Graf-Engelbert-Gymnasium

Mathematik-Facharbeit

Das Problem des kürzesten Weges

Vorstellung und Anwendung des Dijkstra Algorithmus am Beispiel des Weges zwischen der Graf-Engelbert-Schule Bochum und dem Max-Delbrück-Zentrum in Berlin.

Verfasst von Luca Marchetti

Jahrgang Q1

Fachlehrer: Herr Brakensiek

Schuljahr 2019/20

Bochum, den 13.03.2020

Inhaltsverzeichnis

1. Einleitung

Diese Facharbeit befasst sich mit dem Problem des kürzesten Weges und wie man diesen mit mathematischen Methoden bestimmen kann. Als Schwerpunkt soll der kürzeste Weg zwischen der Graf-Engelbert-Schule in Bochum und dem Max-Delbrück-Zentrum in Berlin bestimmt werden.

Der Mensch ist ein stets nach Effizienz strebendes Wesen. Alltäglich versucht er die schnellste, intelligenteste und müheloseste Lösung für alle, ihm entgegen kommenden Probleme zu finden. Genauso steht der Mensch Tag für Tag vor der Frage nach dem kürzesten Weg. Unterbewusst wird ein solches Problem von Menschen algorithmisch gelöst, beispielsweise beim Vergleich von verschiedenen Wegen und den dazugehörigen Längen. Das wohl prägnanteste Exempel wäre das des Weges mit dem Auto von Zuhause bis zur Arbeit oder zur Schule. Kann man den kürzesten Weg also mit einem Algorithmus finden?

Der Dijkstra Algorithmus ist ein Algorithmus der Informatik, basierend auf mathematischen Methoden, durch welchen der kürzeste Weg zwischen zwei Knotenpunkten bestimmt werden kann. Dieser Algorithmus dient unter anderem Google Maps als Grundstein und wird in dieser Facharbeit zur Hilfe und als wichtigstes Werkzeug herangezogen.

Zur Anwendung des Algorithmus wird eine Problemmodellierung mithilfe des Modellierungszyklus vorgenommen, um dann den kürzesten Weg zwischen den genannten Standorten zu bestimmen. Die resultierenden Ergebnisse werden ausgewertet und mit anderen Lösungen verglichen.

Persönlich liegt mir das Thema am Herzen, da ich am MINT-EC Forum 2020 am Max-Delbrück-Zentrum teilnehmen werde. Außerdem bin ich allgemein and Algorithmen und der Funktionsweise von Programmen wie Google Maps stark interessiert, da ich plane Informatik zu studieren.

2. Dijkstra Algorithmus

Der Dijkstra Algorithmus - benannt nach seinem Erfinder, Edsger W. Dijkstra – ist ein Algorithmus zur Ermittlung des kürzesten Weges, ausgehend von einem Startknoten A zu einem Endknoten B (oder allen anderen Knoten) innerhalb eines Graphen (vgl. Esser , 2009, S.4, siehe [4]). Er ist Teil der sogenannten Greedy-Algorithmen, welche die Ermittlung des besten Ergebnisses, in unserem Fall des kürzesten Weges, versprechen (vlg. o.V. , 2015, S.4, siehe [1]). Zur weiteren Erläuterung der Funktionsweise des Algorithmus müssen erst wichtige Begrifflichkeiten erarbeitet werden.

2.1 Graphen

„Graphen sind mathematische Modelle für netzartige Strukturen in Natur und Technik“ (Tittmann, 2003, S.11, siehe [5]).

Eigenschaften und Zusammenhänge von Graphen werden in der Graphentheorie genau untersucht. Der Dijkstra Algorithmus entstammt der mathematischen Graphentheorie und arbeitet mit gewichteten, nicht-negativen und gerichteten Graphen. Diestel (2006) definiert einen Graphen, in seinem Buch zur Graphentheorie, wie folgt:

„Ein *Graph* ist ein Paar G = (V,E) […]. Die Elemente von *V* nennt man *Ecken* (oder *Knoten*) des Graphen *G*, die Elemente von *E* seine *Kanten*. Bildlich kann man *G* darstellen, indem seine Ecken als Punkte zeichnet und zwei dieser Punkte immer dann durch eine Linie verbindet, wenn die entsprechenden Ecken eine Kante sind.“ (S. 2, siehe [3])

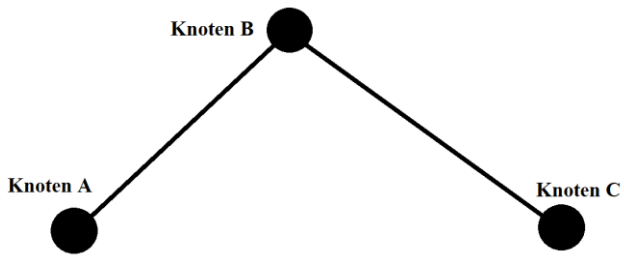


Abbildung 1: einfacher Graph

Folglich besteht ein Graph G aus einer Knotenmenge V und einer Kantenmenge E, wobei Knoten als Punkte und Kanten als verbindende Linie zwischen zwei Knoten, wie in der obigen Abbildung, bildlich dargestellt werden können. Knoten werden als feststehende Punkte behandelt, während Kanten die Beziehung zwischen

Knoten beschreiben. Beim Beispiel des Dijkstra Algorithmus, handelt es sich bei einer Kante um den Weg zwischen zwei Knoten (vlg. o.V. , 2015, S.5, siehe [1]).

2.1.1 Gewichtete Graphen

Als gewichteter Graph, wird ein Graph bezeichnet, dessen Kantenelemente einen reellen Zahlenwert zugeordnet bekommen. Ein solcher Wert wird in der Graphentheorie als Gewicht bezeichnet (vlg. o.V. , 2015, S.5, siehe [1]). Im Sachzusammenhang des kürzesten Weges, könnte ein Kantengewicht zum Beispiel die Strecke zwischen München und Frankfurt in Kilometer sein. Daraus folgend ist die Gewichtung des Graphen essentiell für die Ausführung des Dijkstra Algorithmus, da nur mit Kantengewichten der kürzeste Weg berechnet werden kann. Die Schreibweise eines gewichteten Graphen *G* lautet wie folgt:

$$G = (V, E, c)$$

Die Variabel *c* beschreibt das Gewicht der dazugehörigen Kante aus *E*.

Das Gewicht *c* wird in der bildlichen Darstellung neben die dazugehörige Kante geschrieben.

Wie schon im obigen Teil von Kapitel 2.1.1 erläutert, werden Kanten eines gewichteten Graphens reelle Zahlenwerte, damit auch negative Zahlenwerte, als Gewichte zugeordnet. Negative Gewichte würden bei Algorithmen zur Wegfindung keinen Sinn machen, da es keine negative Wege gibt (vlg. o.V. , 2015, S.5, siehe [1]). Also gilt für alle Gewichte *c* im Falle des Dijkstra Algorithmus $\{c| c \in R, c \geq 0\}$.

2.1.2 Gerichtete Graphen

Gerichtete Graphen sind Graphen mit ausschließlich gerichteten Kanten, die „nur in eine Richtung durchlaufen werden können.“ (Tittmann, 2003, S.126, siehe [5]). Jeder gerichteten Kante *e* aus *E* ist ein Knotenpaar aus *V* zugeordnet:

$$e = (u, v)$$

u ist der Anfangsknoten und v der Endknoten der gerichteten Kante e. Eine gerichtete Kante *e* stellt stehts den kürzesten Weg (kleinstes mögliches Gewicht), von u nach v dar. Hierbei gilt das Gewicht der gerichteten Kante *e* auch nur in dieselbe Richtung (vgl. Tittmann, 2003, S.127, siehe [5]). Wenn ein Weg im Graphen G von Knotenelement *u* nach Knotenelement *v* existiert, nennt man *v* von *u* erreichbar (vgl. Tittmann, 2003, S.127, siehe [5]). Bildlich wird eine gerichtete Kante mit einem Pfeil in die Durchlaufrichtung dargestellt.

Ein Weg innerhalb des Graphen kann mit einer Bogenfolge *P* (Pfad/Path), als „alternierende Folge aus Knoten und gerichteten Kanten des Graphens“ (Tittmann, 2003, S.127, siehe [5]), dargestellt werden:

$$P = \{v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \dots, v_{k-1}, (v_{k-1}, v_k), v_k\}$$

Durch das Aufsummieren der einzelnen Kantengewichten c der Bogenfolge P, kann das ganze Gewicht (länge des Weges) berechnet werden:

$$s(P) = \sum_{k=1}^{n-1} \mathrm{c}\,(v_k, v_{k+1})$$

Gibt es keine Kante (Weg) zwischen zwei Knoten in der Gleichung, erhält man das Ergebnis +∞.

3. Funktionsweise des Dijkstra

Zu Beginn wird der Starknoten A, mit einer Distanz von 0, als aktueller Knoten betrachtet und alle Distanzen zu anderen Knoten aus V werden als unendlich initialisiert. Grundsätzlich wird immer die Kante gesucht, welche die kürzeste Distanz, vom Startknoten A zu einem seiner Nachbarknoten, bildet (vgl. Esser , 2009, S.4, siehe [4]). Nun wird der Nachbarknoten mit der untersuchten Kantendistanz initialisiert und gilt als der bis jetzt bekannte kürzeste Weg vom Startknoten A zum untersuchten Knoten. Daraufhin werden andere Kanten, zwischen Startknoten A und Nachbarknoten, mit höheren Distanzen berücksichtigt, bis jeder Nachbarknoten des Startknotens A mit der entsprechenden Distanz initialisiert wurde. Nachdem alle Kanten zu Nachbarknoten des Startknoten A untersucht wurden, wird der Knoten mit der kürzesten Distanz zum Startknoten zum aktuell betrachteten Knoten. Die neuen Nachbarknoten des aktuell betrachteten Knoten werden mit dem gleichen Vorgehen untersucht. Existiert bereits ein bekannter kürzester Weg zu einem der Nachbarknoten des betrachteten Knotens, wird die Länge des bereits bekannten Weges mit der Länge des neuen Weges verglichen (vlg. o.V. , 2015, S.5, siehe [1]). Der kürzere der beiden verglichenen Wege wird zum neuen bekannten kürzesten Weg vom Startknoten A zum untersuchten Knoten. Dieses Vorgehen wird fortgeführt, bis es keine Kanten mehr zu untersuchen gibt. So wird garantiert, dass alle möglichen Wege zu allen Knoten gefunden werden. Außerdem ist das Finden des kürzesten Weges gesichert, da alle Wege zu einem Endknoten B miteinander verglichen werden, um den kürzesten dieser Wege zu finden. Dabei ist die Anzahl der Schritte abhängig von der Größe des Graphens: „Es zeigt sich, dass in jedem Schritt ein

kürzester Weg zwischen dem Startknoten und einem anderen Knoten im Graphen gefunden wird. Mithin kann der Algorithmus in n(n−1)-Schritten die kürzesten Wege zwischen allen Knoten in einem vollständigen Graphen berechnen.“ (o.V. , 2015, S.7, siehe [1]).

Ein Graph, an dem eine Wegoptimierung mit Dijkstra durchgeführt werden muss, könnte so aussehen:

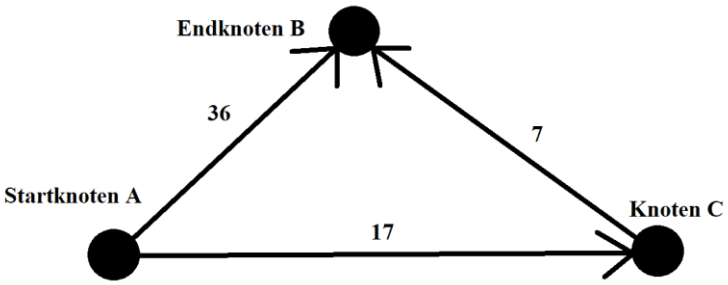


Abbildung 2: gerichteter, gewichteter, nicht negativer Graph

Zunächst wird die Kante (A, C) untersucht, da C der Nachbarknoten mit der Kürzesten Distanz zu A ist. Der kürzeste bekannte Weg zu C vom Startknoten A sind 17. Die Kante (A, B) wird untersucht und der nun kürzeste bekannte Weg zu B ist 36. Als nächstes wird die Kante (C, B) untersucht und die Kantengewichte von Kante (A, C) und (C, B) werden aufsummiert. Die beiden nun bekannten Wege zu Endknoten B werden verglichen. Die zu vergleichenden Pfade P_1 und P_2, aus der obigen Abbildung ,würden lauten:

$$P_1 = \{A, (A, B), B\}$$

$$P_2 = \{A, (A, C), C, (C, B), B\}$$

Wie schon in Kapitel 2.1.2 erläutert, kann die Länge eines Pfades, durch die Aufsummierung aller Kantengewichte auf besagtem Pfad, bestimmt werden.

$$s(P_1) = 36$$

$$s(P_2) = 17 + 7$$

$$s(P_2) = 24$$

$$36 > 24$$

Folglich ist sie Strecke von P_2 kleiner als die Strecke von P_1 und somit ist P_2 der kürzeste Weg von Startknoten A zu Endknoten B. Dieses Verfahren kann auf das

vorliegende Fallbeispiel angewandt werden. Zunächst müssen zur Vorbereitung einige Dinge eingeordnet werden.

4. Problemmodellierung

Der folgende Teil der Facharbeit beschäftigt sich mit der Leitfrage nach dem kürzesten Weg zwischen der Graf-Engelbert-Schule Bochum und dem Max Delbrück Zentrum in Berlin.

Hierbei wird sich auf die erläuterten mathematischen Methoden gestützt. Beginnend mit der Problemmodellierung basierend auf dem mathematischen Modellierungszyklus (siehe Anhang).

Bei der Bestimmung des kürzesten Weges handelt es sich bei der vorliegenden Leitfrage um den Weg mit dem Auto, über Straßen. Die Kantengewichte werden somit in Kilometer angegeben (gekürzt auf erste Nachkommastelle). Zur Beantwortung der Leitfrage muss außerdem eine Modellvereinfachung vorgenommen werden. In anderen Worten, der Graph zur Wegoptimierung wird eingeschränkt und modelliert, um gezielt in den Prozess des Algorithmus eingreifen zu können (Martin Burger, 2007, S.5, siehe [2]). In einer Studienarbeit aus dem Jahr 2015 wird eine Einschränkung des Graphens beschrieben: „Eine Veranschaulichung eines solchen Graphen kann durch eine Vorstellung eines Netzes von verschiedenen Straßen zwischen Orten erfolgen. Die Knotenmenge stellt die Orte dar, wohingegen die Kanten zwischen den Orten die Straßen darstellen.“ (o.V. , 2015, S.5, siehe [1]). Dementsprechend müssen Städte zwischen dem Start und Endknoten als Knoten definiert werden. Die Straßen zwischen den Städten werden zu Kanten und die Strecken die dazugehörigen Kantengewichte. Der Startknoten und Endknoten sind mit Bochum und Berlin vorgegeben. Dazu werden 10 weitere Städte, die zwischen Bochum und Berlin liegen und mindestens 120 000 Einwohnern haben, als Knoten definiert. Die 12 Knoten mit ihrer Bezeichnung lauten:

Bochum Knoten A, Dortmund Knoten B, Münster Knoten C, Paderborn Knoten D, Bielefeld Knoten E, Goslar Knoten F, Braunschweig Knoten G, Hannover

Knoten H, Wolfsburg Knoten I, Magdeburg Knoten J, Potsdam Knoten K, Berlin Knoten L

Basierend auf diesen Knoten, kann ein Netz aus Orten und Straßen in einem Graphen dargestellt werden (o.V. , 2015, S.5, siehe [1]). Der folgende Graph ist auf Grundlage der Deutschen Landkarte, mit den benötigten gerichteten Kanten und Distanzen, erstellt worden:

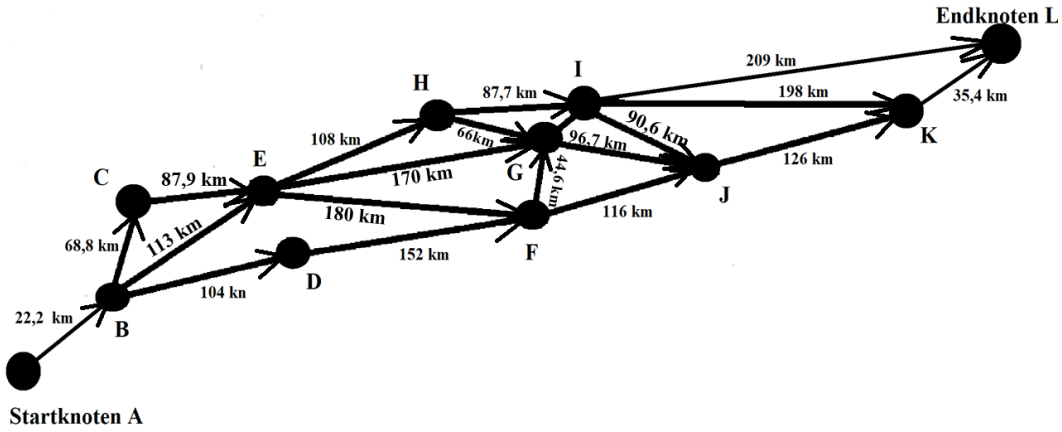


Abbildung 3: Graph zur Problemstellung, ergänzend: Kantengewicht von (G,I) ist c = 31,8 km

Dieser Graph charakterisiert das vorliegende Fallbeispiel mit Kantengewichten, welche aus Google Maps entnommen wurden und die Strecke vom Stadtkern einer Stadt zum Stadtkern einer anderen Stadt darstellen.

Die Adjazenzmatrix A = $[a_{ij}]$, welche mit $a_{ij} = \begin{cases} c_{ij}\ falls\ (i,j) \in E, \\ \infty\ sonst. \end{cases}$ definiert ist (i/j steht für den i-ten/j-ten Knoten), beschreibt den gegebenen Graph näher. In Dieser Matrix sind die Kanten und Kantengewichte, im Gegensatz zu Graphen, übersichtlich dargestellt:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22{,}2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 68{,}8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 104 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 113 & 87{,}9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 152 & 180 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 170 & 44{,}6 & 0 & 66 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 108 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 31{,}8 & 87{,}7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 116 & 96{,}7 & 0 & 90{,}6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 198 & 126 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 209 & 0 & 35{,}4 & 0 \end{pmatrix}$$

In der Obigen Matrix sind 20 Kanten und die zugehörigen Kantengewichte zu erkennen. An einer Stelle an der keine Kante zwischen zwei Knoten existiert, steht eine Null.

5. Ausführung des Dijkstra

Um die Leitfrage abschließend zu beantworten, ist nun der Dijkstra Algorithmus auf das Fallbeispiel anzuwenden. Hierbei wird das in Kapitel 3 geschilderte Verfahren zur Ermittlung des kürzesten Weges verwendet. Bei der Ausführung ist folgender Pfad als kürzester Pfad ermittelt worden:

$$P_{kW} = \{A, (A,B), B, (B,E), E, (E,H), H, (H,I), I, (I,L), L\}$$

Also lauten die angefahrenen Städte, in der obigen Pfadreihenfolge, folgendermaßen:

Bochum, Dortmund, Bielefeld, Hannover, Wolfsburg, Berlin

Durch die, in Kapitel 2.1.2 dargelegte, Aufsummierung der Kantengewichte kann der kürzeste Weg, von der Graf-Engelbert-Schule, Bochum und dem Max Delbrück Zentrum in Berlin, genau bestimmt werden:

$$s(P_{kW}) = \sum_{k=1}^{5} c\ (v_k, v_{k+1})$$

$$s(P_{kW}) = 22{,}2\ km + 113\ km + 108\ km + 87{,}7\ km + 209\ km$$

$$s(P_{kW}) = 539{,}9\ km$$

Die berechnete Route beläuft sich also auf 539,9 km. Zum Abschluss muss das soeben ermittelte Ergebnis mit Daten eines etablierten Routenplaners verglichen werden, um zu determinieren wie akkurat die Problemmodellierung zum Fallbeispiel war. Dazu wird Google Maps benutzt, da die Strecken der einzelnen Kanten auch derselben Quelle entstammen. Google Maps gibt als kürzeste Strecke, von Bochum nach Berlin mit dem Auto, 517 km an. Daraus ergibt sich eine Abweichung von 4,24 %. Wieso diese Abweichung relativ hoch ist, lässt sich mit dem Anfahren der Stadtzentren, welches auf der Problemmodellierung basiert, erklären. Das Anfahren der Stadtzentren generiert bei der vorliegenden Modellierung unnötige Strecke, während die Route von Google Maps ausschließlich über die Autobahn verläuft und so unnötige Strecken vermeidet. Des

weiteren arbeiten etablierte Routenplaner mit weit aus komplexeren Algorithmen, welche viel mehr Möglichkeiten berechnen und in Betracht ziehen als nur 10 Städte als Knotenpunkte.

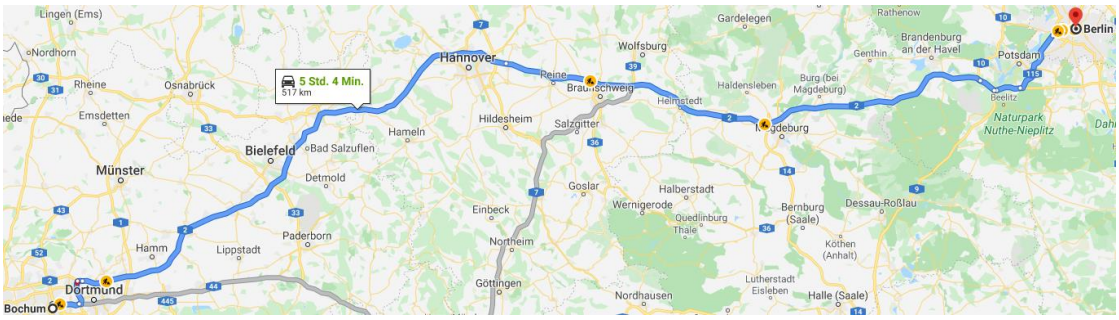


Abbildung 4: Route von Google Maps

Bei der Betrachtung der von Google Maps gestellten Route ist außerdem zu erkennen, dass der erste Teil mit der vorliegenden Modellierung übereinstimmt, da beide Routen über Dortmund, Bielefeld und Hannover verlaufen. Der Rest der beiden Routen decken sich jedoch nicht, weil Google Maps den kürzesten Weg über Magdeburg und Potsdam berechnet, während bei der obigen Ausführung die Route über Wolfsburg als kürzesten Weg berechnet wird.

6. Kritik und Grenzen

Der Dijkstra Algorithmus hilft uns ausnahmslos den kürzesten Weg zu allen Knotenpunkten in einem Graphen zu finden. Jedoch ist dieser nicht frei von Kritik oder Grenzen. Einige davon werden im Folgenden geschildert.

Die erste Grenze ist, dass die Kantengewichte des Dijkstra Algorithmus nicht negativ sein können und so keine Rückwege beschreiben können. Außerdem sind die Kantengewichte nicht dynamisch, was dazu führt, dass der Algorithmus nicht auf potenzielle Änderungen der Strecken, zum Beispiel durch die Renovation einer Straße, reagieren Kann. Die größte Kritik am Dijkstra ist die Berechnung des kürzesten Weges zu allen Knoten, wodurch der Algorithmus sehr ineffizient wird, da durch eine frühere Terminierung des Algorithmus fast alle Kanten in Betracht genommen werden müssen. Der Dijkstra Algorithmus hat bei seiner optimalen Implementierung, bei einem Graphen G=(V,E), eine Laufzeit von $O(|V|\log(|V|) + |E|)$ (Alexander Esser, 2009, S.8). In einer Arbeit zur Implementierung des Dijkstra wird eine „durchschnittliche Laufzeit von $O(n^2m)$“ (Alexander Esser, 2009, S.8)

beschrieben. Diese hohe Ineffizienz würde bei Berechnungen mit weitaus mehr Kanten und Knoten, als bei dem in dieser Arbeit vorliegendem Fallbeispiel, große Probleme bereiten und die Laufzeit stark in die Länge ziehen. Aus diesen Gründen wird zum Beispiel im Modernen Game-Development oft der A* Algorithmus zur Wegoptimierung genutzt, da dieser sehr viel effizienter arbeitet.

7. Fazit und Ausblick

Diese Arbeit beschäftigte sich mit dem Problem des kürzesten Weges und veranschaulichte an einem Beispiel, wie der kürzeste Weg zwischen der Graf-Engelbert-Schule, Bochum und dem Max-Delbrück Zentrum in Berlin, mit dem Dijkstra Algorithmus bestimmt werden kann.

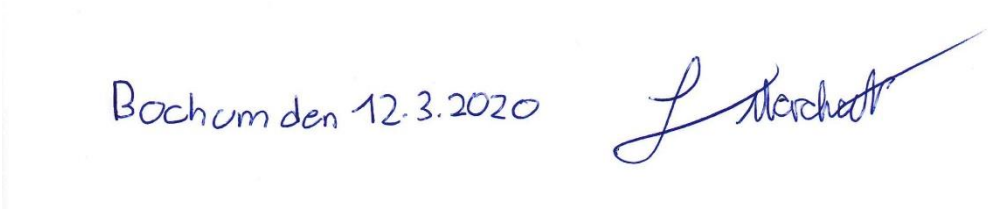
Das Ergebnis bei der Anwendung auf das Fallbeispiel verlief sich auf einer Strecke von 539,9 km. Beim Vergleich dieses Ergebnisses mit etablierten Daten, ist die Fehlerquote nur etwas höher als erwartet. Dies könnte von einem unzureichend ausgearbeiteten Graphen zeugen, welcher eventuell durch mehr Knotenpunkte und Kanten mit dem Algorithmus ein akkurateres Ergebnis erzielen könnte.

Während der Arbeit mit dem Dijkstra Algorithmus fiel mir persönlich mehr und mehr auf, wie wir Menschen alltäglich Algorithmik benutzen, um Prozesse und Probleme algorithmisch zu lösen. Oft passiert dies in der Schule oder gegebenenfalls der Arbeit, zum Beispiel bei der wiederholten Lösung einer Rechnung. Demgemäß ist mir nun viel klarer wie wichtig das algorithmische Vorgehen in unseren Leben ist.

Zuletzt möchte ich beschreiben, wie der Dijkstra Algorithmus spezifisch bei einem modernen Problem hilft. Hierbei handelt es sich um die Limitierung der Treibhausgasemissionen. Der Dijkstra Algorithmus dient als Grundlage zu allen Navigationssystemen und Programmen wie Google Maps, welche uns Tag täglich helfen unnötigen Weg mit dem Auto zu sparen, um damit auch Treibhausgasemissionen zu verhindern. Mit dieser Funktion, ist die Wichtigkeit des Algorithmus auch noch in Zukunft vertreten.

Erklärung

Hierbei erkläre ich, dass ich die Facharbeit ohne Fremdhilfe angefertigt habe und nur die im Literatur- und Quellenverzeichnis aufgeführten Quellen als Hilfsmittel nutzte.



Ort, Datum | Unterschrift

Literaturverzeichnis

[1] o.V.: (2016). *Der Dijkstra-Algorithmus. Ein Algorithmus der Graphentheorie zur Lösung des Kürzesten-Wege-Problem.* Universität Duisburg: GRIN-Verlag.

[2] Burger, M.: (2007). *Mathematische Modellierung*. Universität Münster.

[3] Diestel, R.: (2006). *Graphentheorie.* Universität Hamburg: Springer-Verlag.

[4] Esser, A.: (2009). *Der Dijkstra-Algorithmus zur Berechnung kürzester Wege in Graphen.* Universität zu Köln: GRIN-Verlag.

[5] Tittmann, P.: (2003). *Graphentheorie.* Leipzig: Fachbuchverlag Leipzig.

Internetquellen

*Bild und Streckendaten aus Google Maps entnommen

(URL=https://www.google.de/maps/dir/Bochum/Berlin/@51.9828316,8.0331593,7z/data=!4m14!4m13!1m5!1m1!1s0x47b8e00ff9d35775:0x427f28131548740!2m2!1d7.2162363!2d51.4818445!1m5!1m1!1s0x47a84e373f035901:0x42120465b5e3b70!2m2!1d13.404954!2d52.5200066!3e0) (11.03.2020)

Byrne, Michael: Dieser simple und elegante Algorithmus macht Google Maps erst möglich. Edsger W. Dijkstras erschuf in 20 Minuten die Lösung für ein extrem komplexes Problem.

(URL=https://www.vice.com/de/article/8q8pq3/dieser-simple-elegante-algorithmus-macht-Google-Maps-moeglich-634) (11.03.2020)

Dijkstra, Edsger W.: A note on two problems in Connexion with Graphs.

(URL=http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf) (11.03.2020)

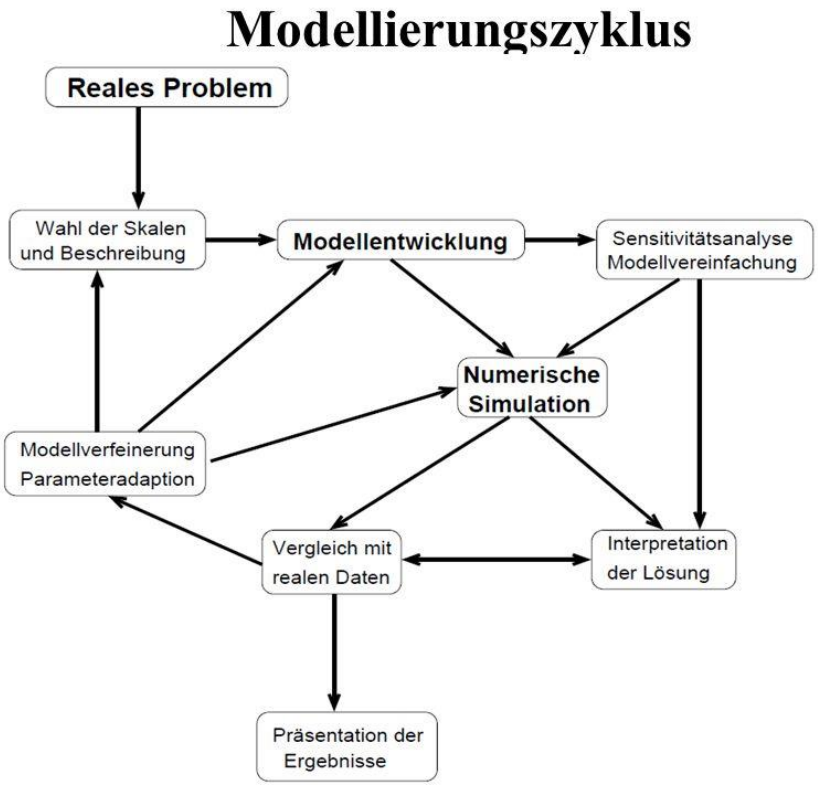
o.V.: Dijkstra: Laufzeit.

(URL=https://crypto.iti.kit.edu/fileadmin/User/Lectures/Algorithmen_SS15/folien_20150622.pdf)

Abbildungsverzeichnis

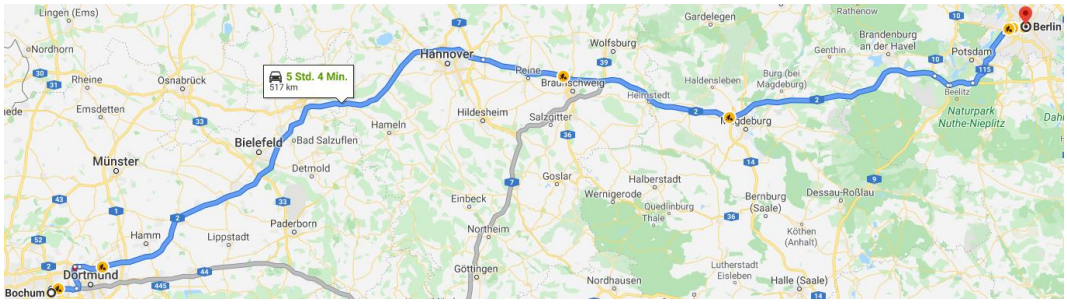
Anhang zu Kapitel 4:

*Quelle zur Abbildung siehe [2]



Anhang zu Kapitel 5:

(URL=https://www.google.de/maps/dir/Bochum/Berlin/@51.9828316,8.0331593,7z/data=!4m14!4m13!1m5!1m1!1s0x47b8e00ff9d35775:0x427f28131548740!2m2!1d7.2162363!2d51.4818445!1m5!1m1!1s0x47a84e373f035901:0x42120465b5e3b70!2m2!1d13.404954!2d52.5200066!3e0) (11.03.2020)



*Alle anderen Abbildungen wurden eigenständig erstellt.