Das Flussproblem

Jan Niklas Hollenbeck
und
Marco Leeske

Hochschule Darmstadt

Abstract. In dieser wissenschaftlichen Arbeit wird das Flussproblem beleuchtet. Solche Probleme des realen Lebens werden als gerichtete Graphen modelliert und mittels Algorithmen gelöst. 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 sind so gewählt, dass sie die Worst-Case-Szenarien getestet werden, die jeweiligen Vor- und Nachteile der Algorithmen aufgezeigt und die Implementierbarkeit geprüft wird. 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 wird ein Laufzeitvergleich durchgeführt. Dieser wird mit Hilfe eines Programmes, welches anhand von Datensätzen die Algorithmen testet, realisiert. Die Test Daten werden so gewählt, dass sie die Worst-Case-Szenarien ausgetestet werden und damit ihre Laufzeit praktisch zu prüfen. Es werden die jeweiligen Vor- und Nachteile der Algorithmen aufgezeigt sowie die Implementierbarkeit geprüft. Anschließend werden die gesammelten Resultate der Laufzeittests verglichen. Durch die Laufzeittest konnte der theoretische Vorteil des Algorithmus von Dinic praktisch nachgewiesen werden. 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.

2 Einführung

Das Flussproblem beschreibt ein mathematisches Problem in Netzwerken.

Flussprobleme können in Netzwerken mithilfe von Graphen modelliert werden. Hierbei ist ein Quelle-Senke-Netzwerk(im Folgenden q-s-Netzwerk) ein kantenbewerteter, gerichteter Graph G = (V, E) mit der Eigenheit, dass eine Ecke q als Quelle sowie eine Ecke s als Senke bezeichnet wird. Die zwischen Quelle und Senke liegenden Knoten und Kanten können als Zwischenstationen aufgefasst werden. Überdies wird jeder Kante, also eine Verbindung von zwei Ecken im Netzwerk, eine Kapazität c zugewiesen. Sie gibt an, wie viel maximal durch die Kante fließen kann. [**?**]

2.1 Algorithmus

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.

2.2 Algorithmus von Ford und Fulkerson

Beschreibung und Erläuterung des Ford und Fulkerson Algorithmus

2.3 Algorithmus von Edmonds und Karp

Beschreibung und Erläuterung des Edmond und Karp Algorithmus

2.4 Algorithmus von Dinic

Beschreibung und Erläuterung des Algorithmus von Dinic

2.5 Netzwerke

Unter dem Begriff Netzwerk verbirgt sich ein System, das 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.

2.6 Gerichtete Graphen

Bei gerichteten oder auch 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 $E \subseteq VxV$ von Kanten dargestellt.

Kanten werden als $e = (a, b)$ mit a als Start- und b als Endknoten bezeichnet. Zwei Kanten $e1$ und $e2$ mit $e1 = (a, b)$ und $e2 = (b, a)$ heißen gegenläufig oder antiparallel.

In Figure 1 unter 2.6 sieht man die Senke auf der linken Seite, gekenn-zeichnet durch "S".

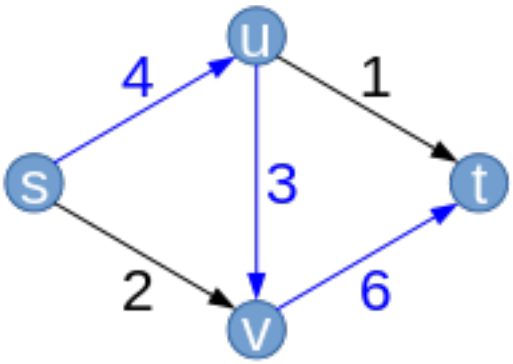

Fig. 1. Bild eines Netzwerk-Graphen

3 Der Inhalt

Kriterien der Evaluierung, Vorstellung des Testaufbaus

4 Experimente

Auflistung Test welche Daten benutzen wir und was wir damit erreichen wollen?

4.1 Laufzeitvergleich

Vergleich der Test Ergebnisse

4.2 Anwendungszenarien der jeweiligen Algorithmen

Die Ergebnisse also wann welcher anzuwenden ist

5 Stand der Technik (Related Work)

5.1 Algorithmen und Datenstrukturen Springer Verlag

Text text text text. Text text text text. Text text text text. Text text text text. Text text text text. Text text text text. Text text text text. Text text text text.

5.2 Graphentheoretische Konzepte und Algorithmen Vieweg und Teubner

Text text text text. Text text text text. Text text text text. Text text text text. Text text text text. Text text text text.

6 Zusammenfassung

Text text text text. Text text text text. Text text text text. Text text text text. Text text text text. Text text text text.

6.1 Ausblick

Erweiterung des Algorithmus von Dinic und Bottleneck erkennen.

Bibliography