Einführung in Netzwerk-Fluss-Probleme Das min-cut max-flow Theorem

Felix Meyner
Betreuer: Stephan Günther
Hauptseminar Innovative Internet-Technologien und Mobilkommunikation WS 11/12
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
E-Mail: felix.meyner@mytum.de

KURZFASSUNG

Graphen lassen sich für eine Vielzahl von Problembeschreibungen und deren Lösung einsetzen. In diesem Paper wird der Schwerpunkt auf Flüssen in Netzwerken liegen. Eines der in diesem Kontext auftretenden Probleme ist die Bestimmung des maximal möglichen Flusses in einem Netzwerk. Diese Herleitung wird im Folgenden Schritt für Schritt dargelegt. Ziel dieser Arbeit ist es zu zeigen, dass eine starke Dualität zwischen dem Schnitt des Netzwerkes mit der kleinsten Kapazität und dem maximal möglichen Fluss existiert. Somit können beide Optimierungsprobleme herangezogen werden, um den maximalen Fluss zu berechnen.

Schlüsselworte

Netzwerk-Fluss-Problem, Graphentheorie, Exzess, Fluss, Augmentation, Sättigung, Kapazität, Ford-Fulkerson Algorithmus, min-cut-max-flow, Transformation

1. EINLEITUNG

Problembeschreibungen in Graphen lassen sich generell in zwei Kategorien unterteilen: Maximierungsprobleme und Minimierungsprobleme. Ziel dieses Papers ist es, die Dualität dieser beiden Problemarten am Beispiel des maximal möglichen Flusses aufzuzeigen. Somit soll gezeigt werden, dass der minimale Schnitt eines Netzwerkes dem maximale Fluss entspricht.
Um dies zu beweisen, ist es notwendig zuerst die Grundbegriffe und deren mathematische Herleitung zu erklären, um anschließend diese Dualität mit einem Ringbeweis zu belegen. Darüber hinaus wird ein Überlick über mögliche Transformationen von Problemen zur Vereinfachung gegeben.

2. DEFINITION NETZWERK

Hinführend hierfür werden im Folgenden die wichtigsten Begriffe eingeführt und kurz erklärt, damit ein gemeinsames Verständnis der verwendeten Begriffe geschaffen werden kann.

2.1 Der Graph

Um den Beweis der Dualität führen zu können wird ein Netzwerk als $N = (G, c, q, s)$ definiert. Hierbei ist G ein endlich gerichteter Graph $G = (V, E)$ mit den Knoten $v \in V(G)$, wobei die Knoten $q \in V$ die Quelle und $s \in V$ die Senke des Netzwerks darstellen. Jede gerichtete Kante $e \in E(G)$ verbindet zwei Knoten miteinander. Jede gerichtete Kante $e = (v, w)$, wobei $v, w \in V(G)$, besitzt eine maximale Kapazität $c(v, w) > 0$. Da die Kapazität jeder Kante $c(e) > 0$ ist folgt:

Definition 2.1:
Die Gesamtkapazität eines Graphen ist > 0.

$$\sum_{e \in E(G)} \mathrm{c(e)}{>}0 \qquad \Rightarrow \qquad c(G) \rightarrow \mathbb{R}^+$$

2.2 Fluss

Über jede Kante $e(v, w)$ im Graphen kann ein Fluss $f(v, w)$ fließen. Dieser Fluss muss aufgrund der gerichteten Kante positiver Natur sein. Es ist jedoch möglich, dass eine Kante keinen Fluss besitzt. Somit gilt:

Definition 2.2:
Der Gesamtfluss in einem Graphen ist ≥ 0.

$$\sum_{e \in E(G)} f(e) \geq 0 \qquad \Rightarrow \qquad f(G) \rightarrow \mathbb{R}_0^+$$

Nun führen wir die unter ***2.1*** definierte Kapazität mit der gerade definierten Flussmenge zusammen und definieren Folgendes:

Definition 2.3: Kapazitätskonformität
Der Fluss über eine Kante kann nicht größer sein, als die Kapazität der Kante.

$$\forall (v, w) \in E(G) : f(v, w) \leq c(v, w) \tag{I}$$

Der Vollständigkeit halber wird ebenfalls Folgendes für jede nicht existierende Kante definiert:

$$(v, w) \notin E(G) : f(v, w), c(v, w) = 0$$

Der Fluss einer Kante $f(e)$ wird über eine Rückkante $\overleftarrow{e}$ abgebildet. Für die Rückkante gilt:

$$\overleftarrow{e}(w, v) = f(v, w) : f(e) > 0$$

Da die Knoten des Graphen über Kanten verbunden sind, soll nun definiert werden wie sich Knoten in einem Netzwerk verhalten. Hierzu sind folgende Grundannahmen zu treffen, die für die Knoten $v \in V(G) \setminus \{q, s\}$ des Graphen gelten:

doi: 10.2313/NET-2012-04-1_5

1. Knoten verbrauchen keine Ressourcen.
2. Knoten vermehren keine Ressourcen.
3. Knoten haben keine Möglichkeit Ressourcen zu speichern.

Daraus kann man ableiten, dass der Fluss in einen Knoten dem Fluss aus einem Knoten entsprechen muss. Formal definieren wir nun folgendes:

δ^+als den Fluss in einen Knoten:

$$\delta^+(v) \coloneqq \{e = (w, v) \in E \mid w \in V\}$$

δ^- als den Fluss aus einem Knoten:

$$\delta^-(v) \coloneqq \{e = (v, w) \in E \mid w \in V\}$$

Definition 2.4: Flusserhaltungsbedingung
Der Fluss in einen Knoten entspricht dem Fluss aus einem Knoten, wobei die Quelle und die Senke ausgenommen sind.

$$\forall v \in V(G) \setminus \{q, s\} : \sum_{(w,v)\in\delta^+(v)} f(w, v) = \sum_{(v,w)\in\delta^-(v)} f(v, w) \quad \text{(II)}$$

Die Quelle und Senke nehmen in einem endlichen Graphen eine Sonderrolle ein, da die Flusserhaltung für diese nicht gilt. Der Grund hierfür ist, dass es keine Zirkulation zwischen der Senke und der Quelle gibt. Betrachtet man nun die Senke eines Graphen, so gilt für diese folgendes:

$\delta^+(s) = \sum_{(v,s)\in E(G)} f(v, s)$
$\delta^-(s) = 0$

Es besteht somit ein Überschuss in der Senke. Dieser Überschuss wird als Exzess definiert:

Definition 2.5:
Der Exzess eines Knotens entspricht dem Überschuss des Flusses in diesem.

$$v_{exz} = \delta^+(v) - \delta^-(v)$$

2.3 Residualkapazität

Wenn die Kapazität einer Kante noch nicht gesättigt ist, dies bedeutet dass $c(e) - f(e) > 0$ ist, so bezeichnet man diese Kante als augmentierend. Sie besitzt folglich noch weitere Restkapazitäten die aufgenommen werden können [10].

Allgemein kann man diese Restkapazitäten oder Residualkapazitäten über eine Rückkante $\overleftarrow{e} = (w, v) : f(v, w) > 0$ als $\overleftarrow{e}(w, v) = f(v, w)$ darstellen.

Definition 2.6:
Die Residualkapazität einer Kante entspricht der möglichen Flusserweiterung über diese unter Beachtung von (I) *und* (II)

$$c_f(v, w) = c(v, w) - f(v, w) + f(w, v)$$

2.4 Pfad

Ein Pfad $P(G)$ von der Quelle q zur Senke s beschreibt eine Menge von Kanten, die aneinandergereiht eine der möglichen Verbindungen $p(q, s)$ darstellen. Somit gilt: $p(q, s) \in P(q, s)$. Für Pfade gelten die selben Bedingungen in Bezug auf die Kapazität und den Fluss, wie für die singulär enthaltenen Kanten dieses Pfades. Somit darf der Fluss über einen Pfad nicht größer sein, als die kleinste Kapazität der beinhalteten Kanten. Also gilt:

$$max\, f(p) \leq min\, c(v, w) \in p(q, s)$$

2.5 Schnitt

Als letzter grundlegender Bestandteil wird nun der Schnitt eines Netzwerks definiert. Ein (q,s)-Schnitt $a(G)$ eines Graphen $G = (V, E)$ teilt diesen in zwei Partitionen $(Q, Q - S)$. Es wird festgelegt, dass $q \in Q$ und $s \in S$ sei. Ein Graph besitzt hierdurch eine endliche Anzahl an möglichen Schnitten $A(G)$.

Definition 2.7:
Ein Schnitt $a(G)$ teilt einen Graphen in zwei Partitionen. Diese sind definiert durch (1.) die Knoten der Partitionen und (2.) durch die durchtrennten Kanten des Schnitts [10]:

1. $V_a(Q, S) : Q \cup S = V(G) \wedge Q \cap S = \emptyset$
2. $E_a = \{(v, w) \in E \mid v \in Q, w \in S\}$

Die Kapazität eines Schnitts $a(G)$ bestimmt sich folglich aus der Summe der Kapazitäten der durchtrennten Kanten.

$$c_a = \sum_{v\in S \wedge w\in Q} c(v, w)$$

Analog kann man den Fluss eines Schnitts wie folgend definieren:

$$f_a = \sum_{v\in S \wedge w\in Q} f(v, w)$$

Da die Kapazitätskonformität (*Definition 2.3*) auch hier ihre Gültigkeit behält, darf auch der Fluss eines Schnittes nicht größer sein als seine Kapazität.

$$\forall a \in A : f_a \leq c_a$$

Final kann nun gesagt werden, dass beide Schnitte mit der minimalen Knotenmenge $Q \subset V(G) \setminus \{s\}$ oder $S \subset V(G) \setminus \{q\}$ auf jeden Fall den Fluss im Netzwerk besitzen.

Somit gelten folgende Annahmen:

$$\begin{aligned} f(G) &= \sum_{\exists(q,v)\in E(G)} f(q, v) = \mid q_{exz} \mid \\ &= f(Q) \\ &= \sum_{\exists(v,s)\in E(G)} f(v, s) = s_{exz} \\ &= f(S) \\ &\leq c_a \end{aligned}$$

doi: 10.2313/NET-2012-04-1_5

3. MAXIMALER FLUSS

Durch die bisher gezeigten Grundannahmen lässt sich bereits Folgendes zur Bestimmung des maximalen Flusses, in Form eines schwach dualen Problems, heranziehen:

$$max\, f(G) \leq min\, c_a(G)$$

Ziel ist es jedoch den maximal Fluss nicht als Näherungswert zu bestimmen, sondern diesen exakt zu berechnen. Hierfür kann ein einfacher Ringbeweis genutzt werden. Wenn man folgende Annahmen hierzu heranzieht [2]:

Annahmen 3.1:
Folgende Aussagen sind äquivalent.

1. Es gibt im Netzwerk N einen maximalen Fluss f.
2. Es gibt im Residualnetzwerk N_f keinen augmentierenden Pfad.
3. Der Fluss des minimalen $f_a(Q,S)$ entspricht dem maximalen Fluss $f(G) = f_a(Q,S)$

Beweis 3.2:
(1) ⇒ (2) :
Wenn es in dem Residualgraphen G_f einen augmentierenden Pfad gibt, so ist der Fluss in diesem Residualgraphen nicht maximal, da $f = f + f(G_f)$. Dies widerspricht der *1. Annahme*, dass der Fluss maximal ist.
(2) ⇒ (3) :
Wenn es in einem Residualgraphen keinen augmentierenden Pfad mehr gibt, so existiert ein natürlicher Schnitt in diesem Graphen, der die Quelle und die Senke voneinander trennt.Gekennzeichnet werden die Partitionen durch die gesättigten Kanten.Vergleiche ***Definition 2.7 Punkt 2***. Somit können folgende Annahmen getätigt werden: Der Fluss in diesem natürlichen Schnitt muss der Kapazität dieses Schnitts entsprechen. Formal: $f(G) = f_a(Q,S) = c_a(Q,S)$.
(3) ⇒ (1):
Da der Fluss in jedem Schnitt f_a durch die jeweilige Kapazität c_a nach oben begrenzt ist, muss in mindestens einem Schnitt die Kapazität gleich dem maximalen Fluss sein, da es sonst kein maximaler Fluss wäre, da eine Erhöhung des Flusses möglich ist.

Dieser geführte Beweis ist der Beleg des min-cut max-flow Theorems. Dieses besagt, dass der maximal mögliche Fluss durch einen Graphen dem minimalen Schnitt entspricht.

Definition 2.8 Min-cut max-Flow:
Der maximale Fluss in einem Graphen entspricht der minimalen Kapazität der möglichen Schnitte des Graphen

$$min\, c_a(G) = max\, f(G)$$

Hierzu gilt jedoch anzumerken, dass es auch mehrere minimale Schnitte in einem Graphen geben kann und somit der maximale Fluss einem minimalen Schnitt entspricht.

4. FORD-FULKERSON ALGORITHMUS

Diese drei Annahmen macht sich der Ford-Fulkerson-Algorithmus zu nutze. Entwickelt wurde dieser 1956 von L.R. Ford und D.R. Fulkerson [3]. Der Algorithmus terminiert nach endlich vielen Schritten unter der Bedingung, dass die Kapazitäten der Kanten keine irrationalen Zahlen sind. Falls dies der Fall ist, kann es sein, dass der Algorithmus nicht terminiert. Folgende Annahmen sind aus den Artikeln [4][3] und dem Buch [5] entnommen.

4.1 Formale Beschreibung

Der beschriebene Algorithmus ist ein iteratives Verfahren, um den maximalen Fluss in einem Netzwerk zu bestimmen. Hierfür wird folgender Ablauf verwendet:

Initialisierung: Setze alle Flüsse der Kanten auf ein gültiges Minimum zurück.

$$\Rightarrow f(e) = 0 \text{ für alle } e \in E(G)$$

Solange es augmentierende Pfade von q nach s gibt:

1. Wähle einen beliebigen Pfad $p(q,s)$ wobei: $e(p) \in E(G)$
2. Bestimme die minimale Residualkapazität κ aller $e(p)$
$\kappa := min\{c_f(e) \mid e \in p\}$
3.1 Für alle $e \in p : f(e) = f(e) + \kappa$
3.2 Für alle $\overleftarrow{e} \in p : f(e) = f(e) - \kappa$

Eine wichtige Eigenschaft dieses Algotithmus ist es, dass das Ergebnis des maximalen Fluss deterministisch ist. Jedoch kann der gefundene minimale Schnitt unter der Vorraussetzung mehrere Schnitte mit der selben minimale Kapazität variieren. In diesem Fall hängt der gefundene minimale Schnitt von den gewählten Pfaden ab. Somit eignet sich der originäre Algorithmus nur dazu, den maximalen Fluss eines Netzwerks zu bestimmen.

4.2 Beispiel

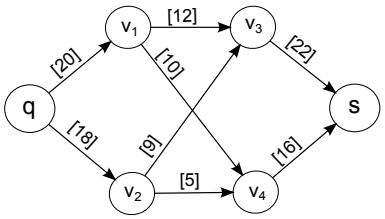

Abbildung 1: Basisgraph

Nun wird der Ford-Fulkerson Algorithmus an einem Beispiel *(Abbildung 1)* gezeigt.
Der Formalität halber wird vor der Quelle q eine virtuelle Superquelle + eingeführt. Diese dient dazu in der Tableaudarstellung den Knoten q korrekt abbilden zu können. Ein weiterer Grund Superquellen einzuführen wird in **Kapitel 5.1.3** aufgeführt.
Es gilt für +:

$$c(+,q) = \infty$$

doi: 10.2313/NET-2012-04-1_5

In der Tableaudarstellung *(Tabelle 1)* wird folgende Notation verwendet:
Es sind unter $V(G)$ alle realen Knoten des Graphen abgebildet.
Für jede Iteration wird eine neue Spalte erzeugt. Diese bildet den gewählten Pfad ab.
Wenn ein Knoten auf dem gewählten Pfad liegt, wird bei diesem der Vorgängerknoten mit der Residualkapazität der verbindenden Kante eingetragen.

(Vorgängerknoten, Residualkapazität der Kante)

Anschließend wird die kleinste Residualkapazität gewählt und die Residualkapazitäten aller Kanten $\in p$ um diesen Fluss verringert. Zuletzt werden die entsprechenden Rückkanten in den Graphen eingetragen. Zu sehen ist dies exemplarisch an *Abbildung 2*, welche den Zustand des Graphen nach der ersten Iteration zeigt.

Tabelle 1: Tableau Ford-Fulkerson-Algorithmus

$V(G)$	Initialisierung	1.Schritt	2.Schritt	3.Schritt	4.Schritt
q	$(+,\infty)$	$(+,\infty)$	$(+,\infty)$	$(+,\infty)$	-
v_1	$(q^+,0)$	$(q^+,20)$	-	$(q^+,8)$	-
v_2	$(q^+,0)$	-	$(q^+,18)$	-	$(q^+,13)$
v_3	$(v_1^+,0)(v_2^+,0)$	$(v_1^+,12)$	-	-	$(v_2^+,9)$
v_4	$(v_2^+,0)(v_1^+,0)$	-	$(v_2^+,5)$	$(v_1^+,10)$	-
s	$(v_3^+,0)(v_4^+,0)$	(v_3^+22)	$(v_4^+,16)$	$(v_4^+,11)$	(v_3^+10)
Δ	-	12	5	8	9
Σ	0	12	17	25	34

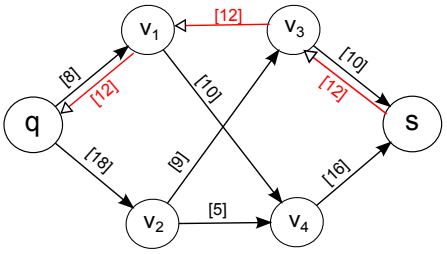

Abbildung 2: Zustand nach 1. Iteration

Initialisierung

Alle Flüsse werden mit einem gültigen Startfluss initialisiert. In diesem Fall ist 0 der gewählte Startfluss.

1. Iteration

Es wird ein beliebiger Pfad p(Q,S) gewählt.
$p_1(q,s) = (q,v_1),(v_1,v_3),(v_3,s)$
Die kleinste Residualkapazizät in p_1 ist auf der Kante $e(v_1,v_3)$ mit 12 Einheiten. Wie auf *Abbildung 2* zu sehen ist, wurden alle Residualkapazitäten um 12 Einheiten verringert und die entsprechenden Rückkanten eingetragen. Die Kante $e(v_1,v_3)$ ist somit saturiert und fällt weg.

2. Iteration

$p_2(q,s) = (q,v_2),(v_2,v_4),(v_4,s)$
$min\, c(p_2) = c(v_2,v_4) = 5$

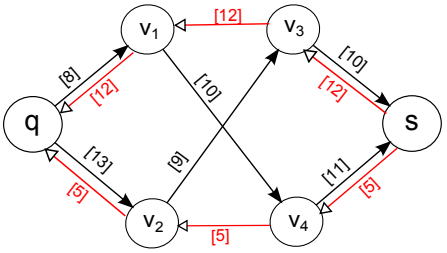

Abbildung 3: Zustand nach 2. Iteration

3. Iteration

$p_3(q,s) = (q,v_1),(v_1,v_4),(v_4,s)$
$min\, c(p_3) = c(q,v_1) = 8$

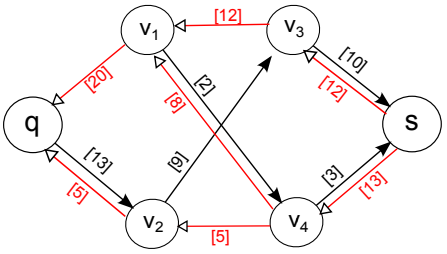

Abbildung 4: Zustand nach 3. Iteration

4. Iteration

$p_4(q,s) = (q,v_2),(v_2,v_3),(v_3,s)$
$min\, c(p_3) = c(v_2,v_3) = 9$

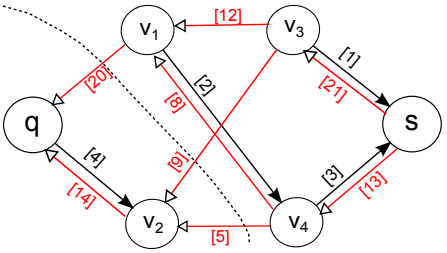

Abbildung 5: Zustand nach 4. Iteration mit gefundenem minimalen Schnitt

Bei Betrachtung der *Abbildung 5* ist erkennbar, dass es keinen weiteren augmentierenden Pfad von der Quelle zur Senke mehr gibt. Der Algorithmus terminiert nun und hat den maximalen Fluss von 34 Einheiten erreicht. Durch die eingetragenen Rückkanten ist ebenfalls der minimale Schnitt klar erkennbar.

In diesem Fall umfasst der minimale Schnitt:

$Q = \{q, v_2\}\ ,\ S = \{v_1.v_3, v_4, s\}$
$E_a = \{(v,w) \in E \mid v \in Q, w \in S\}$

$$f(G) = f(E_a) = \sum_{e \in E_a} f(e) = \sum_{e \in E_a} c(e) = min\, c_a(Q,S) = 34$$

doi: 10.2313/NET-2012-04-1_5

5. GRENZEN UND LÖSUNGSANSATZ

Die Herleitungen und Definitionen wurden bis jetzt bewusst sehr allgemein gehalten, damit ein breiteres Anwendungsgebiet abgedeckt werden kann. Jedoch liegen in der Realität Problemstellungen meist nicht in den vorrausgesetzten Formaten vor.

5.1 Transformationen

Um real auftretende Problemstellungen nun auf diese Verallgemeinerungen zu überführen kann man folgende Transformationen verwenden.

5.1.1. Ungerichtete Kanten

Unter "***2.1 Der Graph***" wurde ein Graph als gerichtet definiert. Dies bedeutet, dass jede Kante eine Flussrichtung besitzen muss. Falls ein Graph jedoch ungerichtet ist, können die Definitionen nicht ohne weiteres übernommen werden. Um die mathematischen Beweise in der einfacheren Form weiternutzen zu können, kann man eine ungerichtete Kante in eine gerichtete Kante überführen. Dies ist auf *Abbildung 6* zu sehen. Die Residualkapazitäten der gerichteten Kanten $c_{f\,gerichtet}(v_1, v_2)$,c($v_2$,$v_1$) verhalten sich zur Kapazität der ungerichteten Kante $c_{f\,ungerichtet}\{v_1, v_2\}$:

$$c_{f\,ungerichtet}(e) = c(e_{ungerichtet}) - \sum_{e \in E_{gerichtet}} f(e)$$

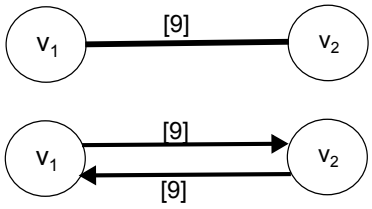

Abbildung 6: Transformation ungerichtete Kante

5.1.2. Kapazitätsbegrenzte Knoten

Falls ein Knoten eine Kapazitätsbegrenzung aufweist, kann dies gelöst werden indem der Knoten v in zwei Knoten v_{in} und v_{out} überführt wird und für die Kapazität gilt, dass [8]:

$$c(v) = c(v_{in}, v_{out})$$

Dies wird in *Abbildung 7* veranschaulicht.

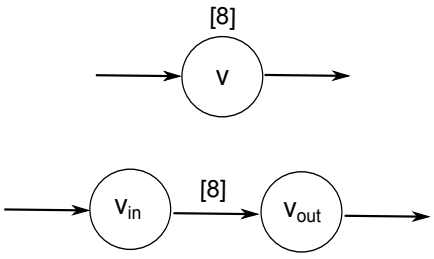

Abbildung 7: Transformation kapazitätsbeschränkter Knoten

5.1.3. Mehrere Quellen und Senken

Ebenso kommt es vor, dass es in einem Netzwerk mehrere Quellen und/oder Senken gibt (*Abbildung 8*).

Diese Problematik kann bei einem homogenen Fluss gelöst werden, indem man eine Superquelle " + ", welche bei den Quellen als Vorgängerknoten ergänzt wird, oder eine Supersenke "−" hinzufügt, welche jeweils die Senken als nachfolgender Knoten erweitert. Die Kapazitäten der neuen Kanten werden auf ∞ gesetzt (*Abbildung 9*).

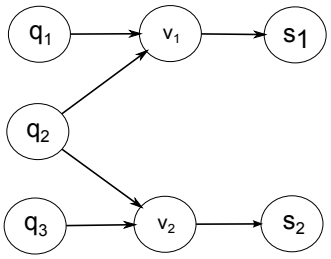

Abbildung 8: Netzwerk mit multiplen Quellen und Senken

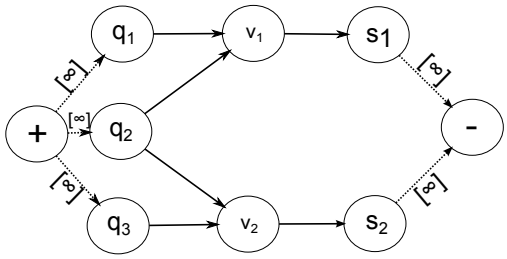

Abbildung 9: Superquellen und Supersenken

5.2 Anwendbarkeit

Die bis hier vorgestellten Methoden lassen sich bisher nur bedingt in der Netzwerktechnik einsetzen. Die meisten Routingprotokolle unterstützen momentan nur solche Methoden, die entweder den kürzesten oder den schnellsten Pfad suchen. Ein Routing über verschiedene Pfade führt, vorallem bei unterschiedlichen Delays der verschiedenen Pfade, zu Problemen, da die Reihenfolge der ankommenden Pakete von entscheidender Wichtigkeit ist.

Eine weitere Grenze der bisherigen Anwendung ist folgender Fall (*Abbildung 10*), in dem es konkurrierende Ströme gibt:
In einem Netzwerk gibt es zwei Quellen q_A und q_B wobei der Index bestimmt, welches Gut von der jeweiligen Quelle verschickt wird. Dazu gibt es zwei Senken die beide Güter benötigen.
Wie zu sehen ist, ist dieses Problem mit den bisherigen Mitteln nicht lösbar, da die Kante (v_1, v_2) die Engstelle darstellt. Unter der Annahme der Flusserhaltung kann der Knoten v_1 nur A oder B erhalten und somit wird eine der Senken nicht mit beiden Gütern versorgt.

Den Lösungsansatz zu diesem Problem bezeichnet man als Netzwerkcodierung [6].
Hierbei werden zuerst die Flüsse aus den Quellen als Informationen interpretiert. Dazu erhalten Knoten die Fähigkeit Informationen zu

doi: 10.2313/NET-2012-04-1_5

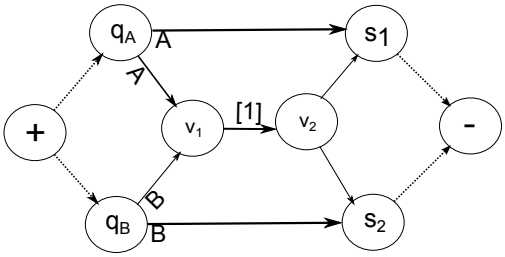

Abbildung 10: Konkurrierende Ströme

codieren und decodieren. In diesem Beispiel ist es der Knoten v_1, der beide Informationen mittels eines XOR Befehls zu $A \oplus B$ codieren kann.
Die Flusserhaltung ist in diesem Fall gegeben, da A und B in den Knoten fließen und $A \oplus B$ den Knoten verlässt. Somit sind beide Informationsflüsse gewährleistet. Die Senken können nun die ihnen fehlende Information aus dem $A \oplus B$ mittels der ihnen vorliegenden nicht codierten Information wieder extrahieren. Das gelöste Problem ist in *Abbildung 11* zu sehen.

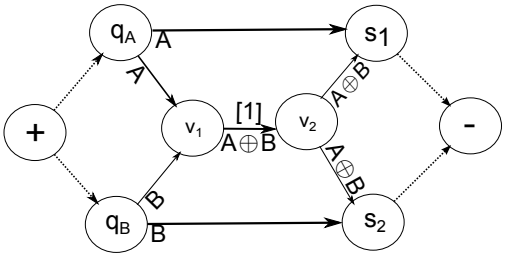

Abbildung 11: Konkurrierende Ströme

6. ZUSAMMENFASSUNG

Zusammenfassend kann man sagen, dass es theoretisch möglich ist den maximalen Fluss in einem Netzwerk zu bestimmen. In der Praxis wird dies auch zum Beispiel in der Logistik und bei homogenen Gütern wie Strom oder Wasser genutzt, um eine bestehende Infrastruktur so effizient wie möglich zu nutzen oder mit minimalem Aufwand zu erweitern.

In der Informationstechnologie liegen meist andere Anforderungen vor, die erfüllt werden müssen. Dies führt dazu, dass die aktuellen Standards mit den jetzigen technischen Anforderungen nicht versuchen den maximal möglichen Fluss zu finden. Der Grund hierfür ist, dass die Verwendung mehrere Pfade relativ problematisch ist, da es zu Auflösungsproblemen kommen kann. Ebenso erfordern die meisten Standards das Pakete in geordneter Reihenfolge ankommen. Würde man nun verschiedene Pfade mit unterschiedlichen Delays nutzen, wäre ein Rechenmehraufwand notwendig um die Pakete zu puffern, sie zu ordnen und vor allem wäre die Fehlerkorrektur erheblich aufwendiger. Da aber die Rechenkapazitäten sehr schnell wachsen ist es abzuwarten, wie sich die verwendeten Protokolle in Zukunft entwickeln werden.

7. LITERATUR

[1] T. H. Cormen, C. E. Leierson, R. Rivest, and C. Stein. *Algorithmen - Eine Einführung*. Oldenbourg, 2010.

[2] S. Dietzel. *Ford-Fulkerson-Methode zur Berechnung des maximalen Flusses in Graphen*. Juni 2005.

[3] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[4] L. R. Ford and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian Journal of Mathematics*, 9:210–218, 1957.

[5] L. R. Ford and D. R. Fulkerson. *Flows in Networks, With a new foreword by Robert G. Bland and James B. Orlin*. Princeton Press, 2010.

[6] S. M. Günther. Optimal cost multicast over coded networks. Ferienakademie Sarntal, 2009.

[7] N. J. Harvey. Comparing network coding with multicommodity flow for the k-pairs communication problem. MIT-LCS-TR-964, 11 2004.

[8] T. H. Thalwitzer. Max-flow min-cut. Master's thesis, Universität Wien, 2008.

[9] L. Trevisan. Lecture 15. the linear programming formulation of maximum cut and its dual. Stanford University CS261: Optimization and Algorithmic Paradigms coursewebsite, 02 2011.

[10] L. Trevisan. Lecture 9. the maximum flow - minimum cut theorem. Stanford University CS261: Optimization and Algorithmic Paradigms coursewebsite, 02 2011.

doi: 10.2313/NET-2012-04-1_5