Maximaler Fluss in Flussnetzwerken

Maximilian Moeller

04.09.2020

Proseminar Theoretische Informatik 2020

Gliederung

1. Maximaler-Fluss-Problem
   - Flussnetzwerke
   - Fluss
2. Ford-Fulkerson-Algorithmen
   - Restnetzwerke
   - Erweiterungspfade
   - generischer Algorithmus
3. Push/Relabel-Algorithmen
   - Grundlagen
   - Operationen
   - generischer Algorithmus

alle Codesnippets und Definitionen aus [Cor+09]

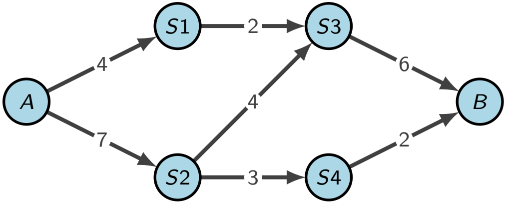
Motivation

Motivation

- (Stoff-) Mengen auf mehreren Pfaden gleichzeitig transportiert
- Pfade durch Kapazitäten beschränkt

Motivation

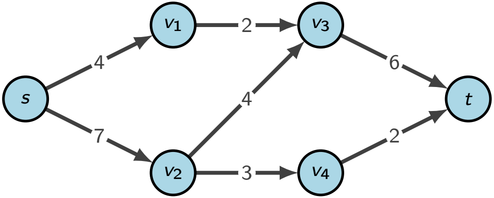
- (Stoff-) Mengen auf mehreren Pfaden gleichzeitig transportiert
- Pfade durch Kapazitäten beschränkt
- Beispiel Rechnernetze: Maximaler Durchsatz von A nach B?



Maximaler-Fluss-Problem

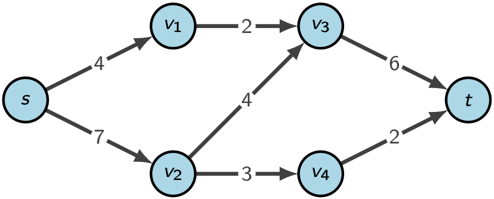
Flussnetzwerke

- gerichteter Graph
  $G = (V, E)$



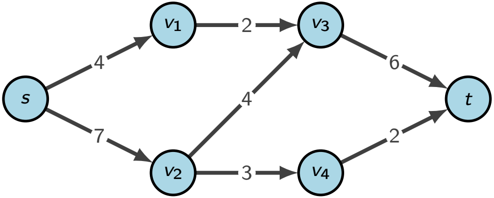
Flussnetzwerke

- gerichteter Graph $G = (V, E)$
- Kapazitäten $c(u, v) \geq 0$ für $u, v \in V$ (meist $\mathbb{N}$)



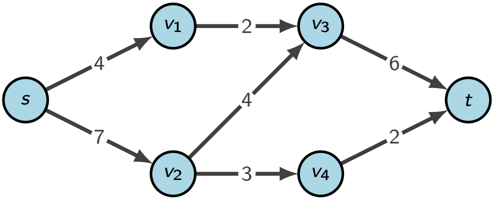
Flussnetzwerke

- gerichteter Graph $G = (V, E)$
- Kapazitäten $c(u, v) \geq 0$ für $u, v \in V$ (meist $\mathbb{N}$)
- Quelle s und Senke t



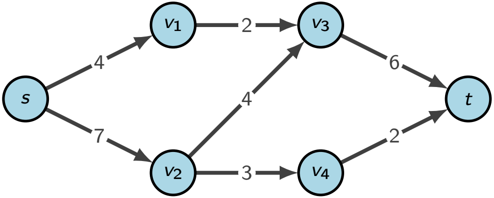
Flussnetzwerke

- gerichteter Graph $G = (V, E)$
- Kapazitäten $c(u, v) \geq 0$ für $u, v \in V$ (meist $\mathbb{N}$)
- Quelle s und Senke t
- jeder Knoten v liegt auf einem Pfad von s nach t



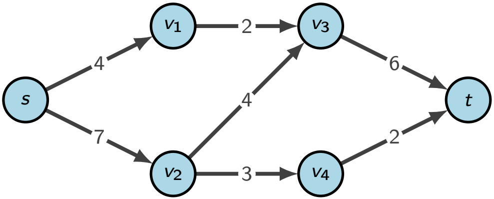
Flussnetzwerke

- gerichteter Graph $G = (V, E)$
- Kapazitäten $c(u, v) \geq 0$ für $u, v \in V$ (meist $\mathbb{N}$)
- Quelle s und Senke t
- jeder Knoten v liegt auf einem Pfad von s nach t
- Keine reflexiven Kanten (gleicher Start- und Zielknoten)



Flussnetzwerke

- gerichteter Graph $G = (V, E)$
- Kapazitäten $c(u, v) \geq 0$ für $u, v \in V$ (meist $\mathbb{N}$)
- Quelle s und Senke t
- jeder Knoten v liegt auf einem Pfad von s nach t
- Keine reflexiven Kanten (gleicher Start- und Zielknoten)
- Keine entgegen gerichteten Kanten



Fluss in Flussnetzwerken

Fluss $f : V \times V \to \mathbb{N}$ erfüllt drei Bedingungen:

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

Fluss in Flussnetzwerken

Fluss $f : V \times V \to \mathbb{N}$ erfüllt drei Bedingungen:

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

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

Fluss in Flussnetzwerken

Fluss $f : V \times V \to \mathbb{N}$ erfüllt drei Bedingungen:

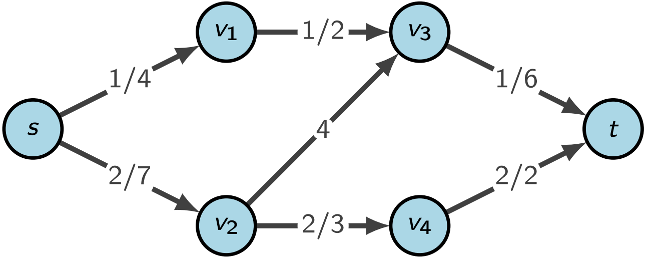
$$\forall u, v \in V\colon (u, v) \notin E \Rightarrow f(u, v) = 0 \tag{1}$$

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

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

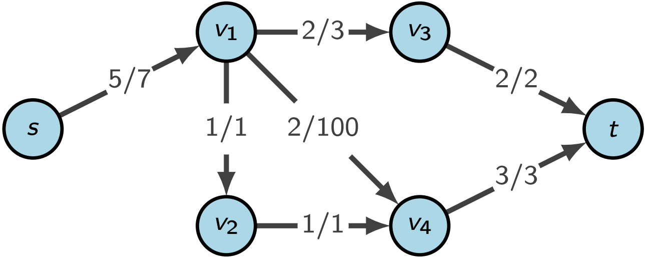
Notation/Beispiele

$f(u, v)/c(u, v)$



Notation/Beispiele

$f(u,v)/c(u,v)$



Wert eines Flusses

Wert eines Flusses f:

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

Wert eines Flusses

Wert eines Flusses f:

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

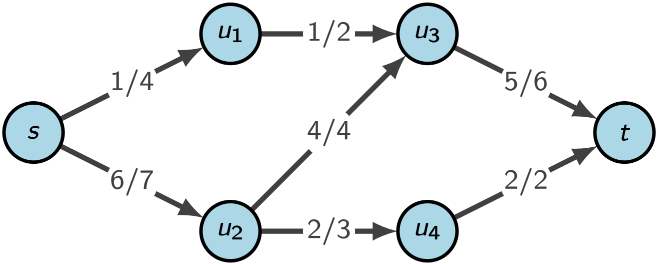


Abbildung: Ein Fluss f mit $|f| = 7$

Maximaler-Fluss-Problem

- Gegeben: ein Flussnetzwerk $G = (V, E)$ und dessen Kapazitätsfunktion c.

Maximaler-Fluss-Problem

- Gegeben: ein Flussnetzwerk $G = (V, E)$ und dessen Kapazitätsfunktion c.
- Gesucht: ein Fluss f, dessen Wert $|f|$ maximal für dieses Flussnetzwerk ist.

Ford-Fulkerson-Algorithmen

Restnetzwerke

Restkapazität:

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{falls } (u, v) \in E \\ f(v, u) & \text{falls } (v, u) \in E \\ 0 & \text{sonst} \end{cases}$$

Restnetzwerke

Restkapazität:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{falls } (u,v) \in E \\ f(v,u) & \text{falls } (v,u) \in E \\ 0 & \text{sonst} \end{cases}$$

Restkanten:

$$E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$$

Restnetzwerke

Restkapazität:

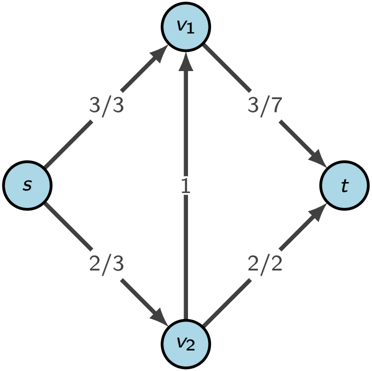
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{falls } (u,v) \in E \\ f(v,u) & \text{falls } (v,u) \in E \\ 0 & \text{sonst} \end{cases}$$

Restkanten:

$$E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$$

Ein Fluss f in einem Flussnetzwerk $G = (V, E)$ induziert das Restnetzwerk $G_f = (V, E_f)$.

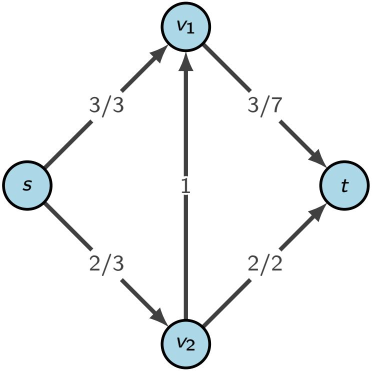
Restnetzwerke



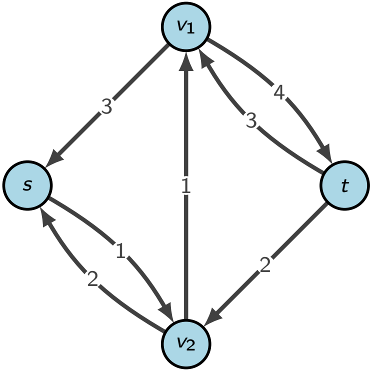
(a) Fluss f in einem Flussnetzwerk
$G = (V, E)$

Definition eines Flusses f' im Restnetzwerk analog zu Fluss in Flussnetzwerken.

Restnetzwerke



(a) Fluss f in einem Flussnetzwerk $G = (V, E)$



(b) durch f induziertes Restnetzwerk $G_f = (V, E_f)$

Definition eines Flusses f' im Restnetzwerk analog zu Fluss in Flussnetzwerken.

Erhöhung eines Flusses

Gegeben:

- Fluss f im Flussnetzwerk $G = (V, E)$
- Fluss f' im Restnetzwerk $G_f = (V, E_f)$

Erhöhung eines Flusses

Gegeben:

- Fluss f im Flussnetzwerk $G = (V, E)$
- Fluss f' im Restnetzwerk $G_f = (V, E_f)$

Erhöhung von f um f':

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{falls } (u, v) \in E \\ 0 & \text{sonst} \end{cases}$$

Erhöhung eines Flusses

Gegeben:

- Fluss f im Flussnetzwerk $G = (V, E)$
- Fluss f' im Restnetzwerk $G_f = (V, E_f)$

Erhöhung von f um f':

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{falls } (u, v) \in E \\ 0 & \text{sonst} \end{cases}$$

Wert der Erhöhung:

$$|f \uparrow f'| = |f| + |f'|$$

Erhöhung eines Flusses

Gegeben:

- Fluss f im Flussnetzwerk $G = (V, E)$
- Fluss f' im Restnetzwerk $G_f = (V, E_f)$

Erhöhung von f um f':

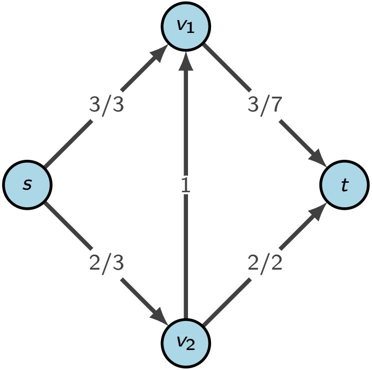
$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & \text{falls } (u, v) \in E \\ 0 & \text{sonst} \end{cases}$$

Wert der Erhöhung:

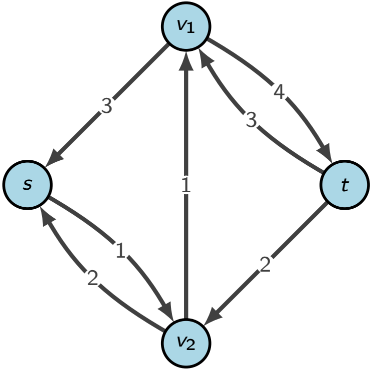
$$|f \uparrow f'| = |f| + |f'|$$

Beweis: über Umformen der Summen. $\square$

Erhöhung eines Flusses

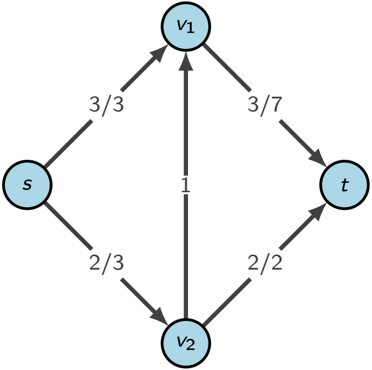


(a) f in G

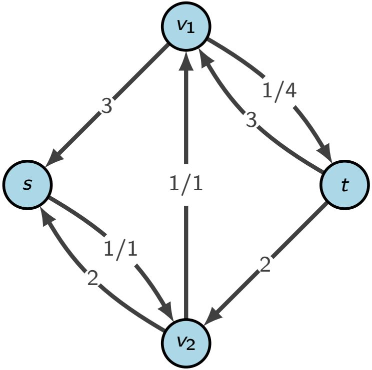


(b) G_f

Erhöhung eines Flusses

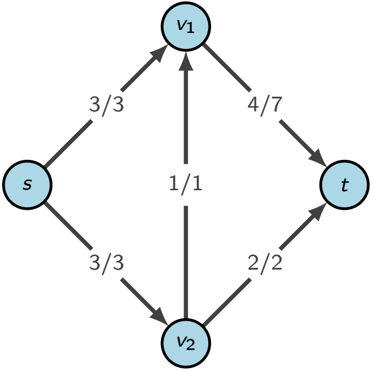


(a) f in G

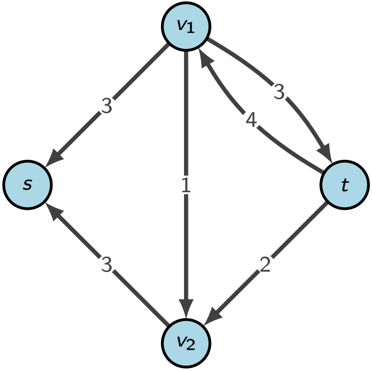


(b) f' in G_f

Erhöhung eines Flusses



(a) $f \uparrow f'$ in G



(b) $G_{f \uparrow f'}$

Erweiterungspfade

Gegeben:

- Pfad p von s nach t im Restnetzwerk G_f

Erweiterungspfade

Gegeben:

- Pfad p von s nach t im Restnetzwerk G_f

Pfadkapazität:

$$c_f(p) = min\{c_f(u,v) \mid (u,v) \text{ liegt auf } p\}$$

Erweiterungspfade

Gegeben:

- Pfad p von s nach t im Restnetzwerk G_f

Pfadkapazität:

$$c_f(p) = min\{c_f(u,v) \mid (u,v) \text{ liegt auf } p\}$$

Definiere Fluss f_p entlang von p:

$$f_p(u,v) = \begin{cases} c_f(p) & \text{falls } (u,v) \text{ auf } p \text{ liegt} \\ 0 & \text{sonst} \end{cases}$$

Erweiterungspfade

Gegeben:

- Pfad p von s nach t im Restnetzwerk G_f

Pfadkapazität:

$$c_f(p) = min\{c_f(u,v) \mid (u,v) \text{ liegt auf } p\}$$

Definiere Fluss f_p entlang von p:

$$f_p(u,v) = \begin{cases} c_f(p) & \text{falls } (u,v) \text{ auf } p \text{ liegt} \\ 0 & \text{sonst} \end{cases}$$

Damit:

$$|f \uparrow f_p| = |f| + |f_p| > |f|$$

Ford-Fulkerson-Basisalgorithmus

Algorithm 1 Ford-Fulkerson(G,s,t)

for jede Kante $(u, v) \in G.E$
    $(u, v).f = 0$
while $\exists$ Pfad p von s nach t in G_f
    $c_f(p) = \min\{c_f(u, v) \mid (u, v)$ liegt auf $p\}$
    for jede Kante (u, v) von p
        if $(u, v) \in G.E$
            $(u, v).f = (u, v).f + c_f(p)$
        else
            $(v, u).f = (v, u).f - c_f(p)$

Ford-Fulkerson-Analyse

Eigenschaften des generischen Ford-Fulkerson-Algorithmus:

- Terminiert stets bei Kantengewichten aus $\mathbb{N}$ (und $\mathbb{Q}$)

Ford-Fulkerson-Analyse

Eigenschaften des generischen Ford-Fulkerson-Algorithmus:

- Terminiert stets bei Kantengewichten aus $\mathbb{N}$ (und $\mathbb{Q}$)
- Fluss ist maximal wegen maxflow-mincut-Theorem
  - u.A.: $|f|$ ist maximal $\Leftrightarrow$ kein Erweiterungspfad in G_f

Ford-Fulkerson-Analyse

Eigenschaften des generischen Ford-Fulkerson-Algorithmus:

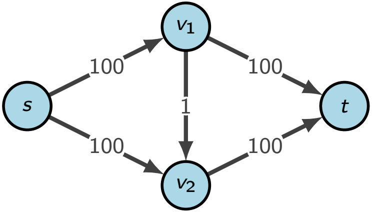
- Terminiert stets bei Kantengewichten aus $\mathbb{N}$ (und $\mathbb{Q}$)
- Fluss ist maximal wegen maxflow-mincut-Theorem
  - u.A.: $|f|$ ist maximal $\Leftrightarrow$ kein Erweiterungspfad in G_f
- Laufzeit: $\mathcal{O}(|f_{max}| \cdot (E + V))$ für einen maximalen Fluss f_{max}

Ford-Fulkerson-Analyse

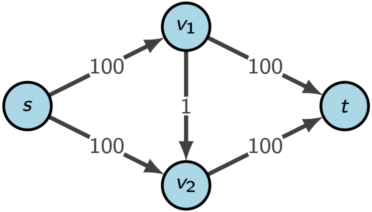
Eigenschaften des generischen Ford-Fulkerson-Algorithmus:

- Terminiert stets bei Kantengewichten aus $\mathbb{N}$ (und $\mathbb{Q}$)
- Fluss ist maximal wegen maxflow-mincut-Theorem
  - u.A.: $|f|$ ist maximal $\Leftrightarrow$ kein Erweiterungspfad in G_f
- Laufzeit: $\mathcal{O}(|f_{max}| \cdot (E + V))$ für einen maximalen Fluss f_{max}
- Problem: schlechte Wegewahl möglich

schlechte Wegewahl



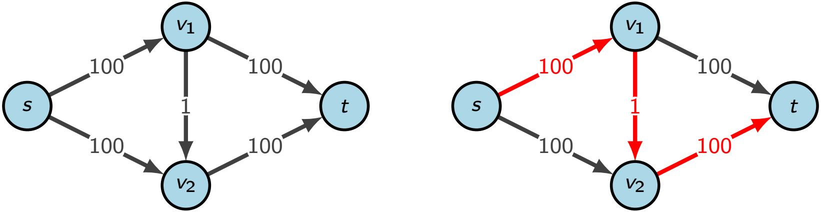
(a) G mit f



(b) G_f

Abbildung: gewählter Erweiterungspfad in rot

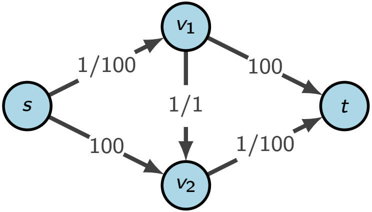
schlechte Wegewahl



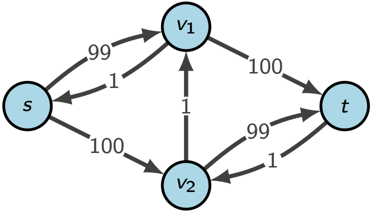
(a) G mit f (b) G_f

Abbildung: gewählter Erweiterungspfad in rot

schlechte Wegewahl



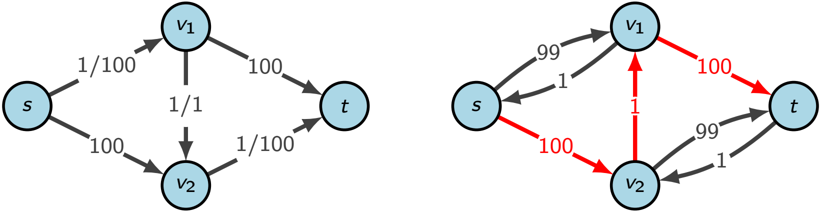
(a) G mit f



(b) G_f

Abbildung: gewählter Erweiterungspfad in rot

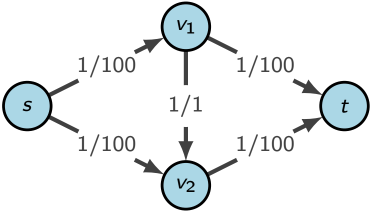
schlechte Wegewahl



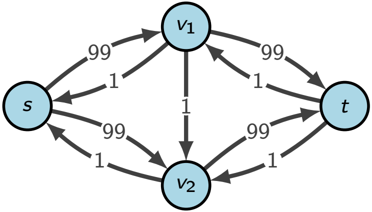
(a) G mit f (b) G_f

Abbildung: gewählter Erweiterungspfad in rot

schlechte Wegewahl



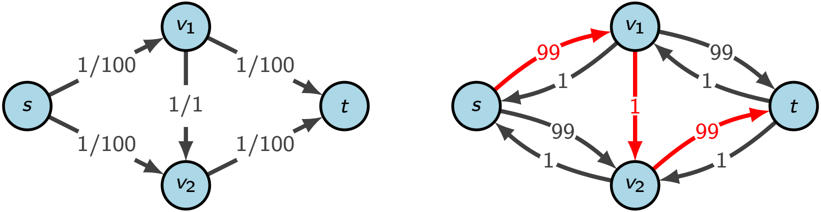
(a) G mit f



(b) G_f

Abbildung: gewählter Erweiterungspfad in rot

schlechte Wegewahl



(a) G mit f (b) G_f

Abbildung: gewählter Erweiterungspfad in rot

Edmonds-Karp-Algorithmus

Wahl des Erweiterungspfades als ein kürzester Pfad (minimale Pfadlänge)

- $\mathcal{O}(V \cdot E^2)$

Push/Relabel-Algorithmen

Push/Relabel-Grundlagen

arbeitet mit Vorfluss:

$$\forall u \in V \setminus \{s\} : \sum_{v \in V} f(v, u) \geq \sum_{v \in V} f(u, v)$$

Push/Relabel-Grundlagen

arbeitet mit Vorfluss:

$$\forall u \in V \setminus \{s\} : \sum_{v \in V} f(v,u) \geq \sum_{v \in V} f(u,v)$$

Flussüberschuss $e(u)$ des Knotens $u \in V$:

$$e(u) = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v)$$

Push/Relabel-Grundlagen

arbeitet mit Vorfluss:

$$\forall u \in V \setminus \{s\} : \sum_{v \in V} f(v, u) \geq \sum_{v \in V} f(u, v)$$

Flussüberschuss $e(u)$ des Knotens $u \in V$:

$$e(u) = \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v)$$

Höhenfunktion $h : V \to \mathbb{N}$:

$$\forall (u, v) \in E_f : h(u) \leq h(v) + 1$$

Push/Relabel-Grundlagen

arbeitet mit Vorfluss:

$$\forall u \in V \setminus \{s\} : \sum_{v \in V} f(v,u) \geq \sum_{v \in V} f(u,v)$$

Flussüberschuss $e(u)$ des Knotens $u \in V$:

$$e(u) = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v)$$

Höhenfunktion $h : V \to \mathbb{N}$:

$$\forall (u,v) \in E_f : h(u) \leq h(v) + 1$$

$h(S) = |V|$, $h(t) = 0$

Push-Operation

Algorithm 2 Push(u,v)

```
// **Anwendbar wenn:** u.e > 0, c_f(u,v) > 0, und u.h = v.h+1.
// **Aktion:** Drücke Δ_f(u,v) = min(u.e, c_f(u,v)) Flusseinheiten von u nach v.

Δ_f(u,v) = min(u.e, c_f(u,v))
if (u,v) ∈ E
    (u,v).f = (u,v).f + Δ_f(u,v)
else
    (v,u).f = (v,u).f − Δ_f(u,v)
u.e = u.e − Δ_f(u,v)
v.e = v.e + Δ_f(u,v)
```

Relabel-Operation

Algorithm 3 Relabel(u)

```
// Anwendbar wenn: u.e > 0, und u.h ≤ v.h für alle v ∈ V mit (u, v) ∈ E_f.
// Aktion: setze u.h auf einen höheren Wert.

u.h = 1+min{v.h | (u, v) ∈ E_f}
```

Initialisierung

Algorithm 4 Initialize-Preflow(G,s)

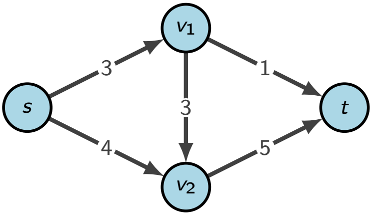
```
for jeden Knoten v ∈ G.V
    v.h = 0
    v.e = 0
for jede Kante (u, v) ∈ G.E
    (u, v).f = 0
s.h = |G.V|
for jeden Knoten v mit (s, v) ∈ G.E
    (s, v).f = c(s, v)
    v.e = c(s, v)
    s.e = s.e − c(s, v)
```

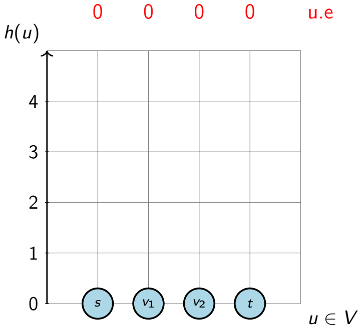
generischer Push/Relabel-Algorithmus

Algorithm 5 Generic-Push/Relabel

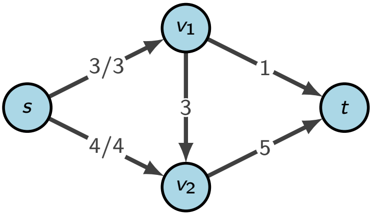
1: Initialize-Preflow(G,s)
2: **while** es existiert eine ausführbare Push- oder Relabel-Operation
3: wähle eine ausführbare Push- oder Relabel-Operation und führe sie aus

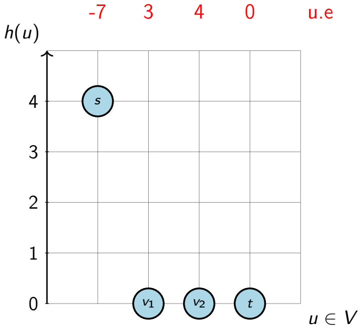
Push/Relabel-Beispiel



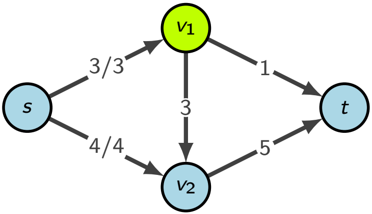


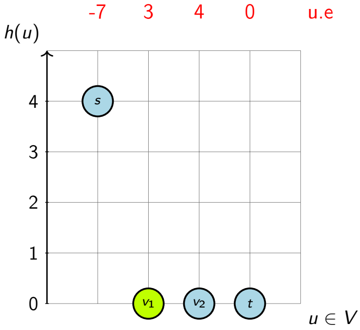
Push/Relabel-Beispiel



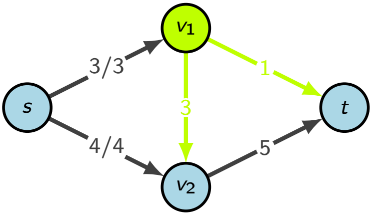


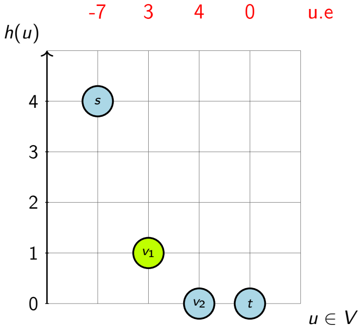
Push/Relabel-Beispiel



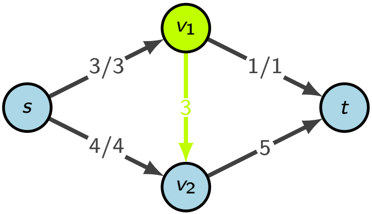


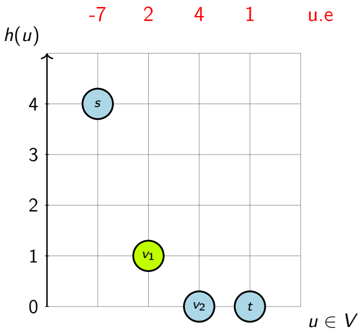
Push/Relabel-Beispiel



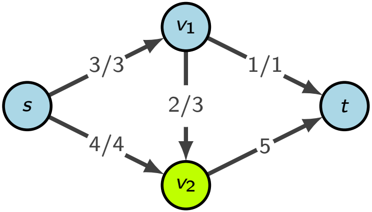


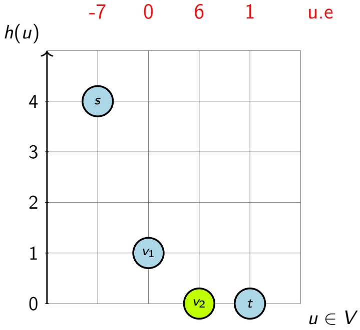
Push/Relabel-Beispiel



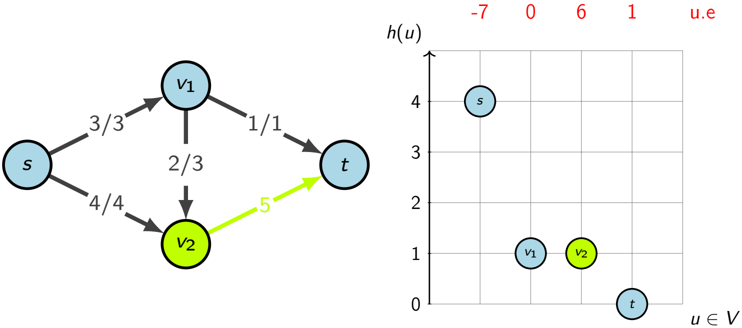


Push/Relabel-Beispiel

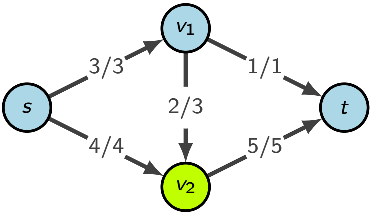


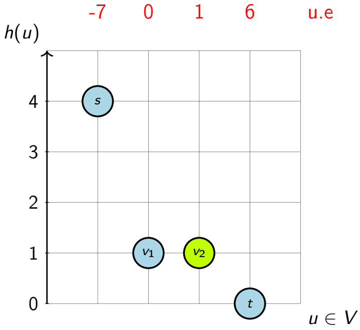


Push/Relabel-Beispiel

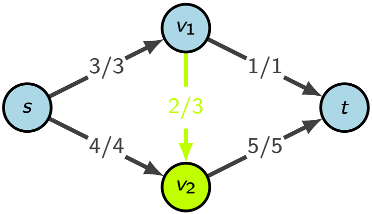


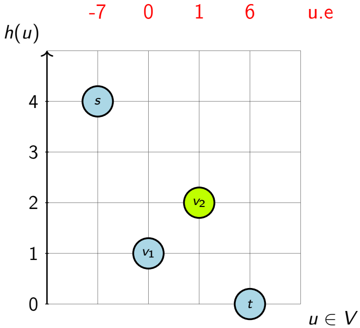
Push/Relabel-Beispiel



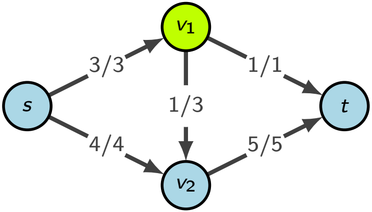


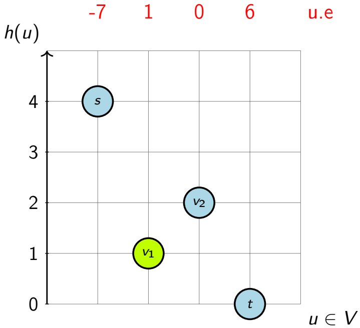
Push/Relabel-Beispiel



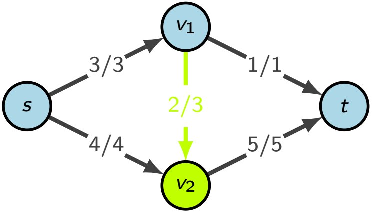


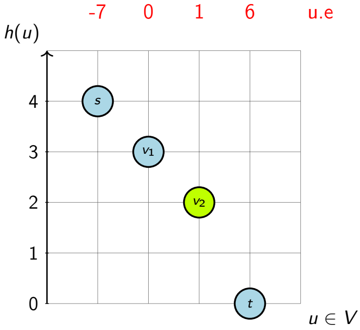
Push/Relabel-Beispiel



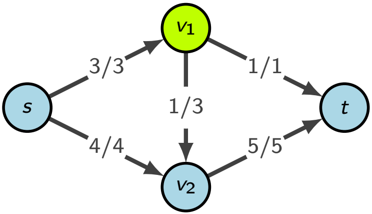


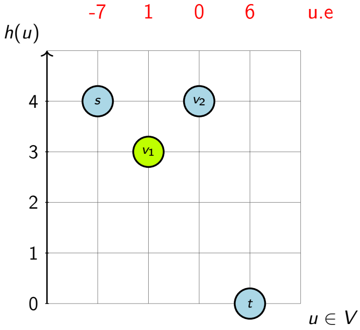
Push/Relabel-Beispiel



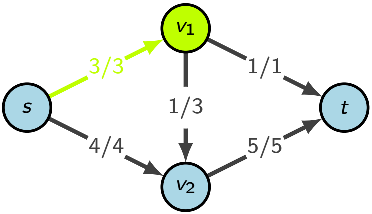


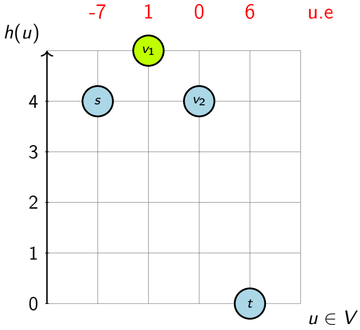
Push/Relabel-Beispiel



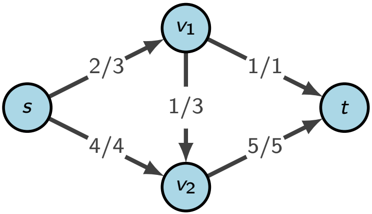


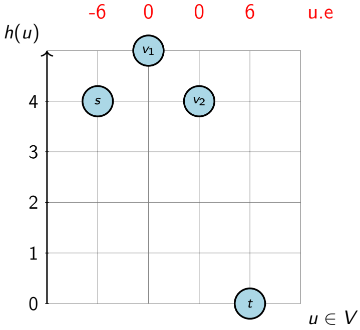
Push/Relabel-Beispiel





Push/Relabel-Beispiel





Push/Relabel-Analyse

Eigenschaften des generischen Push/Relabel-Algorithmus:

- Vorfluss f ist bei Terminierung ein Fluss.

Push/Relabel-Analyse

Eigenschaften des generischen Push/Relabel-Algorithmus:

- Vorfluss f ist bei Terminierung ein Fluss.
- insgesamt $\mathcal{O}(V^2 \cdot E)$

Push/Relabel-Analyse

Eigenschaften des generischen Push/Relabel-Algorithmus:

- Vorfluss f ist bei Terminierung ein Fluss.
- insgesamt $\mathcal{O}(V^2 \cdot E)$
- Problem: beliebige Reihenfolge ineffizient

Push/Relabel-Analyse

Eigenschaften des generischen Push/Relabel-Algorithmus:

- Vorfluss f ist bei Terminierung ein Fluss.
- insgesamt $\mathcal{O}(V^2 \cdot E)$
- Problem: beliebige Reihenfolge ineffizient
- Verbesserung: Relabel-to-Front-Algorithmus
  - $\mathcal{O}(V^3)$

Zusammenfassung

Anwendungsfälle

- Kalter Krieg: Zerstörung von Schienennetzwerken mit geringstem Aufwand (minimum cut)

Anwendungsfälle

- Kalter Krieg: Zerstörung von Schienennetzwerken mit geringstem Aufwand (minimum cut)
- maximale bipartite Matchings (z.B. Zuteilung von Arbeitern und Jobs)

Anwendungsfälle

- Kalter Krieg: Zerstörung von Schienennetzwerken mit geringstem Aufwand (minimum cut)
- maximale bipartite Matchings (z.B. Zuteilung von Arbeitern und Jobs)
- automatisierte Erkennung von Vordergrund und Hintergrund in Bildbearbeitung

Zusammenfassung

Ford-Fulkerson- und Push/Relabel-Algorithmen lösen das maximaler-Fluss-Problem.

Zusammenfassung

Ford-Fulkerson- und Push/Relabel-Algorithmen lösen das maximaler-Fluss-Problem.

Heutzutage sind schnellere Push/Relabel-Algorithmen bekannt.

Zusammenfassung

Ford-Fulkerson- und Push/Relabel-Algorithmen lösen das maximaler-Fluss-Problem.

Heutzutage sind schnellere Push/Relabel-Algorithmen bekannt.

Zahlreiche Anwendungen

Literatur

Thomas H. Cormen u. a. *Introduction to Algorithms*. 3. Aufl. The MIT Press, 2009. ISBN: 978-3-486-74861-1.

Adrian Haarbach. *Goldberg Tarjan Push Relabel Algorithm*. [Online; accessed 20-July-2020]. 2020.