Maximaler Fluss in Flussnetzwerken

Proseminar „Theoretische Informatik“, 2020

Maximilian Moeller

I. Motivation

Die Fragestellung nach dem maximalen Fluss ist ein Problem aus der Graphentheorie, das in vielen praktischen Fällen Anwendung findet. Überall dort, wo die Menge einer zwischen verschiedenen Punkten transportierten Ware maximiert werden soll, kann auf die hier entwickelten Lösungen zurückgegriffen werden. Als Beispiele seien die Informationsübertragung in Computernetzwerken oder der Flüssigkeitstransport in Rohrleitungssystemen genannt. Letzteres Beispiel werde ich immer wieder als Analogie zur Veranschaulichung aufgreifen.

II. Grundlagen

Die hier verwendeten Definitionen entstammen Kapitel 26 aus „Introduction to Algorithms“[1].

A. Flussnetzwerke

Ein Flussnetzwerk ist ein gerichteter Graph $G = (V, E)$, in dem jeder Kante $(u, v) \in E$ eine Kapazität $c(u, v) \geq 0$ zugeordnet ist. Des Weiteren werden entgegengerichtete Kantenpaare sowie reflexive Kanten (gleicher Start- und Endknoten) verboten. Jedes Flussnetzwerk enthält zwei besondere Knoten: eine Quelle s und eine Senke t. Schließlich wird noch gefordert, dass jeder Knoten $v \in E \setminus \{s, t\}$ auf einem Pfad $s \to v \to t$ von der Quelle zur Senke liegt.
Ein Flussnetzwerk kann man sich also als Rohrleitungssystem zwischen verschiedenen Stationen vorstellen, von denen eine (und zwar die Quelle s) eine Pumpe und eine (und zwar die Senke t) einen Abfluss besitzt. Die Kapazität $c(u, v)$ entspricht dem Rohrdurchmesser, wobei Wasser zwischen zwei Stationen stets nur in eine Richtung fließt. Dass Wasser dabei nicht von einer Station zur selben Station fließt, entspricht dem Verbot von reflexiven Kanten.

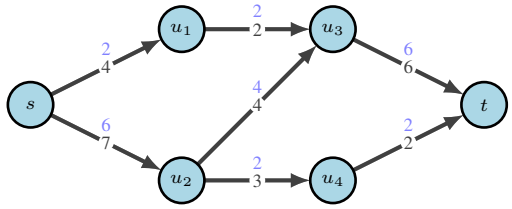

Abbildung 1. Ein Beispiel für ein Flussnetzwerk mit eingezeichneten Kapazitäten (schwarz). Gesucht ist ein maximaler Fluss in diesem Netzwerk. Man sieht auch ohne Algorithmus leicht, dass es einen maximalen Fluss f_{max} gibt mit $|f_{max}| = 8$. Dieser ist in blau über den Kanten angedeutet.

B. Fluss

Ein Fluss in G ist eine Funktion $f \colon V \times V \to \mathbb{R}$, die die folgenden Bedingungen erfüllt:

$$\forall u, v \in V \colon (u, v) \notin E \Rightarrow f(u, v) = 0 \tag{1}$$

$$\forall u, v \in V \colon 0 \leq f(u, v) \leq c(u, v) \tag{2}$$

$$\forall u \in V \setminus \{s, t\} \colon \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v) \tag{3}$$

(1) besagt, dass Wasser nur dort fließen kann, wo auch Rohre liegen. (2) besagt, dass nicht mehr Wasser durch ein Rohr fließen kann, als dessen Kapazität hergibt. (3) besagt, dass aus jeder Station (außer Quelle und Senke) genau so viel Wasser herausfließt, wie hinein. Insbesondere kann sich also in keiner Station zwischen s und t Wasser ansammeln.

Der **Wert** $|f|$ eines Flusses f ist dann definiert als

$$|f| = \sum_{u \in V} f(s, u) - \sum_{u \in V} f(u, s) \tag{4}$$

Der Wert eines Flusses in einem Graphen entspricht in unserer Analogie – ganz intuitiv – der Menge an Wasser, das (netto) aus der Quelle herausfließt. „Netto“, weil es theoretisch ja auch eingehende Kanten (Rohre) zur Quelle geben darf. Dieser Fall kommt in der Praxis jedoch nur selten vor, womit sich der Term zu $|f| = \sum_{u \in V} f(s, u)$ vereinfacht.
Bei einem gegebenen Flussnetzwerk fragt das maximale-Fluss-Problem also nach einem Fluss f, dessen Wert $|f|$ maximal für dieses Flussnetzwerk ist.

III. Lösungsansätze

Es existieren mehrere Algorithmen, die das maximale-Fluss-Problem lösen. Besonders interessant sind dabei die verschiedenen Laufzeiten, die teilweise nicht nur von $|V|$ und $|E|$ sondern auch vom Wert $|f_{max}|$ eines maximalen Flusses f_{max} abhängen können. Viele der bestehenden Lösungen lassen sich anhand ihrer Arbeitsweise in die Kategorien der Ford-Fulkerson- oder Push/Relabel-Algorithmen einordnen. Die Ford-Fulkerson-Algorithmen erhöhen dabei in jedem Iterationsschritt den Fluss global entlang eines Pfades, während die Push/Relabel-Algorithmen stets lokal mit nur einem Knoten und dessen Umgebung arbeiten. Diese beiden Algorithmenfamilien sollen im Rahmen meiner Ausarbeitung näher betrachtet, erklärt und verglichen werden. Dabei möchte ich insbesondere auf die Unterschiede in der Zeitkomplexität eingehen.

Literatur

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 3 ed., 2009.