Kapitel 4

Flüsse in Graphen, Teil 2

Effiziente Algorithmen, SS 2018

Professor Dr. Petra Mutzel

VO 7 am 3. Mai 2018

Zusammenfassung Algo. von Ford und Fulkerson

- Korrektheit gezeigt im Theorem „Max Flow = Min Cut"
- Algorithmus von Ford-Fulkerson (FF) ist nur pseudopolynomiell (für ganzzahlige Kapazitäten):
  Laufzeit $O(B \cdot (|V| + |E|))$,
  mit B: obere Schranke für max. ϕ, z.B. $B = \sum_{e=(Q,\cdot)} c(e)$
- Rationale Kapazitäten können zu ganzzahligen Kapazitäten skaliert werden $\rightarrow$ FF klappt!
- Für irrationale Kapazitäten terminiert FF nicht!
- Lösung: Sucht man kürzeste FVWs (z.B. Breitensuche), dann sind alle obigen Probleme gelöst und die Laufzeit ist $O(ne^2) = O(n^5)$ [unabhängig gezeigt von Dinic 1970 und Edmonds, Karp 1972]
- Seitdem: Wettrennen um den schnellsten Max Flow Algorithmus

Alternative Flussalgorithmen

$$n = |V|, e = |E|, U = \max\{c(e) \mid e \in E\}$$

Jahr	Autoren	Zeit in n, e, U	Zeit bei $e = \Omega(n^2)$
1969	Edmonds/Karp	$O(ne^2)$	$O(n^5)$
1970	Dinic	$O(n^2 e)$	$O(n^4)$
1974	Karzanov	$O(n^3)$	$O(n^3)$
1977	Cherkassky	$O(n^2 e^{1/2})$	$O(n^3)$
1978	Malhotra/Pramodh Kumar/Maheshwari	$O(n^3)$	$O(n^3)$
1978	Galil	$O(n^{5/3} e^{2/3})$	$O(n^3)$
1978	Galil/Naamad, Shiloach	$O(ne \log^2 n)$	$O(n^3 \log^2 n)$
1980	Sleator/Tarjan	$O(ne \log n)$	$O(n^3 \log n)$
1982	Shiloach/Vishkin	$O(n^3)$	$O(n^3)$
1983	Gabow	$O(ne \log U)$	$O(n^3 \log U)$
1984	Tarjan	$O(n^3)$	$O(n^3)$
1985	Goldberg	$O(n^3)$	$O(n^3)$
1986	Goldberg/Tarjan	$O(ne \log(n^2/e))$	$O(n^3)$
1986	Ahuja/Orlin	$O(ne + n^2 \log U)$	$O(n^3 + n^2 \log U)$
...	...	...	...
1994	King, Rao, Tarjan	$O(ne \log_{e/(n \log n)} n)$	$O(n^3 \log_{n/\log n} n)$
1998	Goldberg, Rao	$O(\min(n^{2/3}, e^{1/2}) e \log(n^2/e) \log U)$	$O(n^{8/3} \log U)$

Zwei Ideen zur Beschleunigung

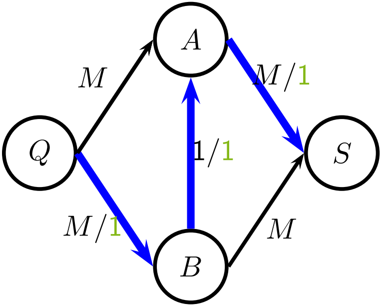
ab jetzt Netzwerk $G = (V, E, c)$ mit $|V| = n$ und $|E| = e$

Beobachtung 1: Ford-Fulkerson braucht
Zeit $O(n + e)$ für einen FVW

Frage: Kann man (in der Zeit) vielleicht
mehrere disjunkte FVW berechnen?

Beobachtung 2: Kürzere FVW
wären besser gewesen.

Frage: Nützt es etwas, nur
kürzeste FVW
zu benutzen?



Niveaunetzwerke

Intuition: Entferne Knoten/Kanten aus Rest_Φ, welche nicht auf kürzestem Q-S-Weg liegen können.

Definition 4.9

Betrachte $(G = (V, E), c)$ mit Fluss Φ, Rest_Φ, Distanzen $d(v) =$ Länge (#Kanten) des kürzesten Q-v-Wegs mit $v \in V$.

Für $0 \leq i \leq d(S)$ definiere i-tes Niveau

$$V_i := \begin{cases} \{v \in V \mid d(v) = i\} & \text{für } i < d(S), \\ \{S\} & \text{für } i = d(S). \end{cases}$$

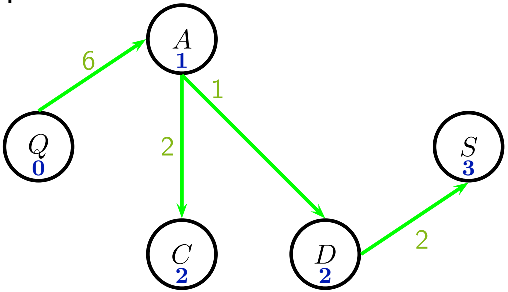
Definiere Niveaunetzwerk $N_\Phi = (V_\Phi, E_\Phi, r_\Phi)$ durch $V_\Phi := \bigcup_{i=0}^{d(S)} V_i$,

$E_i := \{(v, w) \in V_{i-1} \times V_i \mid (v, w) \in \text{Rest}_\Phi\}$, $E_\Phi := \bigcup_{i=1}^{d(S)} E_i$, und

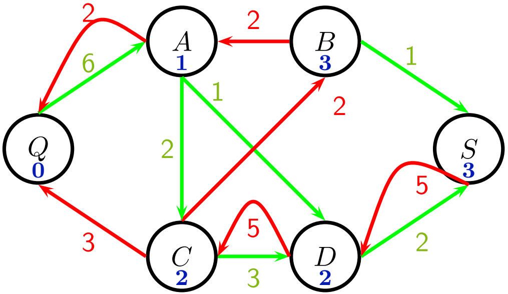
r_Φ wie in Rest_Φ.

klar Niveaunetzwerk berechenbar in Zeit $O(n + e)$

Beispiel Niveaunetzwerk

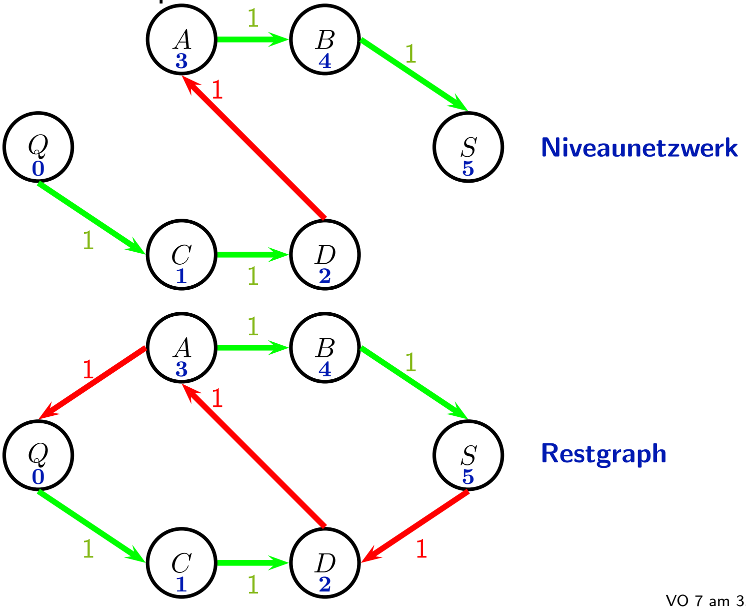


Niveaunetzwerk



Restgraph

Noch ein Beispiel Niveaunetzwerk



Sperrflüsse

Wir wollen „möglichst viel Fluss“ auf einmal addieren ...

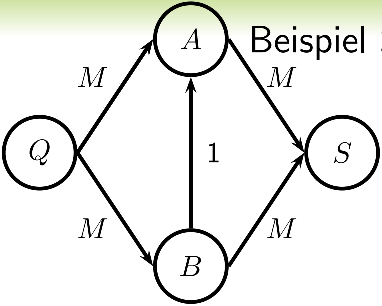
Definition 4.10

Sei ψ ein Fluss **im Niveaunetzwerk** zum Fluss ϕ in G.
Eine Kante e heißt saturiert durch ψ, wenn $\psi(e) = r_\phi(e)$ gilt.
Der Fluss ψ heißt Sperrfluss, wenn auf jedem Q-S-Weg in N_Φ mindestens eine saturierte Kante liegt.

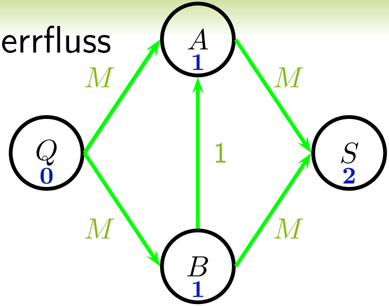
Analog zu FVW sieht man ein:

- Ein Sperrfluss ψ kann zum Fluss ϕ „addiert“ werden.
- Dabei wird zu Vorwärtskanten addiert und von Rückwärtskanten subtrahiert.
- Das Resultat ist ein Fluss ϕ' mit $w(\phi') > w(\phi)$.

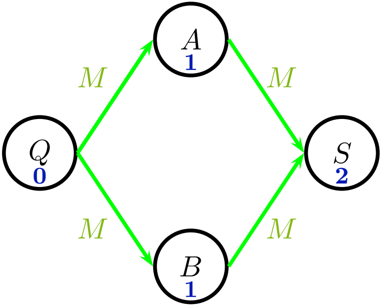
Beispiel Sperrfluss



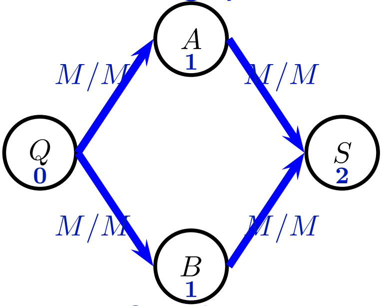
Netzwerk mit Fluss 0



Restgraph



Niveaunetzwerk mit Fluss 0



Sperrfluss

Idee der Sperrflussberechnung: Iterierte Tiefensuche

- Beginne mit dem leeren Sperrfluss Ψ
- Wiederhole solange, bis Q in N_Φ keine ausgehende Kante mehr besitzt:
  - DFS(Q) im Niveaunetzwerk N_Φ, wobei Knoten, die als Sackgasse identifiziert werden, aus N_Φ entfernt werden.
  - Sobald ein Q-S-Weg gefunden wird, wird die kleinste Restkapazität entlang des Weges zu Ψ hinzuaddiert. Dabei werden saturierte Kanten aus N_Φ entfernt.

Sperrflussberechnung

Algorithmus 4.11 (Sperrflussberechnung)

1. $\Psi := 0$
2. $i := 0$; $v_i := Q$ {∗ i: *aktuelles Niveau*, v_i: *aktueller Knoten* ∗}
3. While $v_i \neq S$
4. If $\exists (v_i, w) \in N_\Phi$
5. Then $i := i + 1$; $v_i := w$
6. Else
7. Entferne v_i und Kanten $(\cdot, v_i)$ aus N_Φ.
8. $i := i - 1$; If $i < 0$ Then Exit mit Ausgabe Ψ.
9. $r := \min\{r_\Phi((v_j, v_{j+1})) - \Psi((v_j, v_{j+1})) \mid 0 \leq j < i\}$
10. Für alle $j \in \{0, 1, \ldots, i - 1\}$
11. $\Psi((v_j, v_{j+1})) := \Psi((v_j, v_{j+1})) + r$
12. If $\Psi((v_j, v_{j+1})) = r_\Phi((v_j, v_{j+1}))$
    Then Entferne (v_j, v_{j+1}) aus N_Φ.
13. Weiter bei 2.

Zur Sperrflussberechnung

Lemma 4.12

Algorithmus 4.11 berechnet in Zeit $O(n \cdot e)$ einen Sperrfluss.

Beweis.

zur Korrektheit

- Algorithmus terminiert $\Leftrightarrow$ $\nexists$ Q-S-Weg
- entfernte Knoten/Kanten saturiert oder in Sackgassen
- also am Ende Sperrfluss

zur Laufzeit

- bei jeder (erfolgreichen und erfolglosen) Pfadkonstruktion ≥ 1 Kante entfernt
- $\leq e$ Pfadkonstruktionen
- je Pfadkonstruktion Zeit O(Pfadlänge)
- Länge jedes Q-S-Weges $\leq n$
- also insgesamt Zeit $O(n \cdot e)$ □

Der Algorithmus von Dinic

Algorithmus 4.13 (Algorithmus von Dinic)

1. $\Phi := 0$
2. Repeat
3. Berechne Niveaunetzwerk N_Φ.
4. Berechne Sperrfluss Ψ mit Algorithmus 4.11.
5. „$\Phi := \Phi + \Psi$"
6. Until $\Psi = 0$
7. Ausgabe Φ

Theorem 4.14

Der Algorithmus von Dinic berechnet in Zeit $O(n^2 \cdot e) = O(n^4)$ einen maximalen Fluss.

Zum Algorithmus von Dinic

Beobachtung	Korrektheit und Optimalität offensichtlich $\checkmark$
Erinnerung	Sperrflussberechnung in Zeit $O(n \cdot e)$

Distanzen in Graphen

Betrachte gerichteten Graphen $G = (V, E)$, Knoten $Q \in V$.
Für $v \in V$ bezeichnet $d(v)$ die Länge (=Anzahl Kanten) eines kürzesten Weges von Q nach v.

Lemma 4.8

Gegeben $v, w \in V$, Distanzen d,
nach Einfügen von (v, w): Distanzen d^+,
nach Entfernen von (v, w): Distanzen d^-.

1. $(d(v) \geq d(w) - 1) \Rightarrow (\forall u \in V : d^+(u) = d(u))$
2. $\forall u \in V$ mit $d(u) \leq d(w) - 1$: $d^-(u) = d(u)$
3. $\forall u \in V : d^-(u) \geq d(u)$

Beweis von Lemma 4.8

Erinnerung d^+: nach Einfügen von (v, w),
d^-: nach Entfernen von (v, w)

1. zu zeigen $(d(v) \geq d(w) - 1) \Rightarrow (\forall u \in V : d^+(u) = d(u))$
   trivial $\forall u \in V : d^+(u) \leq d(u)$
   klar $(d^+(u) < d(u)) \Rightarrow (v, w)$ liegt auf kürzestem Weg
   Beobachtung $d(w) = d^+(w)$
   weil Länge des Weges $(Q \rightsquigarrow v \rightarrow w) = d(v) + 1 \geq d(w)$
   also Kante (v, w) keine Abkürzung ✓
2. zu zeigen $\forall u \in V$ mit $d(u) \leq d(w) - 1$: $d^-(u) = d(u)$
   Voraussetzung (v, w) existiert in G
   also $d(w) \leq d(v) + 1 \Leftrightarrow d(w) - 1 \leq d(v)$
   wir haben $d(u) \leq d(w) - 1 \leq d(v)$
   also (v, w) auf kürzestem Weg zu u nicht vorhanden ✓
3. $\forall u \in V : d^-(u) \geq d(u)$: trivial ✓

Zum Algorithmus von Dinic

Beobachtung Korrektheit und Optimalität offensichtlich $\surd$
Erinnerung Sperrflussberechnung in Zeit $O(n \cdot e)$

Also genügt zu zeigen $O(n)$ Sperrflussberechnungen

Wir zeigen: kürzeste Q-S-Wege werden in jeder Runde länger
Da Weglänge $\leq n \Rightarrow O(n)$ Sperrflussberechnungen

Betrachte dazu Fluss Φ und Nachfolger $\Phi' = \Phi + \Psi$ mit $\Psi \neq 0$
und Rest_Φ und $\text{Rest}_{\Phi'}$

Annahme: P ist kürzester Q-S-Weg in $\text{Rest}_{\Phi'}$ und ist nicht länger als die kürzesten Wege in Rest_Φ .

Wir führen diese Annahme zum Widerspruch

Beweis zur Laufzeit des Algorithmus von Dinic

wir zeigen: kürzeste Q-S-Wege werden in jeder Runde länger

Betrachte dazu Fluss Φ und Nachfolger $\Phi' = \Phi + \Psi$ mit $\Psi \neq 0$
und Rest_Φ und $\text{Rest}_{\Phi'}$

Annahme: P ist kürzester Q-S-Weg in $\text{Rest}_{\Phi'}$ und ist nicht länger als die kürzesten Wege in Rest_Φ .

Welche Möglichkeiten gibt es (bezogen auf P)?

1. Fall P benutzt nur Kanten in Rest_Φ
$\Rightarrow$ alle kürzesten Q-S-Wege in Rest_Φ waren saturiert
$\Rightarrow$ diese sind also nicht mehr in $\text{Rest}_{\Phi'}$ vorhanden
$\Rightarrow$ P muss also echt länger sein $\Rightarrow$ Widerspruch ✓

2. Fall Es gibt Kante in P, die in Rest_Φ nicht vorhanden ist

2. Fall Es gibt Kante in P, die in Rest_Φ nicht vorhanden ist

Betrachte $e = (v, u)$ in P, e in $\text{Rest}_{\Phi'}$, e nicht in Rest_Φ

Entweder $(v, u) \in E$, also in Rest_Φ Rückwärtskante (u, v) benutzt
oder $(v, u) \notin E$, also in Rest_Φ „normale Kante" (u, v) benutzt

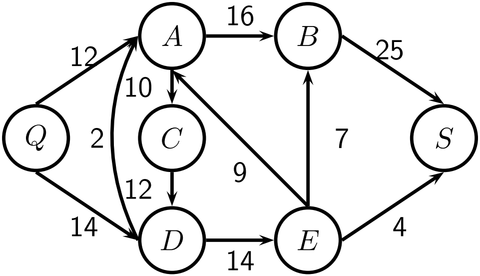
klar in N_Φ nur Kanten (u, v) mit $d(v) = d(u) + 1$
also alle neu eingefügten Kanten in $\text{Rest}_{\Phi'}$ erfüllen die Voraussetzung zu Lemma 4.8(1)

also Wege werden nicht kürzer

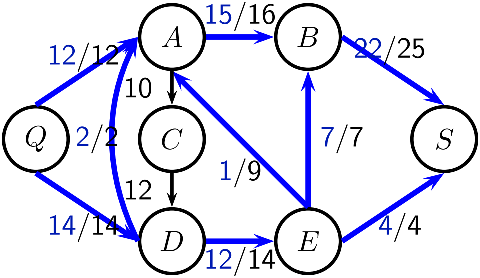
klar Wege werden durch entfernen saturierter Kanten höchstens länger (Lemma 4.8(3))

klar so ein Weg ist: $P = Q \rightsquigarrow v \to u \rightsquigarrow S$
also P muss mind. 2 länger sein als kürz. Q-S-Wege in $\text{Rest}_\Phi \Rightarrow$
Widerspruch ✓ □

Beispiel Algorithmus von Dinic



Eingabe Netzwerk



Netzwerk mit max. Fluss

Übersicht – Flussalgorithmen

Algorithmus von Ford und Fulkerson

- berechnet FVW im Restgraphen
- Laufzeit $O(B \cdot (n + e))$
- Variante: berechne kürzeste FVW mit Breitensuche:
  Laufzeit $O(n \cdot e^2) = O(n^5)$

Algorithmus von Dinic

- neue Idee: Niveaunetzwerk, Sperrflüsse
- Laufzeit $O(n^2 \cdot e) = O(n^4)$
  - $O(n)$ Sperrfluss-Berechnungen
  - $O(n \cdot e)$ für eine Sperrflussberechnung . . . ziemlich lang!

Algorithmus von Malhotra, Pramodh Kumar & Maheshwari

Zeit $O(n \cdot e)$ für nur eine Sperrflussberechnung ist ziemlich lang.

Kann man hier Zeit sparen?

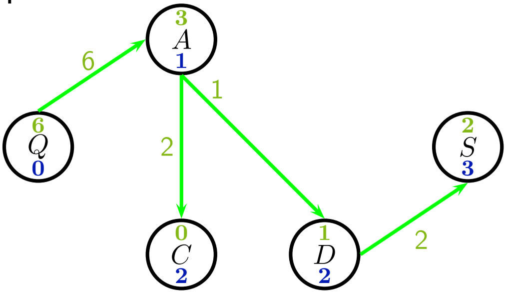
Idee neue Sperrflusskonstruktion: Knoten-basiert

Zentraler Begriff Potenzial pot(v) eines Knotens v:
im Niveaunetzwerk N_ϕ bei aktuellem Fluss ψ
(zum Fluss ϕ und Restgraphen Rest$_\phi$)

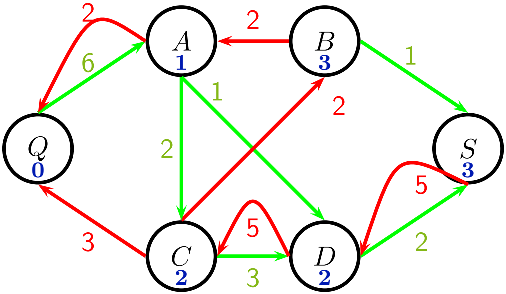
Intuition pot$(v) \approx$ wieviel Fluss höchstens durch v zusätzlich kann

$$\text{pot}(v) := \begin{cases} \min\left\{ \sum\limits_{e=(v,\cdot)} r_\phi(e) - \psi(e), \sum\limits_{e=(\cdot,v)} r_\phi(e) - \psi(e) \right\} & \text{für } v \notin \{Q, S\} \\ \sum\limits_{e=(Q,\cdot)} r_\phi(e) - \psi(e) & \text{für } v = Q \\ \sum\limits_{e=(\cdot,S)} r_\phi(e) - \psi(e) & \text{für } v = S \end{cases}$$

Beispiel Potenzial



Niveaunetzwerk



Restgraph

Idee zur Sperrflussberechnung

Intuition pot$(v) \approx$ wieviel Fluss höchstens durch v zusätzlich kann

Idee

- Wähle den Knoten v mit minimalem pot(v).
- Treibe Fluss pot(v) bis zur Senke S. (Forward)
- Treibe Fluss pot(v) zurück bis zur Quelle Q. (Backward)

Wir berechnen Sperrflüsse im Niveaunetzwerk $\rightsquigarrow$ azyklisch!

Fluss $v \rightsquigarrow S$

Forward(v)

1. Für alle $w \in V$ Überschuss$[w] := 0$
2. Überschuss$[v] :=$ pot(v)
3. $Qu := \emptyset$; Qu.Enqueue(v) *{∗ Qu: Knoten mit Überschuss > 0 ∗}*
4. While $Qu \neq \emptyset$
5. $\quad$ $u := Qu$.Dequeue() *{∗ u: aktiver Knoten ∗}*
6. $\quad$ Für alle $e = (u, w) \in E_\phi$ *{∗ und solange Überschuss$[u] > 0$ ∗}*
7. $\quad\quad$ $\delta := \min\{r_\Phi(e) - \Psi(e),$ Überschuss$[u]\}$
8. $\quad\quad$ $\Psi(e) := \Psi(e) + \delta$
9. $\quad\quad$ Update pot(u), pot(w)
10. $\quad\quad$ If Überschuss$[w] = 0$ und $\delta > 0$ und $w \neq S$
    $\quad\quad\quad$ Then Qu.Enqueue(w)
11. $\quad\quad$ Überschuss$[w] :=$ Überschuss$[w] + \delta$;
    $\quad\quad$ Überschuss$[u] :=$ Überschuss$[u] - \delta$
12. $\quad\quad$ If $\Psi(e) = r_\Phi(e)$ Then Entferne e aus E_ϕ
13. $\quad\quad$ If Überschuss$[u] = 0$ Then break

Fluss $v \rightsquigarrow Q$

Backward(v, p_v)

1. Für alle $w \in V$ Überschuss$[w] := 0$
2. Überschuss$[v] := p_v$
3. $Qu := \emptyset$; Qu.Enqueue(v) *{∗ Qu: Knoten mit Überschuss > 0 ∗}*
4. While $Qu \neq \emptyset$
5. $u := Qu$.Dequeue() *{∗ u: aktiver Knoten ∗}*
6.   Für alle $e = (w, u) \in E_\phi$ *{∗ und solange Überschuss$[u] > 0$ ∗}*
7. $\delta := \min\{r_\Phi(e) - \Psi(e), \text{Überschuss}[u]\}$
8. $\Psi(e) := \Psi(e) + \delta$
9.   Update pot(u), pot(w)
10.   If Überschuss$[w] = 0$ und $\delta > 0$ und $w \neq Q$
            Then Qu.Enqueue(w)
11.   Überschuss$[w]$:= Überschuss$[w] + \delta$;
        Überschuss$[u]$:= Überschuss$[u] - \delta$
12.   If $\Psi(e) = r_\Phi(e)$ Then Entferne e aus E_ϕ
13.   If Überschuss$[u] = 0$ Then break

Schnellere Sperrflussberechnung

Algorithmus 4.16 (Forward-Backward-Propagation)

1. $\Psi := 0$
2. Für alle $v \in V_\phi$ berechne pot(v)
3. While $\{Q, S\} \subseteq V_\phi$
4. Wähle $v \in V_\phi$ mit minimalem Potenzial; $p_v :=$ pot(v)
5. If $p_v > 0$ Then Forward(v); Backward(v, p_v)
6. Entferne v aus V_ϕ und Kanten $(\cdot, v)$, $(v, \cdot)$ aus E_ϕ und update pot(u) $\forall u$ mit $(u, v) \in E_\phi$ oder $(v, u) \in E_\phi$

Lemma 4.17

Algorithmus 4.16 berechnet in Zeit $O(n^2)$ einen Sperrfluss in N_Φ.

Korrektheit von Forward-Backward-Propagation

Beobachtung $p_v = \min\{\text{pot}(v) \mid v \in V_\phi\}$
also gesamter Überschuss/Potenzial
als Fluss zur Senke und Quelle transportierbar

Beobachtung nachher alle eingehenden oder
alle ausgehenden Kanten von v saturiert
also v entfernbar

Beobachtung wenn Q oder S so saturiert → Sperrfluss ✓

schwieriger Laufzeitschranke $O(n^2)$

Laufzeit Sperrflussberechnung

klar Initialisierung in Zeit $O(n+e) = O(n^2)$

Beobachtung Potenzial wird in Forward/Backward korrigiert

Beobachtung $p_v = \min\{\text{pot}(v) \mid v \in V_\phi\}$ in $O(n)$
da $O(n)$ Iterationen $\Rightarrow O(n^2)$

Nun Analyse Forward (Backward analog):

Beobachtung nur w mit Überschuss$(w) = 0$ $(\delta > 0)$
kommen in die Queue (Zeile 10)

Beobachtung nur für u selbst sinkt Überschuss (Zeile 11)

also kein Knoten in einem Durchlauf mehrfach in Queue
(da Niveaunetzwerk azyklisch)

also je Forward-Durchlauf $O(n)$ Knoten in Queue $\Rightarrow O(n^2)$

es fehlt noch Schleife über Kanten in Forward

Forward: Die innere Schleife

Beobachtung für jede betrachtete Kante $e = (u, w)$ gilt:
entweder wird e saturiert oder Schleife endet

Aufteilung Laufzeit auf sat. und nicht-sat. Kanten
klar saturierte Kanten werden entfernt
also insgesamt Laufzeit $O(e)$ für saturierte Kanten

fehlen noch nicht-saturierte Kanten:
Beobachtung pro Knoten u nur eine nicht-saturierte Kante
da anschließend Überschuss[u]=0
da $\leq n$ Forward-Aufrufe:
insgesamt Laufzeit $O(n^2)$ für nicht-saturierte Kanten

zusammen Laufzeit $O(n^2 + e) = O(n^2)$ □

Algo. von Malhotra, Pramodh Kumar & Maheshwari

Algorithmus 4.18 (Algorithmus von Malhotra, Pramodh Kumar & Maheshwari)

1. $\Phi := 0$
2. Repeat
3. Berechne Niveaunetzwerk N_Φ.
4. Berechne Sperrfluss Ψ mit Algorithmus 4.16.
5. „$\Phi := \Phi + \Psi$“
6. Until $\Psi = 0$
7. Ausgabe Φ

Beobachtung wie Dinic mit anderer Sperrflussberechung

Über den Algorithmus von Malhotra et. al

Theorem 4.19

Der Algorithmus von Malhotra, Pramodh Kumar und Maheshwari berechnet einen maximalen Fluss in Zeit $O(n^3)$.

Korrektheit wie bei Dinic

Anzahl Sperrflussberechnungen wie bei Dinic: $O(n)$

Aber Sperrflussberechnung mit Forward-Backward-Propagation in $O(n^2)$

also Gesamtlaufzeit $O(n \cdot n^2) = O(n^3)$

□

Beispiel Algorithmus von Malhotra, Pramodh Kumar und Maheshwari