

Universität Passau

Fakultät für Mathematik und Informatik
(Prof. Dr. F. J. Brandenburg)

Proseminar Algorithmen: SS 2001

Thema: Flüsse in Netzwerken
von Markus Maier

Vortrag gehalten am: 07.06.2001

1. Einleitung

Ziel bei der Betrachtung von Flüssen in Netzwerken ist es, den „Fluss“ verschiedenster Güter durch ein Verbundnetz zu modellieren. Dabei soll zum Zwecke der Optimierung eine Maximierung des „Flusses“ erreicht werden. Angenommen es sei folgende Situation gegeben:

- Ein Netz von Ölleitungen unterschiedlicher Abmessungen
- Ein Ölfeld als Quelle und eine Raffinerie als Bestimmungsort (Ziel)
- Die Ölleitungen sind auf komplexe Weise untereinander verbunden und in den Verzweigungspunkten steuern Schieber die Richtung des Flusses.

Die Frage ist nun, welche Schieberstellungen in den Verzweigungspunkten gewährleisten eine Maximierung des Ölflusses von der Quelle zum Bestimmungsort? Dabei handelt es sich um eine der interessantesten Anwendungen der Graphentheorie, denn offensichtlich eignen sich gewichtete gerichtete Graphen zur Modellierung derartiger Probleme. Der gleiche Ansatz eignet sich, um den Informationsfluss in Computernetzwerken (Datenpakete „strömen“ bei der Übertragung von Information), Fluss von elektrischem Strom in einem Leitungsnetz, Verkehrsfluss auf Autostraßen oder den Materialfluss durch Betriebe zu beschreiben. Die Kantengewichte repräsentieren dabei die jeweilige Kapazität der Leitung (Liter/Sekunde), der Straße (Anzahl der Fahrzeuge/Minute) etc. Zu dem Maximalflussproblem gibt es außerdem zahlreiche Varianten von denen einige im Grenzgebiet zwischen Informatik und dem Operations Research liegen.

2. Flüsse in Netzwerken

Definition: Netzwerk

Ein (Fluss-)Netzwerk $N = (G, q, s, c)$ ist ein gerichteter Graph $G = (V, E)$ mit zwei ausgezeichneten Knoten $q, s \in V$, der Quelle und der Senke, sowie einer Funktion $c : V \times V \rightarrow \mathbb{N}$, wobei für $(u, v) \notin E$ gilt $c(u, v) = 0$.

Der Wert $c(u, v)$ repräsentiert die Kapazität der Kante.

Definition: Fluss

Eine Funktion $f : V \times V \rightarrow \mathbb{Z}$ heißt zulässiger Fluss auf N gdw. die folgenden Bedingungen erfüllt sind:

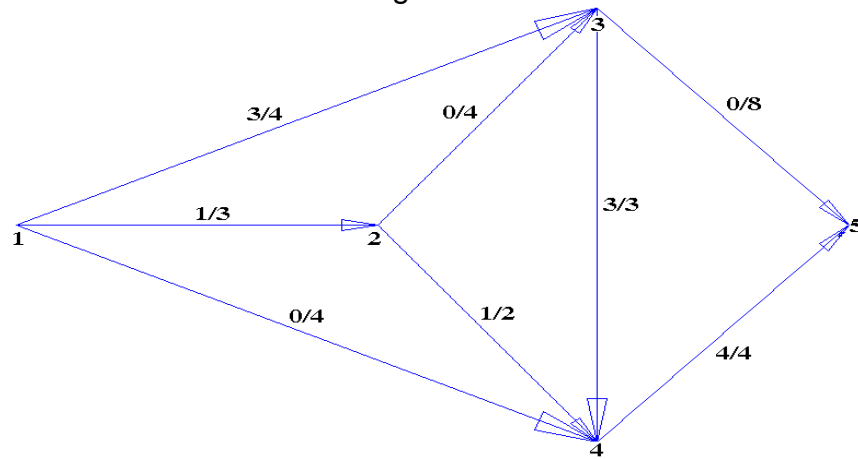
1. Kapazitätsbedingung
 $\forall (u, v) \in V \times V: f(u, v) \leq c(u, v)$
2. Symmetriebedingung
 $\forall (u, v) \in V \times V: f(u, v) = -f(v, u)$
3. Flusserhaltung
 $\forall u \in V - \{q, s\}: f(u, V) := \sum_{v \in V} f(u, v) = 0$

Der Betrag eines Flusses f ist: $|f| := \sum_{v \in V} f(q, v)$.

Der Betrag des Flusses stellt also die Flussmenge dar, die aus der Quelle fließt. Das Ziel besteht darin einen zulässigen Fluss f auf N zu finden, so dass $|f|$ maximal ist.

Folgendes Beispiel zeigt ein Netzwerk N mit zulässigem Fluss f.

Denn für alle Kanten ist der Flusswert kleiner gleich dem Kapazitätswert, Symmetriebedingung gilt nach Definition und für alle inneren Knoten (2,3,4) gilt offenbar Flusserhaltung.



Man sieht in diesem Netzwerk sehr leicht Wege auf denen der Fluss erhöht werden kann, ohne dabei eine der Flussbedingungen zu verletzen.

Ziel ist es nun, derartige Wege im Netzwerk schneller zu finden. Man erkennt, dass die Restkapazität einer Kante (=Kapazität der Kante – Flusswert) der Betrag ist, um den der Fluss auf dieser Kante gerade noch erhöht werden kann, wenn die Kapazität nicht überschritten werden soll.

Aus diesem Grund ist es sinnvoll, das Konzept des Restnetzwerks einzuführen.

Definition: Restnetzwerk

Gegeben sei ein Netzwerk $N = ((V,E),q,s,c)$ mit darauf zulässigem Fluss f.

Das zugeordnete Restnetzwerk ist definiert als $N_f := ((V,E_f),q,s,c_f)$.

Dabei ist c_f die zugeordnete Restkapazität, mit $c_f(u,v) = c(u,v) - f(u,v)$ (Restkapazität der Kante (u,v)) und $E_f = \{ (u,v) \mid c_f(u,v) > 0 \}$.

In einem Restnetzwerk lassen sich nun wesentlich leichter (vorwärtsgerichtete) Pfade von der Quelle zur Senke ermitteln. Hat man solch einen Pfad gefunden, so ist das Minimum der Restkapazitäten der Kanten (des Pfads), gerade der Betrag um den der Fluss in dem eigentlichen Netzwerk entlang des entsprechenden Wegs erhöht werden kann. Ein solcher Pfad wird Erweiterungspfad genannt. Der entsprechende Weg im ursprünglichen Netzwerk heißt zunehmender Weg.

Definition: Erweiterungspfad / zunehmender Weg

Ein Erweiterungspfad ist ein zyklenfreier Pfad p in N_f von q nach s ($N_f = ((V,E_f),q,s,c_f)$). Ein Erweiterungspfad p in N_f entspricht einem zunehmenden Weg in $N = ((V,E),q,s,c)$.

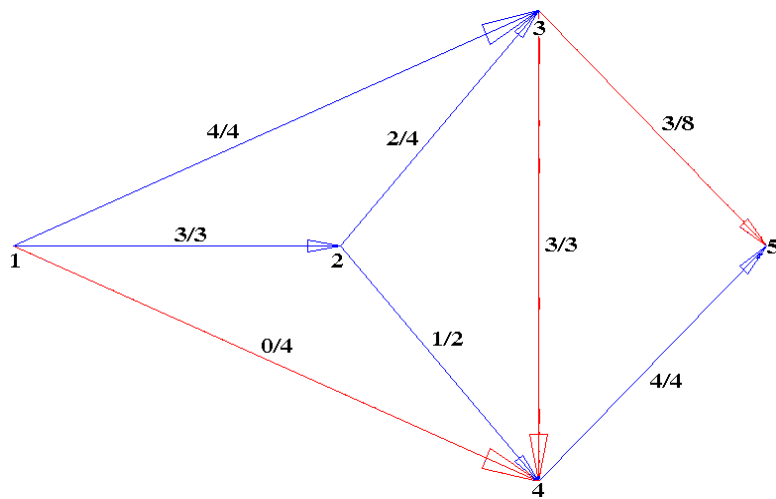
Definition: Restkapazität eines Erweiterungspfads p

Die Restkapazität eines Erweiterungspfads p ist $c_f(p) = \min\{c_f(u,v) \mid (u,v) \text{ liegt auf } p\}$

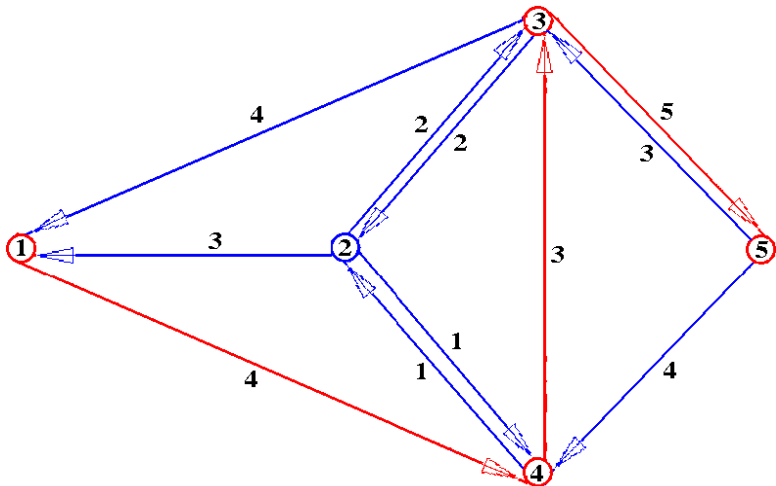
Offenbar ist $g(u,v) = \begin{cases} c_f(p), & \text{falls } (u,v) \text{ auf } p \text{ liegt} \\ 0, & \text{sonst} \end{cases}$ ein zulässiger Fluss auf N_f .

Die folgenden Abbildungen veranschaulichen das bisher Gesagte:

Nebenstehende Skizze zeigt ein Netzwerk mit zulässigem Fluss f , sowie einen zunehmenden Weg p (1-4-3-5) von der Quelle (1) zur Senke (5).



Aus obigem Netzwerk lässt sich nun leicht das zugehörige Restnetzwerk N_f konstruieren (rechts). Alle Pfade von der Quelle (1) zur Senke (5) sind dabei Erweiterungspfade (z.B. 1-4-3-5). Setzt man in dem Restnetzwerk gemäß obiger Definition $g(u,v) = 3$ für Kanten auf diesem Pfad 1-4-3-5 und $g(u,v) = 0$ sonst, dann ist g ein zulässiger Fluss in N_f . Offenbar ist 3



genau der Wert, um den der Fluss entlang des entsprechenden zunehmenden Wegs in dem eigentlichen Netzwerk erhöht werden kann.

Mit Hilfe der Definition des Flusses und Restnetzwerks kann man ziemlich leicht das folgende Lemma beweisen.

Lemma:

Sei $N = (G,q,s,c)$ ein Netzwerk, f ein darauf zulässiger Fluss, N_f das zugehörige Restnetzwerk und g ein zulässiger Fluss auf N_f .

Dann gilt: $(f+g)$ ist ein zulässiger Fluss auf N mit $|(f+g)| = |f| + |g|$.

Man kann also den Fluss immer dann erhöhen, wenn es noch Erweiterungspfade im Restnetzwerk gibt. Die Frage ist nun, ob sich der Fluss auch vergrößern lässt, wenn sich keine mehr finden lassen. Um dies zu klären, muss man den Fluss über einen Schnitt des Netzwerks betrachten.

Definition: Schnitt eines Netzwerks N

Ein Schnitt (X,Y) ist eine Zerlegung der Knotenmenge $V=X \cup Y$, $X \cap Y = \emptyset$, mit $q \in X$ und $s \in Y$.

Die Kapazität eines Schnitts (X,Y) ist $c(X,Y) := \sum_{x \in X} \sum_{y \in Y} c(x,y)$.

Sei f ein Fluss.

Dann ist der Fluss über den Schnitt (X,Y) definiert durch $f(X,Y) := \sum_{x \in X} \sum_{y \in Y} f(x,y)$.

Das nachfolgende Lemma lässt sich aus der Definition des Flusses leicht beweisen.

Lemma:

Für jeden Schnitt (X,Y) in einem Netzwerk $N = ((V,E),q,s,c)$ mit Fluss f gilt:

$$f(X,Y) = |f|.$$

Offensichtlich gilt nun $|f| = f(X,Y) \leq c(X,Y)$ und somit für jeden zulässigen Fluss f und jeden Schnitt (X,Y) : $|f| \leq c(X,Y)$

Anders ausgedrückt: $\max\{|f| : f \text{ Fluss auf } N\} \leq \min\{c(X,Y) : c(X,Y) \text{ Schnitt}\}$

Daraus ergibt sich der zentrale Satz, auf den der Algorithmus von Ford und Fulkerson beruht, das sogenannte Max-Flow-Min-Cut-Theorem.

Max-Flow-Min-Cut-Theorem:

Sei f ein zulässiger Fluss in dem Netzwerk $N=(G,q,s,c)$. Dann sind die folgenden Aussagen äquivalent:

- (1) f ist ein maximaler Fluss auf N
- (2) Das Restnetzwerk N_f enthält keinen Erweiterungspfad,
- (3) Es gilt $|f| = c(X,Y)$ für einen Schnitt (X,Y) von N

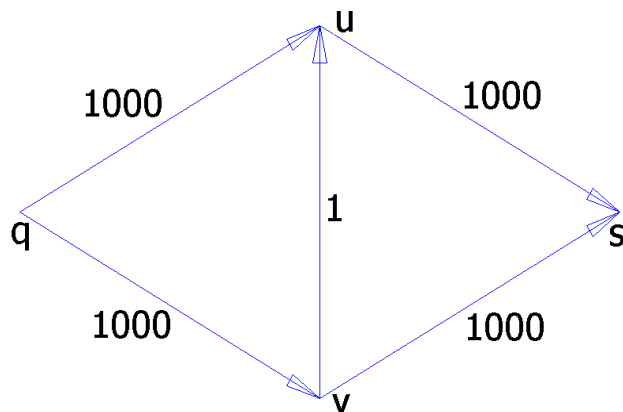
3. Ford-Fulkerson Algorithmus

Der Algorithmus von Ford und Fulkerson zur Lösung des Maximalflussproblems lautet wie folgt:

```
// Initialisiere Nullfluss
for all (u,v) ∈ E {
    f(u,v) = 0;
    f(v,u) = 0;
}
// iterierte Flussvergrößerung
while (es gibt einen zunehmenden Weg p in N) {
    r = c_f(p); // Restkapazität von p
    erhöhe f entlang p um r;
}
// f ist ein maximaler Fluss auf N
```

Eigentlich handelt es sich dabei nicht um einen Algorithmus, da in keiner Weise angegeben wird, wie man einen zunehmenden Weg in N findet bzw. falls mehrere Wege zur Auswahl stehen, welchen man wählen sollte. Dies ist aber entscheidend für die Laufzeit wie das nebenstehende Beispiel zeigt.

Wählt man abwechselnd die Wege $q-v-u-s$ und $q-u-v-s$ als zunehmende Wege,



so benötigt der Ford-Fulkerson Algorithmus 2000 Iterationen. Der maximale Fluss kann aber durch Wahl der Wege q-u-s und q-v-s bereits nach 2 Iterationen erreicht werden. Da der Betrag des Flusses bei jedem Auffinden eines Erweiterungspfades echt zunimmt und das Auffinden eines solchen Pfads mit BFS oder DFS in $O(|E|)$ durchgeführt werden kann, ist die Gesamtkomplexität auf $O(f^* |E|)$ begrenzt (f^* ist maximaler Fluss).

Der Betrag des maximalen Flusses ist ungünstigerweise in der Komplexität enthalten. Durch die Edmonds-Karp-Strategie, welche besagt, dass man nur kürzeste Erweiterungspfade wählen soll, kann dies vermieden werden. Derartige Pfade können mit BFS in $O(|E|)$ gefunden werden. Dann sind nie mehr als $|V| |E|$ Schleifendurchläufe nötig, wodurch die Gesamtkomplexität auf $O(|V| |E|^2)$ beschränkt ist.

Die Begründung lautet dabei wie folgt:

Die Pfadlängen der Erweiterungspfade bleiben in jedem Schleifendurchlauf gleich oder nehmen echt zu. Die Pfadlängen können höchstens $2|E| = O(|E|)$ Schritte gleich bleiben (Der Schichtengraph des Restgraphen hat zu Beginn höchstens $2|E|$ viele Kanten). Bei jedem Auffinden eines Erweiterungspfades wird mindestens eine Kante entfernt. Mögliche Pfadlängen sind Zahlen zwischen 1 und $|V| - 1$. Insgesamt sind also nie mehr als $O(|V| |E|)$ Schleifendurchläufe möglich.

Der folgende Ausschnitt zeigt den wesentlichen Teil einer Java-Implementierung des Ford-Fulkerson Algorithmus mit BFS und Adjazenzmatrix.

```
private static int sink; // Anzahl der Knoten bzw.
                        // Nummer der Senke

static void createFields(int maxint) {
    c = new int [maxint+1][maxint+1]; // Kapazitätsmatrix
    f = new int [maxint+1][maxint+1]; // Flussmatrix
    val = new int [maxint+1]; // Knotenwert
    dad = new int [maxint+1]; // Vaterknoten im Pfad
    reached = new int [maxint]; // Liste der erreichten Knoten
    temp = new int [maxint]; // temporäre Liste mit Knoten
}

public static void main(String[] args) throws IOException {
    Start();
    sink = Input.getNumberOfVertices();
    createFields(sink);
    init_c();
    init_f();
    do {
        newPath();
    } while (!isEmpty(reached));
    Ausgabe();
}

static void newPath() throws IOException {
    init_reached();
    init_val();
    init_dad();
    do {
        getAvailables();
    } while ((!isElementOfList(reached,sink)) && (!isEmpty(reached)));
    // Pfad gefunden oder es gibt keinen möglichen Pfad mehr
    if (!isEmpty(reached)) {
        int x,y;
        y = sink;
    }
}
```

```

do {
  x = dad[y];
  f[x][y] += val[sink];
  f[y][x] = - f[x][y];
  y = x;
} while (y != 1);
}

```

Die Breitensuche startet von der Quelle aus. Neue Schichten werden mit der Methode „getAvailables()“ solange konstruiert, bis es entweder keine neue Schicht mehr gibt, da der Fluss bereits maximal ist, oder bis die Senke erreicht wird. Während der Pfadkonstruktion wird jedem neu gefundenen Knoten x ein Wert $val(x)$ zugeordnet, der angibt um wie viel der Fluss bis dahin erhöht werden kann. Die Funktion „dad“ weist jedem Knoten im Pfad einen eindeutigen Vaterknoten zu. Bei Erreichen der Senke wird von dort aus bis zur Quelle „zurückgegangen“ („dad“) und der Fluss auf jeder Kante des Pfads erhöht.

4. Varianten des Maximalflussproblems

Zu dem eigentlichen Maximalflussproblem gibt es z.B. noch folgende Varianten:

- a) Netzwerke mit mehreren Quellen q_1, \dots, q_n und Senken s_1, \dots, s_m
Dieses Problem kann auf ein Maximalflussproblem mit einer Quelle und einer Senke zurückgeführt werden:
 - Konstruktion einer Hilfsquelle q sowie einer Hilfssenke s
 - Gerichtete Kanten von q nach q_i bzw. von s nach s_i mit unendlicher Kapazität
 - Konstruktion eines Maximalflusses auf dem neuen Netzwerk
 - Entfernen der Hilfskanten und Hilfsknoten
- b) Kapazitätsbegrenzung für Knoten
(Lösung: Ersetze alle Knoten v durch Kante $v' \rightarrow v''$)
- c) Maximaler Fluss mit minimalen Kosten
Einführung einer Kostenfunktion für Kanten $cost: E \rightarrow \mathfrak{R}$
Die Kosten einer Flußfunktion sind $cost(f) = \sum_{e \in E} cost(e)f(e)$.
- d) Einführung von unteren Schranken für die Werte des Flusses durch die Kanten
- e) Flussproblem für mehrere Medien Einführung von mehreren Quellen, Senken und Materialien

5. Literaturverzeichnis

- [1] U. Schöning : Algorithmen – kurz gefasst, Spektrum
- [2] R. Sedgewick : Algorithmen, Addison-Wesley
- [3] A. Brandstädt : Graphen und Algorithmen, Teubner
- [4] T. H. Cormen : Introduction to Algorithms, The MIT Press
- [5] J. Clark, D. A. Holton: Graphentheorie: Grundlagen und Anwendungen, Spektrum