Edmonds und Karp oder Dinic
Ein Laufzeitvergleich

Jan Niklas Hollenbeck
und
Marco Leeske

Hochschule Darmstadt

Abstract. In dieser Arbeit wird das Problem zur Findung des maximalen Flusses in Netzwerken beleuchtet. Für diese Flussprobleme gibt es unterschiedliche Algorithmen, welche auf dem von Ford und Fulkerson basieren. Die vorhandene Literatur geht vor allem auf die theoretische Grundlage und Funktion der einzelnen Algorithmen ein, aber bietet keinen zufriedenstellenden praktischen Vergleich zwischen diesen. Mit dieser Arbeit soll diese Lücke gefüllt werden und damit als Entscheidungshilfe für die Nutzung in der Praxis dienen. Basierend auf dem Algorithmus von Ford und Fulkerson untersuchen wir die beiden optimierten Algorithmen von Edmonds und Karp sowie Dinic. Ein Laufzeitvergleich wird mit Hilfe eines Programmes, welches anhand von Datensätzen die Algorithmen testet, realisiert. Die Test Daten werden so gewählt, dass man die Unterschiede der Algorithmen erkennen, ihre Laufzeit praktisch testen und die jeweiligen Vor- und Nachteile der Algorithmen aufzeigen kann. Anschließend werden die gesammelten Resultate der Laufzeittests verglichen, wodurch der theoretische Vorteil des Algorithmus von Dinic praktisch nachgewiesen wird. Trotzdem bleibt die Frage, welcher Algorithmus bei unterschiedlichen Ausgangssituationen und Erwartungen den Vorzug erhält, dies kommt unter anderem auf den Anwendungsfall und persönliche Anforderungen an.

1 Einleitung

Auf den folgenden Seiten behandeln wir das Flussproblem, welches ein mathematisches Problem zur Findung des maximalen Flusses in Netzwerken beschreibt. Solche Probleme des realen Lebens, beispielsweise in Kanal- oder Verkehrsleitsystemen, werden als gerichtete Graphen modelliert und mittels Algorithmen gelöst. Zur Lösung des Flussproblems gibt es unterschiedliche Algorithmen, welche sich in Laufzeit und Funktion unterscheiden. Diese basieren auf dem Algorithmus von Ford und Fulkerson der den Grundstein für Weiterentwicklungen gelegt hat. Die vorhandene Literatur geht vor allem auf die theoretische Grundlage und Funktion der einzelnen Algorithmen ein, aber bietet keinen zufriedenstellenden praktischen Vergleich zwischen diesen. Mit dieser Arbeit soll diese Lücke gefüllt werden und damit als Entscheidungshilfe für die Nutzung in der Praxis dienen. Basierend auf dem Algorithmus von Ford und Fulkerson untersuchen wir die

beiden optimierten Algorithmen von Edmonds und Karp sowie Dinic. Zwischen diesen werden Laufzeitvergleiche durchgeführt. Diese werden mit Hilfe eines Programmes, welches anhand von Datensätzen die Algorithmen testet, realisiert. Die Test Daten werden so gewählt, dass die Unterschiede der Algorithmen aufgezeigt, ihre Laufzeit praktisch geprüft und die jeweiligen Vor- und Nachteile der Algorithmen ersichtlich werden. Anschließend werden die gesammelten Resultate der Laufzeittests verglichen. Durch die Laufzeittests konnte der theoretische Vorteil des Algorithmus von Dinic praktisch nachgewiesen werden. Die Frage, welcher Algorithmus bei unterschiedlichen Ausgangssituationen und Erwartungen den Vorzug erhält, bleibt weiterhin bestehen, denn dies kommt unter anderem auf den Anwendungsfall und persönliche Anforderungen an.

2 Einführung

Das Flussproblem beschreibt ein mathematisches Problem in Netzwerken.
Flussprobleme können in Netzwerken mithilfe von Graphen modelliert und mittels Algorithmen gelöst oder vereinfacht werden. In den folgenden Zeilen werden die für diese Arbeit nötigen Voraussetzungen erläutert.

2.1 Algorithmen allgemein

Ein Algorithmus ist eine konkrete und eindeutige Handlungsvorschrift, um Probleme oder Klassen von Problemen zu lösen. Beispiele für einfachste Algorithmen können Gebrauchsanweisungen, Rezepte, Bauanleitungen oder Hashfunktionen sein. Wir begegnen Algorithmen im täglichen Leben wie auch bei mathematischen oder informationstechnischen Anwendungen. Algorithmen sind keine neuzeitliche Erfindung, bereits im 9. Jahrhundert beschreibt der arabische Mathematiker Al-Chwarismi (Namensgeber des Algorithmus) Algorithmen. Aus unserem heutigen Leben sind Algorithmen nicht mehr wegzudenken, Navigationssysteme zeigen uns den kürzesten Weg, Smartphones schlagen uns die nächsten zu schreibenden Worte vor oder unsere Texte werden auf Rechtschreibfehler geprüft. Das sind nur wenige von unzähligen Anwendungen, welche auf Algorithmen beruhen. Ein Algorithmus gibt die Vorgehensweise vor, wie Eingabedaten in Einzelschritten in Ausgabedaten umgewandelt werden, um ein bestimmtes Problem lösen zu können. Man spricht im Allgemeinen von Algorithmen, wenn folgende Eigenschaften erfüllt sind:

1. Ausführbarkeit
   Jeder der Einzelschritte eines Algorithmus muss ausführbar sein.

2. Endlichkeit / Finitheit
   Der Algorithmus bzw. dessen Beschreibung muss endlich sein.

3. Eindeutigkeit
   Algorithmen dürfen keine widersprüchliche Beschreibung haben, diese muss eindeutig sein.

4. Terminierung
   Ein Algorithmus muss nach endlich vielen Schritten ein Ergebnis liefern.

5. Determiniertheit
   Bei gleichen Voraussetzungen muss ein Algorithmus stets zum gleichen Ergebnis kommen.

6. Determinismus
   Der Folgeschritt muss immer bestimmt sein. Ein Algorithmus darf zu jedem Zeitpunkt nur maximal einen möglichen Schritt zu Fortsetzung haben.

   [Czernik, 2016]

2.2 Netzwerke

Hinter dem Begriff Netzwerk verbirgt sich ein System, das in unserem Fall mittels Knoten und Kanten dargestellt wird. In dieser Arbeit werden Netzwerke betrachtet, welche sich als mathematische Graphen modellieren lassen. Mithilfe solcher Netzwerke können Problemstellungen aus unserem Alltag so beschrieben werden, dass sie durch Anwendung geeigneter Algorithmen vereinfacht oder sogar gelöst werden können. Hier im speziellen werden wir uns dem (s, t)-Fluss in einem Netzwerk (G, u, s, t) widmen, wobei G einem kantenbewerteten, gerichteten Graph mit den oberen Kapazitäten u entspricht. Ein Knoten s wird als Quelle, sowie ein Knoten t als Senke bezeichnet. Die zwischen Quelle und Senke liegenden Knoten und Kanten können als Zwischenstationen aufgefasst werden. Überdies wird jeder Kante, einer Verbindung von zwei Knoten im Netzwerk, eine Kapazität u (> 0) zugewiesen. Sie gibt an, wie viel maximal durch die Kante fließen kann.

[Reintjes, 2016]

2.3 Gerichtete Graphen

Bei gerichteten, orientierten Graphen bzw. Digraphen werden die Kanten als Pfeile anstelle von Linien dargestellt. Die Pfeile beschreiben die Flussrichtung der Kanten wobei verdeutlicht wird, dass jede der Kanten nur in eine Richtung durchlaufen werden kann.

Der Graph selbst wird als $G = (V, E)$ mit einer Menge V von Knoten und einer Menge geordneter Knotenpaare $E \subseteq V \times V$ von Kanten dargestellt.

Kanten werden als $e = (a, b)$ mit a als Start- und b als Endknoten bezeichnet. Zwei Kanten e_1 und e_2 mit $e_1 = (a, b)$ und $e_2 = (b, a)$ heißen gegenläufig oder antiparallel.

Der Knoten s zeigt den Startpunkt des Flusses. Alle durch das Netzwerk zu transportierenden Mengen starten ihren Fluss an diesem Knoten, mit dem Ziel, den Endknoten t zu erreichen.

In Figure 1 unter 2.3 sieht man die Quelle auf der linken Seite, gekennzeichnet durch "s" und die Senke auf den rechten Seite dargestellt als "t" Des Weiteren sind die Kapazitäten "u" an jeder Kante angegeben.

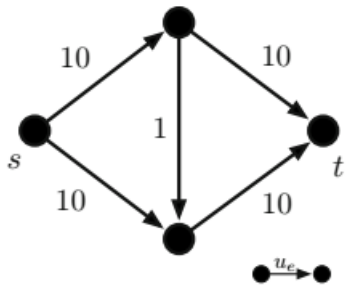


Fig. 1. Bild eines (s, t)-Netzwerkes als gerichteter Graph [Büsing, 2010]

2.4 Algorithmus von Ford und Fulkerson

Der Ford-Fulkerson Algorithmus ist der erste effiziente Algorithmus der in einem Netzwerk (G, u, s, t) den maximalen Fluss f errechnet. Der eigentliche Ablauf des Algorithmus ist sehr einfach: Man sucht einen beliebigen Pfad mit positiven Kapazitäten an allen Kanten von s nach t und bestimmt den maximal möglichen Fluss (geringste Kapazität auf dem Weg von s nach t). Anschließend merkt man sich den Fluss, ändert die Kapazitäten an den Kanten entsprechend des ersten Durchlaufs ab und iteriert diesen Schritt bis kein Weg mehr zu finden ist. Die erhaltenen Flüsse werden aufaddiert und das Ergebnis entspricht dem maximalen Fluss.

Ablauf:

Input: Netzwerk (G, u, s, t). Output: Maximaler Fluss f.

Schritt 1: Setzen Sie $f(e) = 0$ für alle Kanten $e \subseteq E$.

Schritt 2: Bestimmen Sie G^f und $u^f(e)$.

Schritt 3: Konstruieren Sie einen einfachen (s,t)-Weg p in G^f. Falls keiner existiert: STOPP.

Schritt 4: Verändern Sie den Fluss f entlang des Wegs p um $\gamma := min_{e \subseteq p} u^f(e)$.

Schritt 5: Gehen Sie zu Schritt 2

[Büsing, 2010]

2.5 Die Breitensuche

Im Folgenden wird die Breitensuche behandelt, da sie zum Verständnis der Algorithmen von Edmonds und Karp und Dinic benötigt wird. Die Breitensuche (breadth-first search) ist ein Suchalgorithmus für Graphen, der zunächst alle von Ursprung ausgehenden Knoten markiert, bevor die Folgeknoten untersucht werden (siehe Figur 2). Mit ihm ist es möglich, den kürzesten Pfad zwischen zwei Knoten zu finden. In unserem Anwendungsfall wird die Quelle s als Startknoten definiert und von ihr ausgehend alle Pfade zur Senke t gesucht. Der kürzeste Weg wird dann zurückgeliefert.

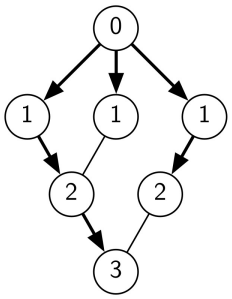


Fig. 2. Die Ebenen werden nacheinander abgearbeitet, wenn ein Knoten bereits besucht wurde, muss dieser nicht noch einmal markiert werden.

2.6 Algorithmus von Edmonds und Karp

Der Algorithmus von Edmonds und Karp 2.6 ist eine Weiterentwicklung des Ford und Fulkerson Algorithmus der 1972 publiziert wurde. Er unterscheidet sich zum Ford Fulkerson durch seine zusätzliche Breitensuche 2.5, welche immer den

kürzesten Weg von s nach t, gemessen an der Anzahl von Kanten, liefert. Entlang dieser Kanten wird der Fluss erhöht. Im Vergleich zum Ablauf in 2.4 ändert sich Schritt Nr. 3 von einem einfachen (s,t)-Weg p in G^f zu dem Kürzesten.

2.7 Algorithmus von Dinic

Dinic's Algorithmus ist dem von Edmonds und Karp sehr ähnlich. Auch hier wird im Gegensatz zu Ford und Fulkerson eine Breitensuche 2.5 durchgeführt. Der Unterschied zu Edmonds und Karp zeigt sich im Umgang mit den gefundenen Wegen. Bei Dinic wird nicht nur der kürzeste (s,t)-Weg p in G^f durchlaufen und berechnet, stattdessen werden große (s,t)-Flüsse aus mehreren kürzesten (s,t)-Wegen zusammengesetzt, anschließend abgearbeitet und der maximale Fluss ermittelt.

3 Testablauf und Prüfungskriterien

Wir führen im Bezug auf die Mathematische Laufzeit eine Evaluierung der genannten Maximal-Fluss Algorithmen durch. Außerdem beschäftigen wir uns mit der praktischen Implementierung der Algorithmen. Zuletzt wird ein Laufzeittest der Algorithmen von Edmons und Karp sowie Dinic anhand von Testdaten durchgefürt. Für die Laufzeittests wurde ein Java Programm an unser Bedürfnisse angepasst, welches den maximalen Fluss errechnet. Für jede Berechnung wird die Zeit gemessen, da bei wenigen Durchläufen eine genaue Zeitmessung unmöglich ist, werden jeweils einhundert Messungen erstellt. Diese bestehen aus einer Million Durchläufe der Algorithmen, von denen der Mittelwert errechnet wird. Erst bei dieser Anzahl an Durchläufen konnte ein konsistentes Ergebnis erzielt werden. Die erhaltenen Ergebnisse werden zur einfacheren Analyse in eine .csv Datei (Comma-separated values) geschrieben. Mit diesen Messungen wird das Verhältnis der Laufzeiten errechnet und auf Plausibilität geprüft.

4 Implementierung und Test

Es wurden beide Algorithmen in unser Programm Implementiert und wir konnten feststellen, dass beide Algorithmen ohne große Probleme implementierbar sind. Allerdings ist Dinic durch seine zusätzliche Berechnung des blockierenden Flusses komplexer, daher ist der Implementierungsaufwand größer.

4.1 Laufzeitvergleich

Die Laufzeiten in der Theorie (Fig. 3) setzen sich aus den inneren und äußeren Schleifendurchläufen zusammen. Die Laufzeit des Ford Fulkerson ist vom gewählten Weg bzw. von der maximalen Kapazität C einer Kante, der Anzahl der Knoten n und Kanten m abhängig. Durch den Einfluss der Kapazität hat der Algorithmus von Ford und Fulkerson einen Nachteil gegenüber den anderen. Deshalb

vergleichen wir in unserem Laufzeittest nur die Algorithmen von Edmonds und Karp sowie Dinic.

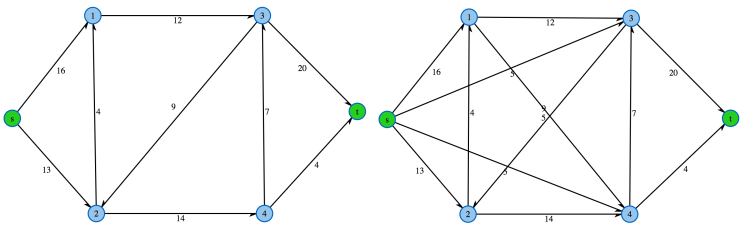
Algorithmus	Laufzeit	Bemerkungen
Ford-Fulkerson	$\mathcal{O}(nmC)$	bei ganzzahligen Kapazitäten
Edmonds-Karp	$\mathcal{O}(nm^2)$	Erhöhung längs kürzester Wege
Dinic	$\mathcal{O}(n^2m)$	Benutzung blockierender Flüsse

Fig. 3. Allgemeine Laufzeiten in der Theorie [Noltemeier, 2009]

Unser Test wurde mit jeweils drei Testpaaren durchgeführt, die sich in ihrer Knotenanzahl unterscheiden. Die einzelnen Paare unterscheiden sich nur in der Anzahl ihrer Kanten. Mit diesem Test zeigen wir, dass der

4.2 Verwendete Graphen

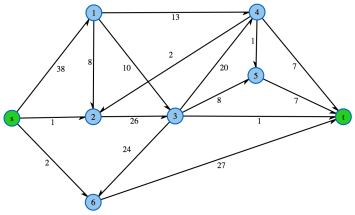
Hier eine bildliche Darstellung der in unserem Test genutzten Graphen.



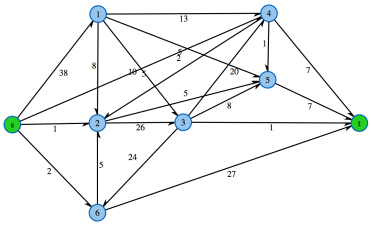
(a) Testgraph 1a, 6 Knoten und 9 Kanten

(b) Testgraph 1b, 6 Knoten und 12 Kanten

Fig. 4. Titel unterm gesamten Bild

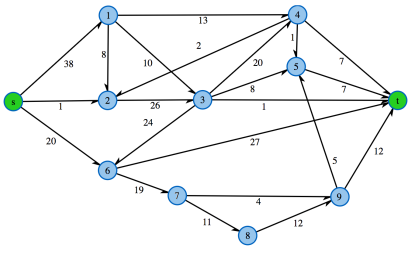


(a) Testgraph 2a, 8 Knoten und 16 Kanten

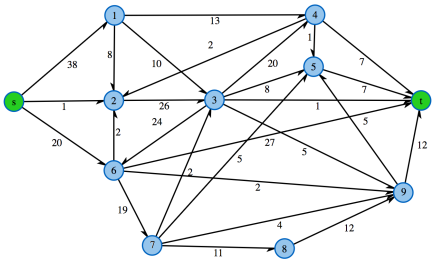


(b) Testgraph 2b, 8 Knoten und 20 Kanten

Fig. 5. Titel unterm gesamten Bild



(a) Testgraph 3a, 11 Knoten und 22 Kanten



(b) Testgraph 3b, 11 Knoten und 27 Kanten

Fig. 6. Die Graphen der Laufzeittests

5 Related Work

Wie schon in der Einleitung 1 erwähnt wurde, konnten wir keine, für uns befriedigende, vorangegangene Arbeit mit einem direkten, praktischen Laufzeitvergleich der beiden Algorithmen von Edmonds und Karp sowie dem von Dinic finden.

6 Zusammenfassung

Anhand der unter 4 durchgeführten Laufzeittests, zeichnet sich der Vorteil des Dinic Algorithmus deutlich ab. Über je mehr Kanten ein Graph verfügt, desto vorteilhafter ist die Nutzung des Dinic Algorithmus. Fairerweise sollte aber gesagt sein, dass der Vorteil erst bei sehr großen Graphen spürbar zum tragen kommt. Der Zeitvorteil des Dinic Algorithmus bewegt sich bei den von uns gewählten Graphen in einem kaum messbaren Bereich, aber die prozentuale Überlegenheit bei steigender Kantenanzahl war deutlich zu erkennen. Dem Algorithmus von Dinic den Vorzug zu geben, ist unserer Ansicht nach erst bei großen Projekten oder sehr großen Graphen lohnenswert. Für die Abwasser oder Kanalberechnung eines Hauses beispielsweise, liegt der zeitliche Vorteil in einem nicht

merklichen Bereich und ist daher zu Vernachlässigen. Sollte aber die Routenberechnung einer Strecke durch das Straßennetz eines ganzen Landes das Ziel sein, bietet der Algoithmus von Dinic deutliche Zeitvorteile. Demnach ist die Entscheidung, welcher Algorithmus gewählt wird, vom Anwendungsfall und den zeitlichen Voraussetzungen abhängig.

Bibliography

Christina Büsing. *Graphen- und Netzwerkoptimierung*, volume 1. March 2010.

Agnieszka Czernik. Was ist ein algorithmus – definition und beispiele, 10 2016. URL `https://www.datenschutzbeauftragter-info.de/was-ist-ein-algorithmus-definition-und-beispiele/`.

Sven Oliver Krumke , Hartmut Noltemeier. *Graphentheoretische Konzepte und Algorithmen*, volume 2. July 2009.

Christian Reintjes. Eine mathematische optimierungsmodell zur statischen anordnung von fachwerktraegern. pages 17–21, April 2016.