1 Fluss in Graphen

Es sei ein gerichteter Graph $G = (V, E)$ gegeben. Jeder Kante e des Graphen sei eine *Kapazität* $c(e) \in \mathbb{N}$ zugeordnet. Weiter seien zwei Knoten des Graphen ausgezeichnet: eine *Quelle* s (engl. *source*) und eine *Senke* t (engl. *target*). Man will nun möglichst viel *Fluss* von s nach t schicken, ohne die Kantenkapazitäten zu überschreiten. Dabei muss für jeden Knoten außer s und t gelten, dass genauso viel Fluss in den Knoten hinein fließt wie aus ihm heraus.

Genauer ist ein *gültiger Fluss* von s nach t eine Abbildung $f : E \to \mathbb{R}$, die für jede Kante des Graphen den zugehörigen Fluss angibt und folgende Bedingungen erfüllt:

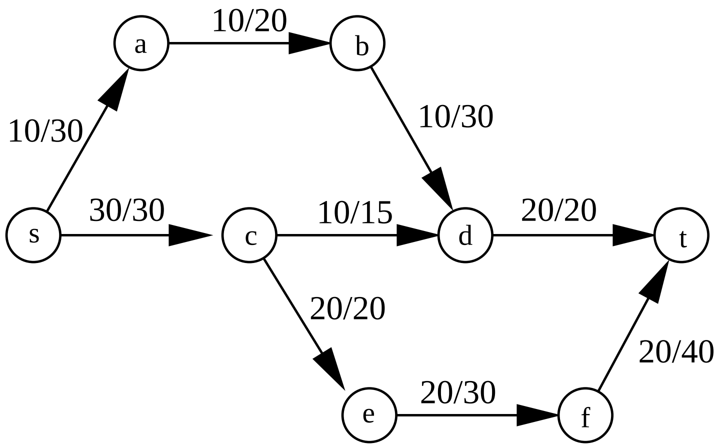
- Einhaltung der Kapazitäten: $\forall e \in E : \ 0 \le f(e) \le c(e)$
- Flusserhaltung: $\forall v \in V \setminus \{s, t\} : \ \sum_{e=(u,v)\in E} f(e) = \sum_{e=(v,w)\in E} f(e)$

Der (Gesamt-)Fluss F von f ist der Fluss, der insgesamt von s nach t gelangt, also:

$$F = \sum_{e=(s,w)\in E} f(e) - \sum_{e=(u,s)\in E} f(e) = \sum_{e=(u,t)\in E} f(e) - \sum_{e=(t,w)\in E} f(e)$$

Ein *maximaler Fluss* ist ein Fluss, der unter allen gültigen Flüssen den Wert F maximiert.

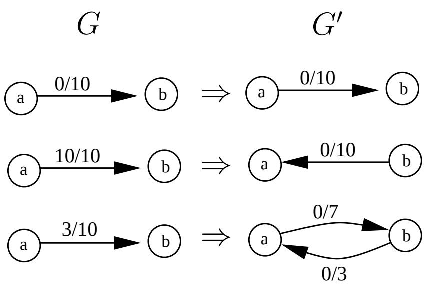
Das folgende Bild zeigt einen maximalen Fluss von s nach t in einem Beispielgraphen. Die Kantenlabels zeigen jeweils den Fluss über die Kante und die Kapazität der Kante. Der dargestellte maximale Fluss hat den Wert 40.



1.1 Das Residuennetzwerk

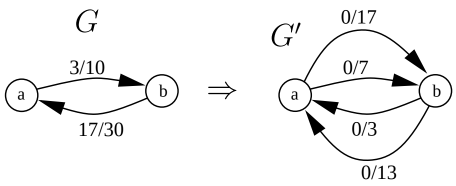
Zu einem (nicht notwendigerweise gültigen) Fluss $f : E \to \mathbb{R}$ in $G = (V, E)$ lässt sich das so genannte *Residuennetzwerk* $G' = (V, E)$ wie folgt konstruieren.

Erzeuge den Graphen G' aus G, indem alle Knoten von G kopiert werden und Kanten nach den folgenden Regeln in G' eingefügt werden.



Falls $f(e) < c(e)$ für Kante $e = (a, b) \in E$, füge die Vorwärtskante $e' = (a, b)$ mit Kapazität $c'(e') = c(e) - f(e)$ in E' ein, und falls $f(e) > 0$ für Kante $e = (a, b) \in E$, füge die Rückwärtskante $e' = (b, a)$ mit Kapazität $c'(e') = f(e)$ in E' ein. Für jede Kante aus G werden also eine oder zwei Kanten in G' eingefügt. Dies wird durch nebenstehendes Bild veranschaulicht.

Wenn in G zwei antiparallele Kanten zwischen zwei Knoten a und b vorhanden sind, können damit sogar vier Kanten zwischen a und b in G' eingefügt werden, wie in nebenstehendem Bild demonstriert wird.



Um das Residuennetzwerk G' zu erzeugen, müssen also lediglich alle Kanten $e \in E$ durchlaufen und für jede Kante abhängig von deren Fluss und Kapazität bis zu zwei Kanten in G' eingefügt werden.

Im Fall, dass $f(e) = 0$ für alle $e \in E$ gilt, ist das Residuennetzwerk G' eine exakte Kopie des Graphen G und enthält insbesondere nur Vorwärtskanten. Ändert sich der Fluss f an einer einzigen Kante e, so kann die entsprechende Änderung im Residuennetzwerk G' (bei Verwendung geeigneter Datenstrukturen) in konstanter Zeit durchgeführt werden.

Bemerkung: Der Algorithmus von Dinic berechnet den maximalen Fluss in einem Graphen durch die Suche nach kürzesten augmentierenden Pfaden, d.h. kürzesten Pfaden von s nach t im Residuennetzwerk. Die Laufzeit des Algorithmus von Dinic ist $O(|V|^2|E|)$. Wir betrachten einen anderen Algorithmus mit Laufzeit $O(|V|^3)$.

1.2 Der Fluss-Algorithmus von Goldberg und Tarjan

Zur Vereinfachung nehmen wir an, dass jeder Knoten $v \in V \setminus \{s, t\}$ von der Quelle aus erreichbar ist und dass von jedem solchen Knoten die Senke erreichbar ist. (Alle Knoten, die diese Bedingung nicht erfüllen, können nichts zu einem Fluss von s nach t beitragen; sie lassen sich in linearer Zeit bestimmen und aus dem Graphen entfernen.) Wir wollen einen *maximalen Fluss* in G von s nach t berechnen und betrachten dazu den so genannten Preflow-Push-Algorithmus von Goldberg und Tarjan. Dieser Algorithmus (bzw. die Implementierung, die wir betrachten) hat Laufzeit $O(|V|^3)$.

Der Algorithmus verwendet neben dem Konzept des Residuennetzwerks zwei neue Konzepte: *Präfluss* und *Entfernungsfunktion*. Ein Präfluss $\tilde{f}$ ist eine Abbildung $\tilde{f} : E \to \mathbb{R}$, die die folgenden Bedingungen erfüllt.

(a) $\forall e \in E : \ 0 \leq \tilde{f}(e) \leq c(e)$

(b) $\forall v \in V \setminus \{s, t\} : \ \sum_{e=(u,v)\in E} \tilde{f}(e) - \sum_{e=(v,w)\in E} \tilde{f}(e) \geq 0$

Im Unterschied zu einen gültigen Fluss muss für einem Präfluss also die Flusserhaltung nicht gelten: es darf mehr Fluss in einen Knoten hineinfließen als aus ihm heraus. Der

Überschuss (engl. *excess*) eines Knotens v ist definiert als

$$e(v) = \sum_{e=(u,v)\in E} \tilde{f}(e) - \sum_{e=(v,w)\in E} \tilde{f}(e)$$

und wird vom Algorithmus bei jedem Knoten gespeichert und bei Änderungen aktualisiert. Die Knoten $v \in V \setminus \{s,t\}$ mit $e(v) > 0$ heißen *aktive* Knoten und werden in einer Queue verwaltet. Der Algorithmus startet mit einem Präfluss und versucht dann immer, den Überschuss eines aktiven Knotens weiter in Richtung Senke zu drücken (engl. *push*), bis am Ende ein gültiger maximaler Fluss vorliegt.

Wie in Abschnitt 1.1 beschrieben gibt es zu einem Präfluss $\tilde{f}$ in G ein entsprechendes Residuennetzwerk G', wobei $c'(e')$ die Kapazität einer Kante e' im Residuennetzwerk bezeichnet, d.h. $c'(e') = c(e) - \tilde{f}(e)$ für Vorwärtskanten und $c'(e') = \tilde{f}(e)$ für Rückwärtskanten. Hier bezeichnet e die e' entsprechende Kante in G. Der Algorithmus von Goldberg und Tarjan aktualisiert das Residuennetzwerk G' bei jeder Änderung des Präflusses in G, so dass zu jedem Zeitpunkt das dem aktuellen Präfluss entsprechende Residuennetzwerk verfügbar ist. (Hinweis: Es ist auch möglich, bei der Implementierung auf den expliziten Aufbau des Residuennetzwerks G' zu verzichten und stattdessen den ganzen Algorithmus auf dem Graphen G mit zusätzlicher Verwaltungsinformation laufen zu lassen.)

Eine Funktion $d : V \to \mathbb{N}_0$ ist eine Entfernungsfunktion (engl. *distance*) im Graphen G bezüglich Präfluss $\tilde{f}$, wenn $d(t) = 0$ und für jede Kante $e = (v,w)$ im Residuennetzwerk (!) gilt: $d(v) \leq d(w) + 1$. Eine gültige Entfernungsfunktion gibt für jeden Knoten v eine untere Schranke für die Länge des kürzesten Pfades von v nach t im Residuennetzwerk an. Bezüglich einer Entfernungsfunktion heißt eine Kante $e = (v,w)$ im Residuennetzwerk *zulässig*, falls $d(v) = d(w) + 1$ gilt.

Der Algorithmus von Goldberg und Tarjan wählt wiederholt einen aktiven Knoten v aus und führt für v eine so genannte Push/Relabel-Operation aus: solange v Überschuss hat und es eine zulässige ausgehende Kante von v gibt, wird ein möglichst großer Teil des Überschusses von v über diese Kante gedrückt (Push); wenn es keine zulässige ausgehende Kante von v mehr gibt und v noch Überschuss hat, so wird das Distanzlabel $d(v)$ von v neu berechnet (Relabel). Der gesamte Algorithmus lässt sich damit wie folgt beschreiben:

1. **Präfluss-Initialisierung:**
   Setze $\tilde{f}(e) = c(e)$ für alle ausgehenden Kanten der Quelle s und $\tilde{f}(e) = 0$ für alle anderen Kanten. Füge die Endknoten v von ausgehenden Kanten (s,v) aus s mit $v \neq t$ in die Queue der aktiven Knoten ein.

2. **Initialisierung der Entfernungsfunktion:**
   Initialisiere für jeden Knoten $v \neq s$ die Entfernungsfunktion $d(v)$ mit der Länge (Anzahl Kanten) eines kürzesten gerichteten Pfades von v nach t im Residuennetzwerk. Setze dabei $d(t) = 0$. Diese Berechnung kann mit einer rückwärtsgerichteten Breitensuche (Startknoten t, verwende Kanten im Residuennetzwerk in der entgegengesetzten Richtung) in Zeit $O(|V| + |E|)$ implementiert werden. Die BFS erreicht die Quelle s nicht, initialisiere daher $d(s) = |V|$.

3. **Hauptschleife:**
   Solange die Queue mit aktiven Knoten nicht leer ist, hole den ersten solchen Knoten v aus der Queue und führe für v die Push/Relabel-Operation aus.

Die Push/Relabel-Operation für einen Knoten v läuft dabei wie folgt ab:

1. Solange $e(v) > 0$ und es im Residuennetzwerk G' eine bezüglich der aktuellen Entfernungsfunktion zulässige ausgehende Kante von v, d.h. eine Kante $e' = (v, w)$ mit $d(v) = d(w) + 1$, gibt, führe einen Push aus:

   Push: Drücke $\delta = \min\{e(v), c'(e')\}$ Flusseinheiten von v nach w. Genauer: falls e' eine zu der Kante e in G gehörende Vorwärtskante ist, so erhöhe den Präfluss $\tilde{f}(e)$ auf e um δ, und falls e' eine zu e gehörende Rückwärtskante ist, so erniedrige den Präfluss $\tilde{f}(e)$ auf e um δ. Dadurch erniedrigt sich $e(v)$ um δ und erhöht sich $e(w)$ um δ. Falls w noch nicht aktiv war und $w \neq t$ und $w \neq s$, so wird w jetzt aktiv und wird hinten an die Queue aktiver Knoten angehängt.

2. Falls v noch Überschuss hat, d.h. $e(v) > 0$, und es im Residuennetzwerk keine zulässige ausgehende Kante von v gibt, so führe ein Relabel wie folgt aus und hänge v anschließend hinten an die Queue aktiver Knoten an.

   Relabel: Erhöhe das Distanz-Label $d(v)$ von v auf den Wert

   $$\min\{d(w) + 1 \mid (v, w) \text{ ist Kante im Residuennetzwerk } G'\}.$$

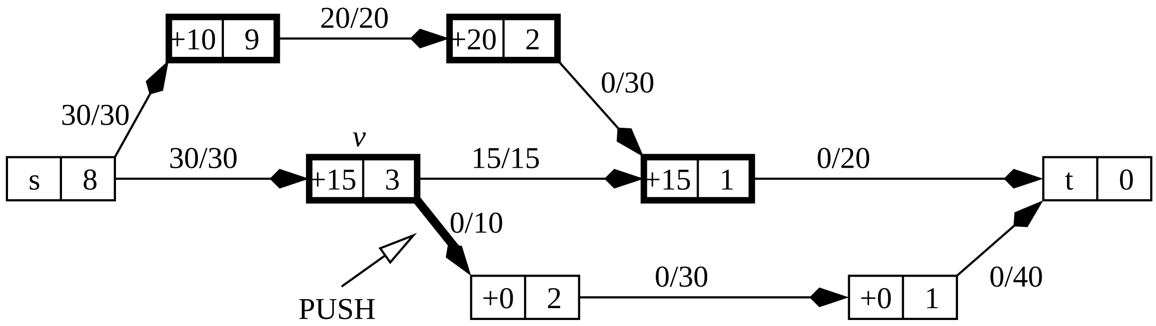
Bei der Implementierung ist zu beachten, dass der Algorithmus bei jeder Push-Operation gleichzeitig auch das Residuennetzwerk aktualisieren muss.

Der Algorithmus terminiert, wenn die Queue aktiver Knoten leer wird. Dann ist der Präfluss ein gültiger maximaler Fluss, weil es im Residuennetzwerk keinen augmentierenden Pfad von s nach t gibt (wegen $d(s) = |V|$). Der Beweis, dass die Laufzeit des Algorithmus $O(|V|^3)$ ist, ist dagegen etwas komplizierter und soll hier nicht ausgeführt werden. Es soll nur erwähnt werden, dass sowohl von praktischen als auch theoretischen Gesichtspunkten gegenwärtig modifizierte Varianten (z.B. bzgl. Auswahl von aktiven Knoten) des Preflow-Push-Algorithmus von Goldberg und Tarjan die besten bekannten Fluss-Algorithmen sind.

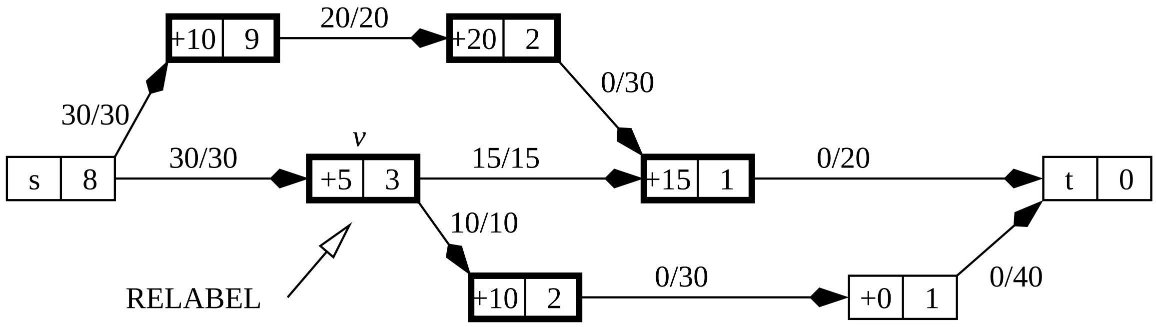
Literatur: Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin: *Network Flows.* Prentice Hall, 1993, S. 207 ff.

Beispiel einer Push/Relabel-Operation

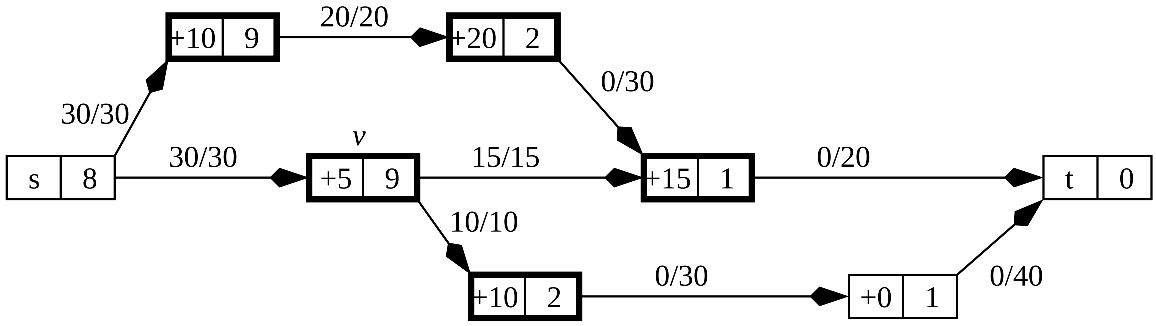
Hier soll eine Push/Relabel-Operationen anhand eines Beispiels verdeutlicht werden. Wir betrachten den folgenden Graphen mit dem eingezeichneten Präfluss. Aktive Knoten sind fett umrandet. In den Knoten wird jeweils links der Überschuss (außer bei Quelle s und Senke t), rechts der Distanzwert angegeben. Die Kantenlabels haben die Form $\tilde{f}(e)/c(e)$.



Es werde eine Push/Relabel-Operation für den Anfangsknoten v der fett eingezeichneten Kante e ausgeführt. Die Vorwärtskante e', die der Kante e im Residuennetzwerk entspricht, ist zulässig, weil sich die Distanzwerte der Endknoten um genau Eins unterscheiden. Deshalb werden über e jetzt $\min\{c'(e'), e(v)\} = \min\{10, 15\} = 10$ Flusseinheiten gedrückt. Nach dem Push ergibt sich das folgende Bild.



Da der Knoten v nun im Residuennetzwerk keine zulässige ausgehende Kante mehr hat, wird seine Entfernungsfunktion neu berechnet (Relabel). Die einzige ausgehende Kante von v im Residuennetzwerk führt zu s mit $d(s) = 8$, daher wird der Distanzwert von v auf 9 erhöht.



Dann wird v hinten an die Queue aktiver Knoten angehängt. Wenn später noch einmal eine Push/Relabel-Operation für v ausgeführt wird, so kann sein Überschuss von 5 Flusseinheiten zur Quelle s zurückgedrückt werden.