat, mit Ausführung der N-ten Schleife⁶.

⁶ Ausführliche Beschreibung von Bellman-Ford Pseudo Code“ [AIS⁺20]

3

Optimierungsstrategien

In diesem Kapitel wird beschrieben, warum es unterschiedliche Konsistenzmodelle gibt. Außerdem werden die Unterschiede zwischen strengen Konsistenzmodellen (Linearisierbarkeit, sequentielle Konsistenz) und schwachen Konsistenzmodellen (schwache Konsistenz, Freigabekonsistenz) erläutert. Es wird erklärt, was Strenge und Kosten (billig, teuer) in Zusammenhang mit Konsistenzmodellen bedeuten.

3.1 Warum existieren unterschiedliche Konsistenzmodelle?

Laut [Mal97] sind mit der Replikation von Daten immer zwei gegensätzliche Ziele verbunden: die Erhöhung der Verfügbarkeit und die Sicherung der Konsistenz der Daten. Die Form der Konsistenzsicherung bestimmt dabei, inwiefern das eine Kriterium erfüllt und das andere dementsprechend nicht erfüllt ist (Trade-off zwischen Verfügbarkeit und der Konsistenz der Daten). Stark konsistente Daten sind stabil, das heißt, falls mehrere Kopien der Daten existieren, dürfen keine Abweichungen auftreten. Die Verfügbarkeit der Daten ist hier jedoch stark eingeschränkt. Je schwächer die Konsistenz wird, desto mehr Abweichungen können zwischen verschiedenen Kopien einer Datei auftreten, wobei die Konsistenz nur an bestimmten Synchronisationspunkten gewährleistet wird. Dafür steigt aber die Verfügbarkeit der Daten, weil sie sich leichter replizieren lassen.

4

Anwendungen

In diesem Kapitel wird beschrieben, wie Dijkstra- und Bellman-Algorithmen in realen Anwendungen angewendet werden können.

4.1 GIS(Geoinformationssystem)

4.1.1 Definition

nach Vaibhavi Patel und Prof.Chitra Baggar „Ein geografisches Informationssystem (GIS) ist ein computergestütztes Werkzeug. Mit diesen Werkzeugen können wir räumliche Informationen erstellen, manipulieren, analysieren, speichern und anzeigen. Räumliche Informationen sind Informationen über Objekte, die sich auf der Erde befinden, wie z. B. Städte, Eisenbahnstrecken, Flüsse usw“¹.

4.1.2 Geoinformationssystem mit Dijkstra-Algorithmus

Obwohl dieser Dijkstra-Algorithmus sehr effektiv zu sein scheint, kann es im Grunde genommen sehr lange dauern, die Route vom Start bis zum Ende weiterzuleiten, wobei eine große Menge an Informationen verwendet wird, wie in der Abbildung 4.1 zu sehen ist, was überhaupt nicht angemessen ist, so dass eine Alternative entwickelt werden muss, um die Zeit zu verkürzen[HAMTMA20].

¹ GIS Definition [VP14]

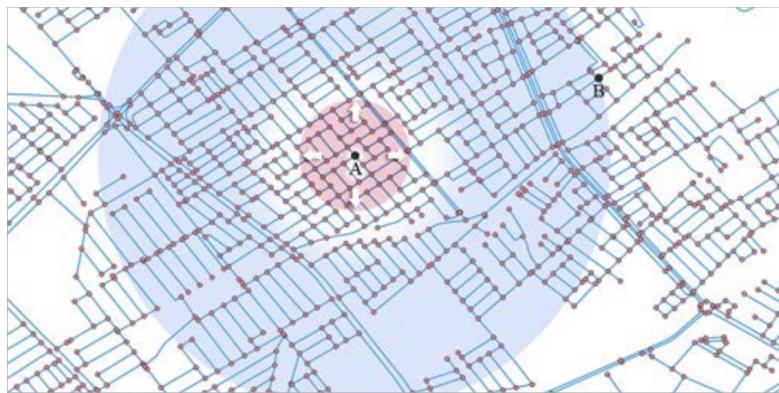


Abb. 4.1. Dijkstra's Fortschritt[HAMTMA20].

So haben die Autoren des Artikels² ihr Experiment wie folgt beschrieben:

„Der rote Kreis stellt den Verlauf des Dijkstra-Algorithmus dar, während der blaue Kreis die tatsächlichen Knoten und Linien des Graphen anzeigt, die der Algorithmus verarbeiten muss, bevor er die korrekte Wurzel von A nach B berechnen kann. Der verbesserte Algorithmus kann den Suchbereich erheblich verkleinern, wie in Abbildung 4.2 dargestellt. Dies ist nur ein Bruchteil des Bereichs in der vorherigen Abbildung. Er ermöglicht wesentlich schnellere Berechnungen und verkürzt damit die Zeit, die der Routing-Prozess benötigt, um einen gültigen Pfad zu finden. Die Verbesserungsmethode besteht im Wesentlichen darin, einen temporären Datensatz zu erstellen, bevor der Dijkstra-Algorithmus selbst gestartet wird, und diesen so zu behandeln, als wäre er der Graph, mit dem der Algorithmus arbeiten muss. Dieser Datensatz wird erstellt, nachdem die Start- und Endknoten erfasst wurden. Mit den Koordinaten der Start- und Endknoten kann der Datensatz aus den Hauptdaten ausgeschlossen werden, indem nur die Knoten ausgewählt werden, die sich innerhalb des von den beiden Knoten gebildeten Quadrats befinden“.



Abb. 4.2. Verbesserter Suchbereich[HAMTMA20].

² GIS Beispiel [HAMTMA20]

4.2 Mobile Roboter mit verbessertem Dijkstra-Algorithmus

4.2.1 traditioneller Dijkstra vs besserter Dijkstra

Die Autoren des Artikels³ haben zitiert dass: „Der traditionelle Dijkstra-Algorithmus beruht auf einer gierigen Strategie zur Pfadplanung. Er wird verwendet, um den kürzesten Weg in einem Graphen zu finden. Er befasst sich mit der Lösung des kürzesten Weges, ohne sich formal um die Pragmatik der Lösung zu kümmern.“

Der modifizierte Dijkstra-Algorithmus zielt darauf ab, alle Knoten mit gleichem Abstand zum Ausgangsknoten als Zwischenknoten zu reservieren, und sucht dann von allen Zwischenknoten aus weiter, bis er erfolgreich zum Zielknoten durchläuft, siehe Abbildung 4.3. Durch Iteration werden alle möglichen kürzesten Wege gefunden und können dann ausgewertet werden“.

³ Vergleich zwischen Standard und verbessert Dijkstra [KSDS21]

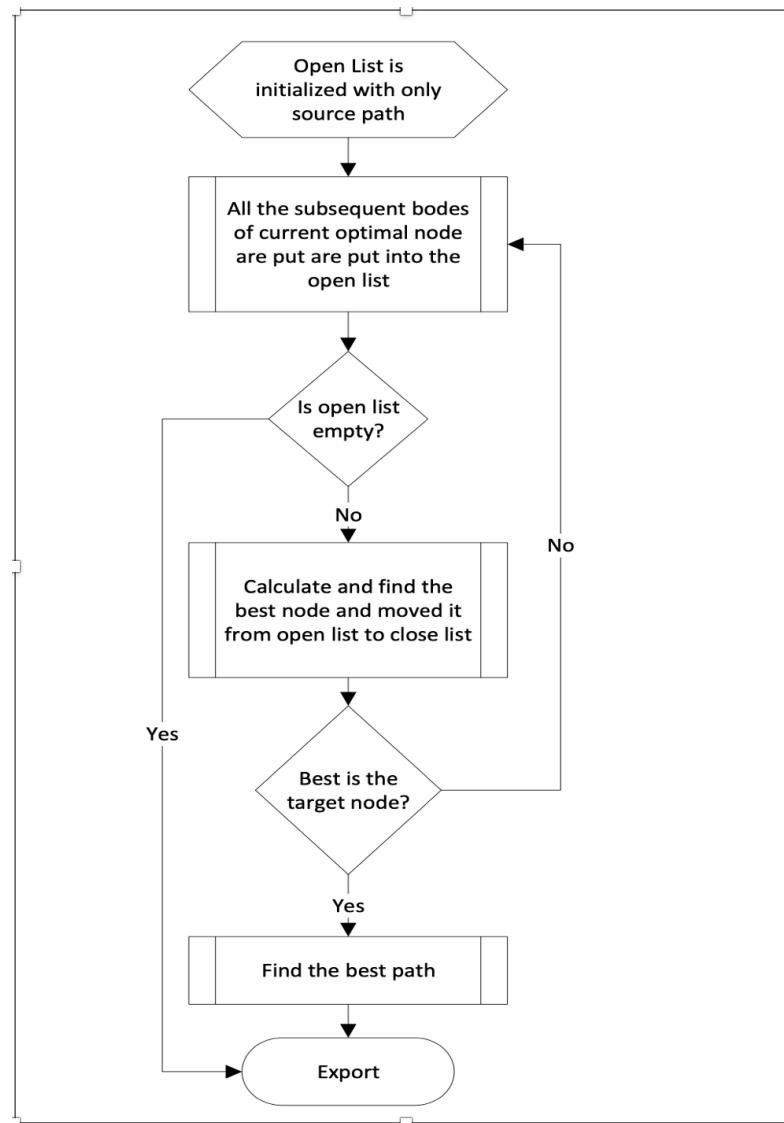


Abb. 4.3. Flussdiagramm des verbesserten Dijkstra für die Pfadplanung[KSDS21].

4.2.2 Verbesserter Dijkstra

Die Autoren des Artikels⁴ haben der verbesserte Dijkstra-Algorithmus so beschrieben: „Der Dijkstra-Algorithmus kann außer den bereits durchlaufenen Knoten keine Daten speichern. Um diesen Nachteil zu überwinden, wird ein Speicherschema eingeführt, das ein mehrschichtiges Wörterbuch implementiert, das aus zwei Wörterbüchern und einer Liste von Datenstrukturen besteht, die in hierarchischer Reihenfolge organisiert sind. Das erste Wörterbuch bildet jeden einzelnen Knoten auf seine Nachbarknoten ab. Das zweite Wörterbuch speichert die Pfadinformationen jedes benachbarten Pfades.“

⁴ verbesserter Dijkstra für Mobile Roboter [KSDS21]

Ein mehrschichtiges Wörterbuch bietet eine umfassende Datenstruktur für den Dijkstra-Algorithmus in einer Innenraumanwendung, bei der die Koordinaten des globalen Navigationssatellitensystems und die Kompassorientierung nicht zuverlässig sind. Die Pfadinformationen in der Datenstruktur helfen dabei, den Grad des Drehwinkels zu bestimmen, den der Roboter an jedem Knoten oder jeder Kreuzung ausführen muss. Der vorgeschlagene Algorithmus liefert den kürzesten Pfad in Bezug auf die Länge und gleichzeitig den navigierbarsten Pfad in Bezug auf den niedrigsten erforderlichen Gesamtdrehwinkel in Grad, der mit dem traditionellen Dijkstra-Algorithmus nicht berechnet werden kann“.

5

Zusammenfassung und Ausblick

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die z.B. Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

Literaturverzeichnis

- AIS⁺20. Samah Abusalim, Rosziati Ibrahim, Mohd Saringat, Sapiee Jamel, and Jahari Wahab. Comparative analysis between dijkstra and bellman-ford algorithms in shortest path optimization. *IOP Conference Series: Materials Science and Engineering*, 917:012077, 09 2020.
- HAMTMA20. Abed Alasadi Hamid Ali, Azizand Mohammed Talib, Dhiya Mohammed, and Abdulmaje Ahmed. A network analysis for finding the shortest path in hospital information system with gis and gps. *Journal of Network Computing and Applications (2020) 5: 10-22 Clausius Scientific Press, Canada*, 2020.
- HG12. Haosheng Huang and Georg Gartner. Collective intelligence-based route recommendation for assisting pedestrian wayfinding in the era of web 2.0. *Journal of Location Based Services*, 6:1–21, 03 2012.
- Jav19. Adeel Javaid. Research on optimal path based on dijkstra algorithms. *SSRN Electronic Journal*, 3, 2019.
- KSDS21. Karthik Karur, Nitin Sharma, Chinmay Dharmatti, and Joshua E. Siegel. A survey of path planning algorithms for mobile robots. *Vehicles*, 3(3):448–468, 2021.
- Mal97. Peter Malte. Replikation in Mobil Computing. Seminar No 31/1997, Institut für Telematik der Universität Karlsruhe, Karsruhe, 1997.
[http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?
document=/ira/1997/31](http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/1997/31).
- MS20. Fadhil Mukhlif and Abdu Saif. Comparative study on bellman-ford and dijkstra algorithms. *International Conference on Communication, Electrical and Computer Networks*, 02 2020.
- VP14. Patel Vaibhavi and Prof. ChitraBaggar. A survey paper of bellman-ford algorithm and dijkstra algorithm for finding shortest path in gis application. *International Journal of P2P Network Trends and Technology (IJPTT)*, 2014.
- ZG13. Minhang Zhou and Nina Gao. Understanding dijkstra algorithm. *Proceedings of the 3rd International Conference on Mechatronics Engineering and Information Technology*, 3, 01 2013.

A

Glossar

DisASTer	DisASTer (Distributed Algorithms Simulation Terrain), A platform for the Implementation of Distributed Algorithms
DSM	Distributed Shared Memory
AC	Linearisierbarkeit (atomic consistency)
SC	Sequentielle Konsistenz (sequential consistency)
WC	Schwache Konsistenz (weak consistency)
RC	Freigabekonsistenz (release consistency)

Arbeitsverteilung

Bitte tragen Sie auf dieser Seite ein, welches Mitglied Ihrer Gruppe welche Inhalte formuliert hat, da die Notengebung für jeden Studenten individuell vorgenommen wird.

Teilnehmer 1:

Inhalte:

Teilnehmer 2:

Inhalte:

Teilnehmer 3:

Inhalte: