Effiziente Algorithmen (SS2015)

Kapitel 2
Weitere Flüsse

Walter Unger

Lehrstuhl für Informatik 1

13:58 Uhr, den 19. Dezember 2018

Inhalt I

1 Spezielle Flüsse (Mindestfluss)
- Mit Mindestfluss

2 Spezielle Flüsse (Alternativen)
- Mit Alternativen

3 Flüsse mit Kostenfunktion
- Einleitung
- Idee
- Algorithmus
- Verbesserung der Laufzeit

Das Flussproblem mit Mindestfluss

Definition (Flussproblem mit Mindestfluss)

Eingabe: $G = (V, E, s, t, c, c')$ mit:

- (V, E) ist ein gerichteter Graph $(n = |V|, m = |E|)$
- $s, t \in V$ mit $s \neq t$
- $c : E \mapsto \mathbb{N}^+$
- $c' : E \mapsto \mathbb{N}^+$

Ausgabe: $f : E \mapsto \mathbb{R}_0^+$ mit:

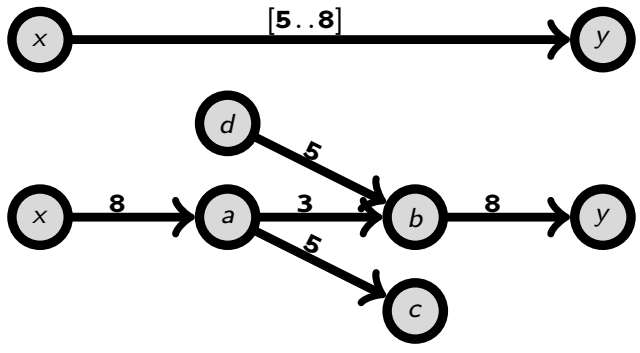
- $\forall e : c'(e) \leqslant f(e) \leqslant c(e)$
- $\forall v \in V \setminus \{s, t\} : \sum_{(a,v)\in E} f((a, v)) = \sum_{(v,a)\in E} f((v, a))$

Ziel: Bestimme, ob es so einen Fluss gibt. Falls ja, dann maximiere $w(f) = \sum_{(s,v)\in E} f((s, v))$.

Idee

Die Motivation beim Mindestfluss ist, daß bei einem Kanalsystem immer ein Grundfluss vorhanden ist, um ein Versanden zu verhindern. Bemerkung, so ein Problem hat im Allgemeinen nicht immer eine Lösung.

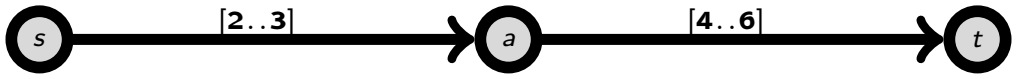
Algorithmisch ist das aber trotzdem kein Problem, da wir ein Flussproblem mit Mindestfluss auf das normale Flussproblem reduzieren können.



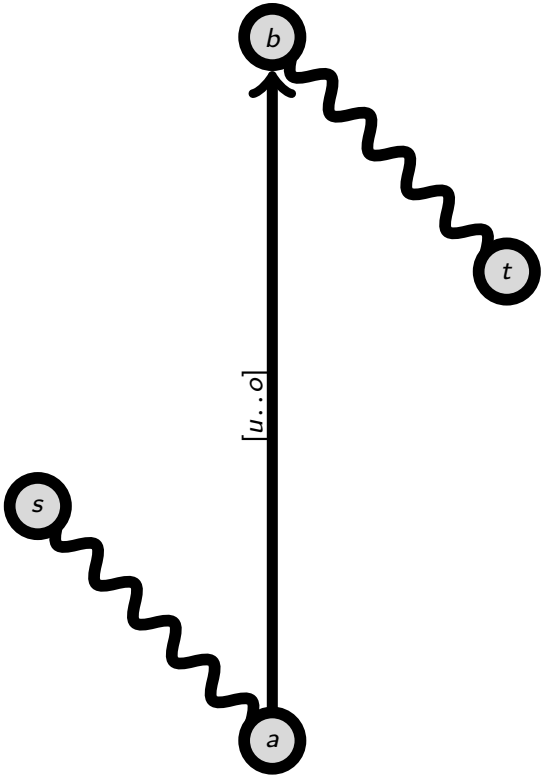
Dazu wird eine Kante, die einen Fluss zwischen 5 und 8 haben soll, durch einen Pfad der Länge drei dargestellt. Die mittlere Kante des Pfades simuliert die Variablilität des Flusses. In unserem Beispiel kann der Fluss zwichen 5 und 8 liegen, also um den Wert 3 variieren. Daher erhällt die mittlere Kante eine Kapazität von 3. Die beiden andern Kanten (Endkanten des Pfades) erhalten als Kapazität 8. Damit die beiden Endkanten einen Fluss zwischen 5 und 8 bekommen, wird mit zwei Zusatzkanten zwischen vor und hinter der mittleren Kante ein Fluss von maximal 5 eingespeist (bzw. herausgenommen). Im Folgenden sehen wir, daß diese Modelierung korrekt ist.

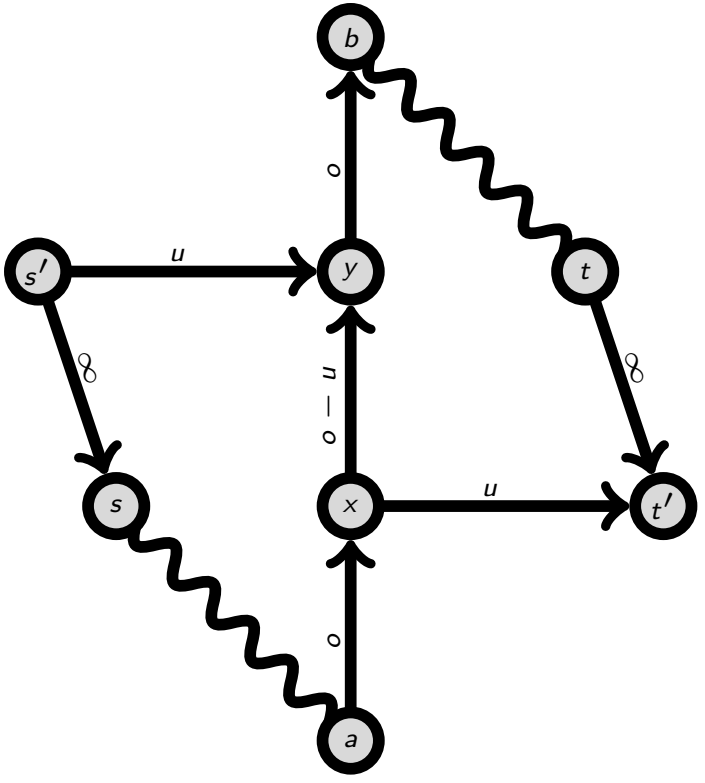
Lösbarkeit

Es muss nicht immer eine Lösung geben. Hier ein einfaches Beispiel:



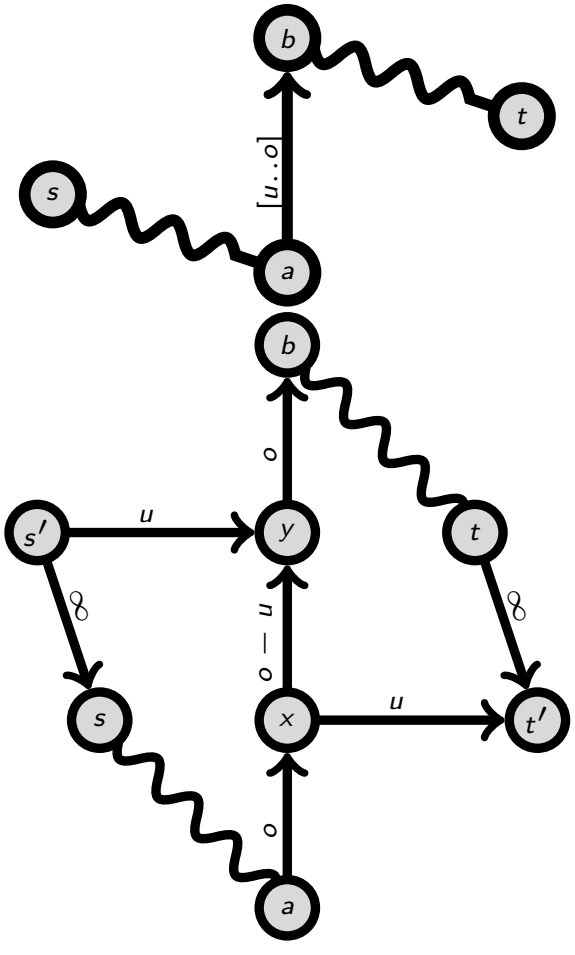
Idee





Verfahren

- Erzeuge aus $G = (V, E, s, t, c, c')$ einen neuen Graphen $G' = (V', E', s', t', c'')$.
- Füge neue Quelle s' und neue Senke t' hinzu.
- Ersetze jede Kante (v, w) durch einen Weg der Länge 3:
  - für jede Kante (v, w) erzeuge zwei neue Knoten x, y und setze:
  - $c''(v, x) = c(v, w)$ und $c''(y, w) = c(v, w)$.
  - $c''(x, y) = c(v, w) - c'(v, w)$.
  - $c''(s', y) = c'(v, w)$ und $c''(x, t') = c'(v, w)$.
- Setze $c''(t, t') = c''(s', s) = \sum_{e \in E} c(e)$.



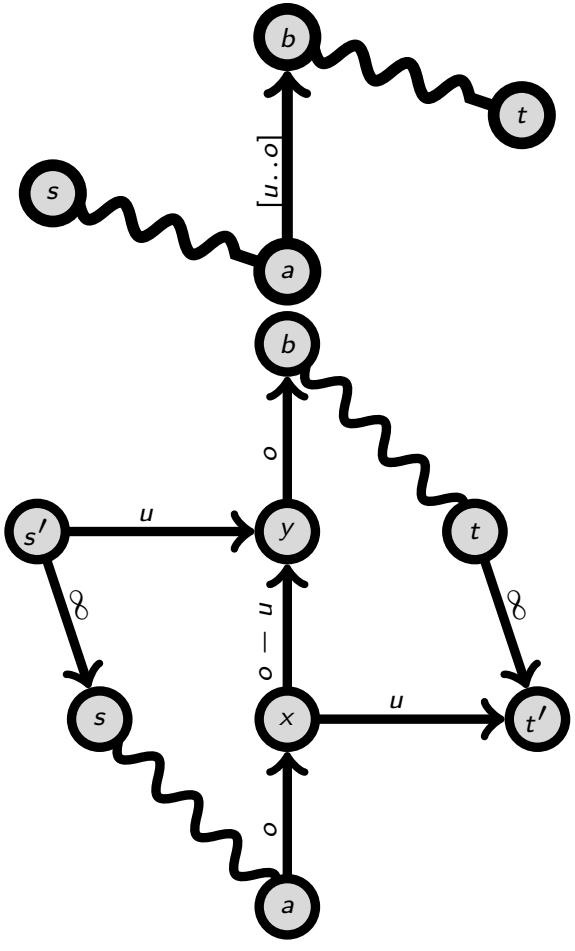
Aussage

Lemma

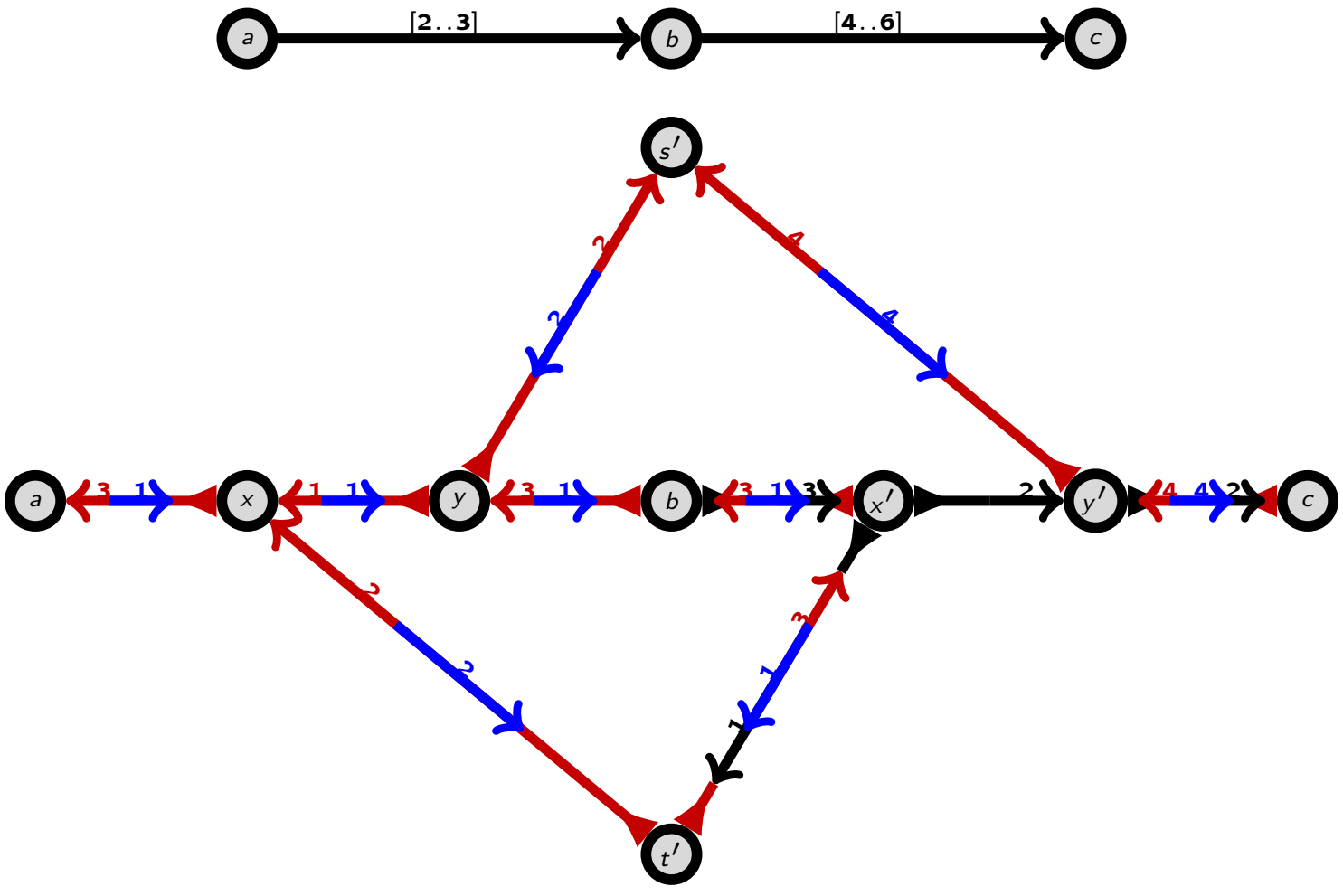
Es gibt in G einen korrekten Fluss, der die Mindestflussbedingung erfüllt genau dann, wenn es in G' einen maximalen Fluss gibt, der alle Kanten der Form (s', y) und (x, t') saturiert.

Beachte: x und y sind neu eingefügte Knoten, also nicht s oder t.
Beweis:

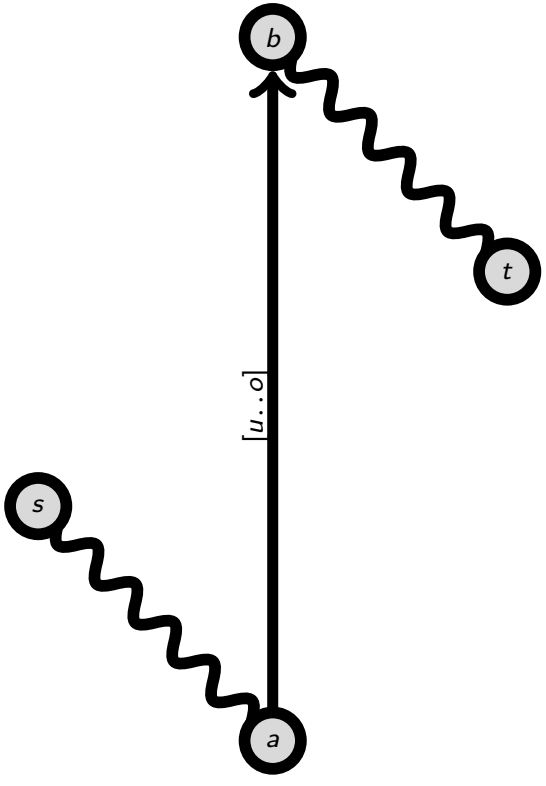
- Zeige: $\Longrightarrow$
- Zeige: $\Longleftarrow$



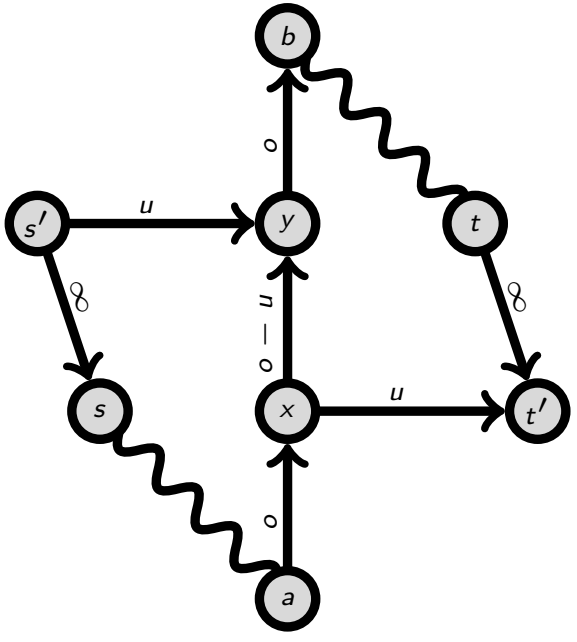
Beispiel (warum (s', y) und (x, t'))



Zeige: $\Longrightarrow$



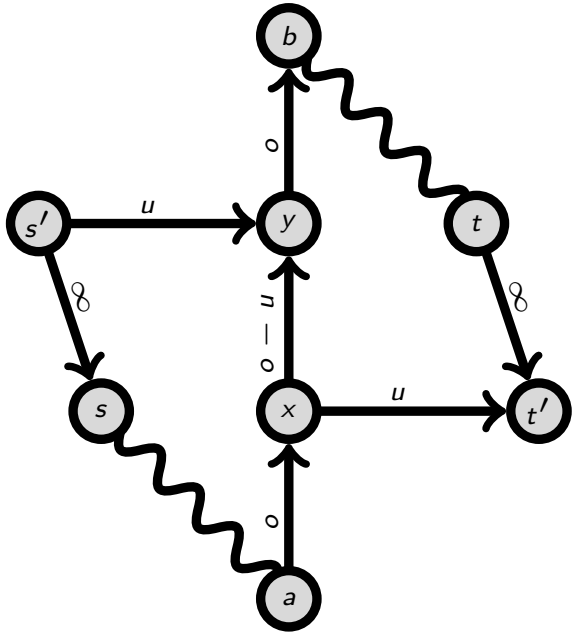
$u \leqslant f(a,b) \leqslant o$



$f(s',y) = f(x,t') = u$
$f(a,x) = f(y,b) = f(a,b)$
$f(x,y) = f(a,b) - u$

Zeige: $\Longleftarrow$

- Zeige: Wenn es in G' einen maximalen Fluss gibt, der alle Kanten der Form (s', y) und (x, t') saturiert, dann gibt es in G einen korrekten Fluss, der die Mindestflussbedingung erfüllt.
- Dann gilt: $f(a, x) = f(y, b)$ für jede ursprüngliche Kante (a, b).
- Dann definiert $f(a, b) = f(a, x)$ einen korrekten Fluss auf G.



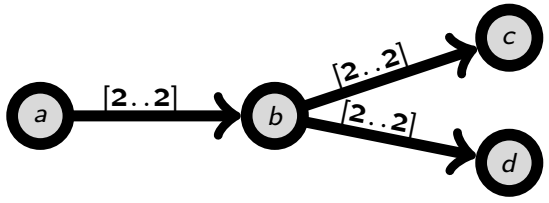
Algorithmus

- Sei f (resp. f') der Fluss auf G (resp. G').
- Sei weiter f'' der Fluss auf G ohne die untere Schranke c'. Dann gilt, wenn es eine Lösung für G' gibt:
  - $f = f''$, denn untere Schranken verringern den Fluss auf G nicht mehr.
  - $f' = f + \sum_{e \in E(G)} c'(e)$
- Damit haben wir folgendes Verfahren:
  1. Bestimme aus G: G', G'', f, f', f''.
  2. Bevorzuge auf G' die Kanten der Form (s', y) und (x, t').
  3. Falls $f' < f + \sum_{e \in E(G)} c'(e)$ gilt, so gibt es keine Lösung.
  4. Ansonsten bestimme f aus f', d.h. $f(a, b) = f(a, x)$.

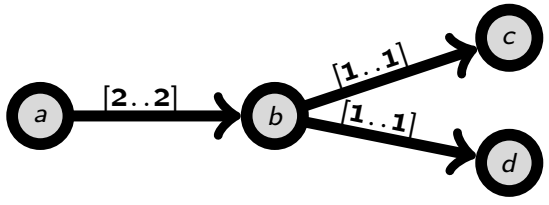
Beschreibung der Situation

Wenn man das obrige Flussmodell erweitert und erlaubt, daß eine Kante auch gar keinen Fluss haben kann, dann ändert sich die Komplexität gravierend. Denn nun werden die folgenden Situationen möglich.

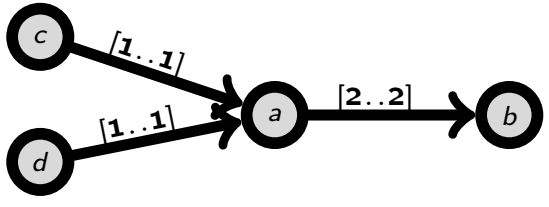
Ein Fluss auf einer Kante kann nur auf einer von zwei ausgehenden Kanten weiterfließen.



Ein Fluss auf einer Kante muss sich genau auf zwei ausgehende Kanten verteilen.



Eine Kante kann nur dann einen Fluss haben, wenn zwei eingehende Kanten auch ein Fluss ist.



Damit erscheint aber eine Reduktion auf ein NP-hartes Problem möglich. Denn die oben genannten Situationen sehen aus wie: Auswahl, verteilen von Information, kombinieren von Information.

Das Flussproblem mit Alternativen

Definition (Flussproblem)

Eingabe: $G = (V, E, s, t, c, c')$ mit:

- (V, E) ist ein gerichteter Graph $(n = |V|, m = |E|)$
- $s, t \in V$ mit $s \neq t$
- $c : E \mapsto \mathbb{N}^+$
- $c' : E \mapsto \mathbb{N}^+$

Ausgabe: $f : E \mapsto \mathbb{R}_0^+$ mit:

- $\forall e : c'(e) \leqslant f(e) \leqslant c(e)$ oder $f(e) = 0$.
- $\forall v \in V \setminus \{s, t\} : \sum_{(a,v)\in E} f((a, v)) = \sum_{(v,a)\in E} f((v, a))$

Ziel: Bestimme, ob es so einen nicht trivialen Fluss gibt. Falls ja, dann maximiere $w(f) = \sum_{(s,v)\in E} f((s, v))$.

Reduktion

Theorem

Zu einem gegeben Flussproblem $G = (V, E, s, t, c, c')$ ist es NP-vollständig zu bestimmen, ob es so einen nicht trivialen Fluss gibt.

Beweis: Übung, b.z.w. Reduktion auf Exact-3-SAT.

Exact-3-SAT

Definition

Eine Boolesche Formel $\mathcal{F}$ ist in Exact-3-KNF:

$$\mathcal{F}(x_1, x_2, ..., x_r) = \bigwedge_{i=1}^{k} c_i$$

$$(\textit{Klauseln}) \quad c_i = (l_i^1 \vee l_i^2 \vee l_i^3) \qquad \forall\ 1 \leqslant i \leqslant k$$

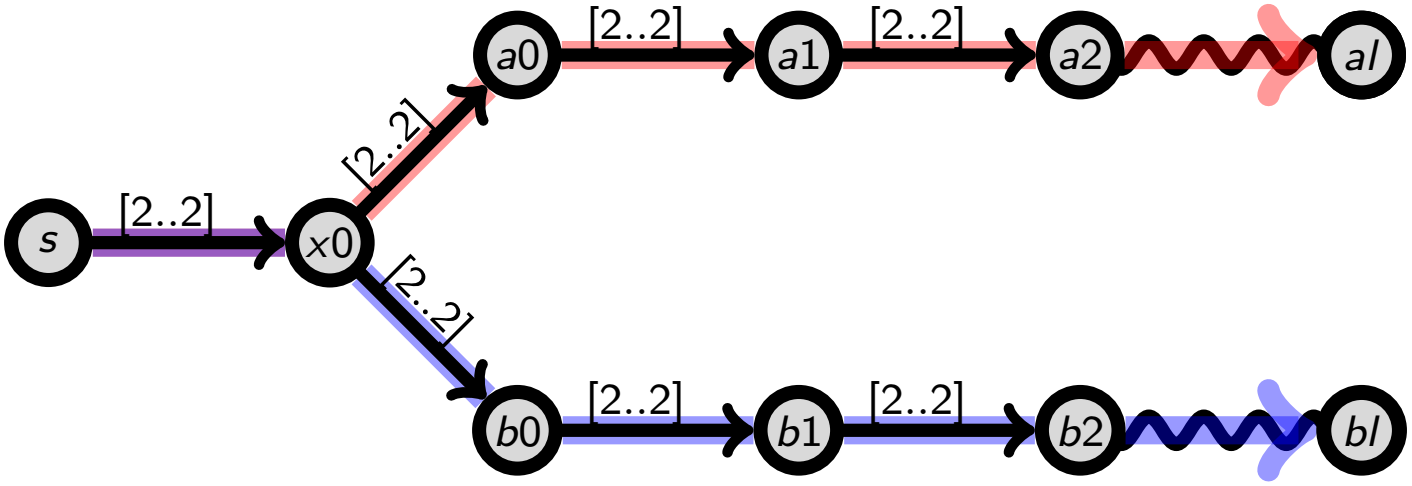
$$(\textit{Literale}) \quad l_i^j = \left\{ \begin{array}{ll} \neg x_l & \text{oder} \\ x_l & \text{für ein } l : 1 \leqslant l \leqslant r \end{array} \right\} \quad \begin{array}{l} \forall\ 1 \leqslant i \leqslant k \text{ und} \\ \forall\ 1 \leqslant j \leqslant 3 \end{array}$$

Eine Belegung ist eine Funktion $W : \{x_1, x_2, ..., x_r\} \mapsto \{0, 1\}$.

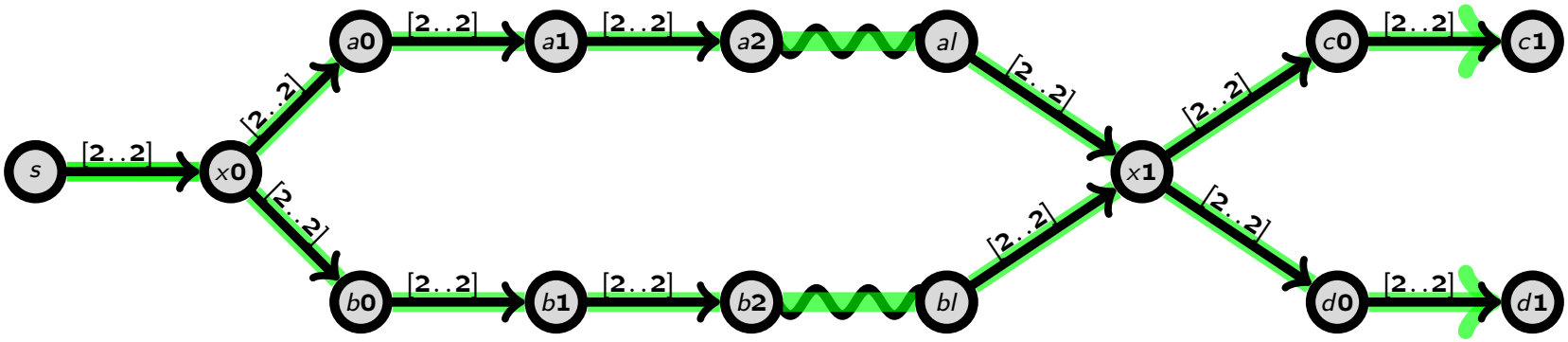
Theorem (Exakt-3-SAT)

Es ist NP-vollständig, festzustellen, ob es für $\mathcal{F}$ aus Exact-3-KNF eine erfüllende Belegung gibt, bei der in jeder Klausel genau ein Literal "true" ist.

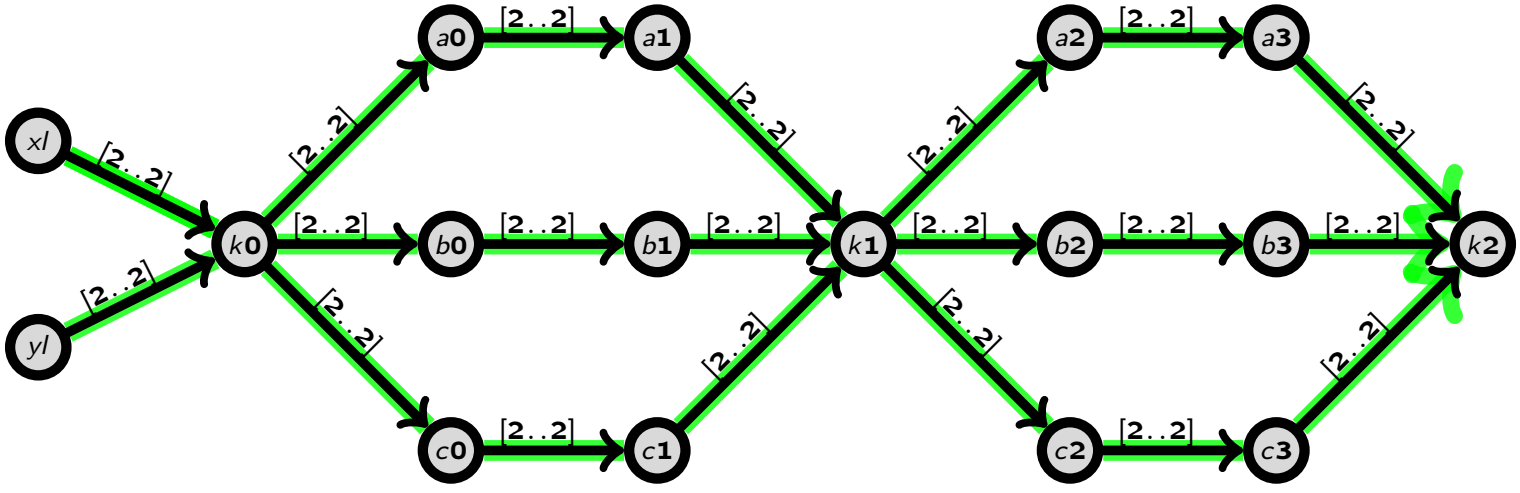
Erste Variable x_0



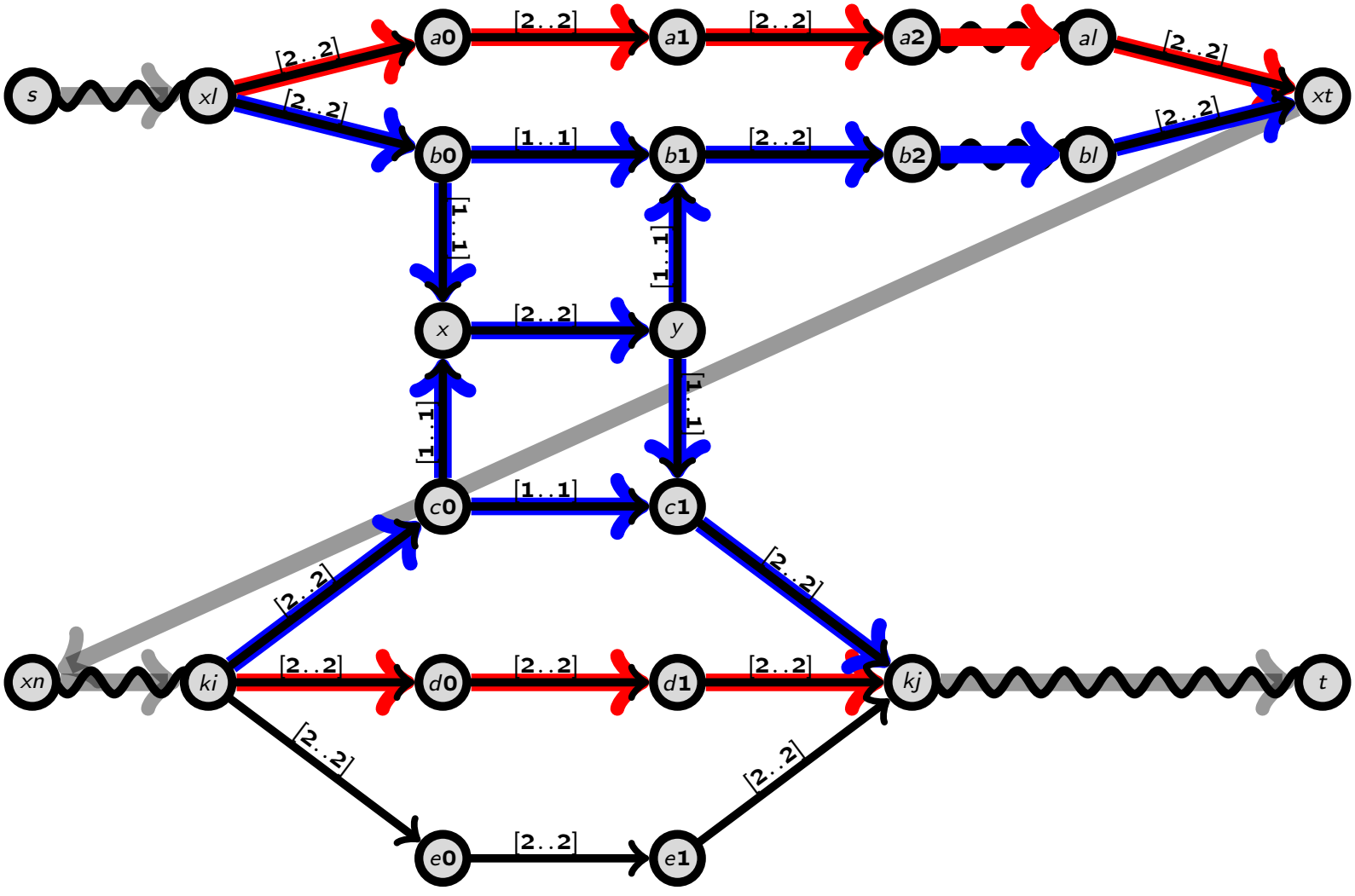
Zweite Variable x_1



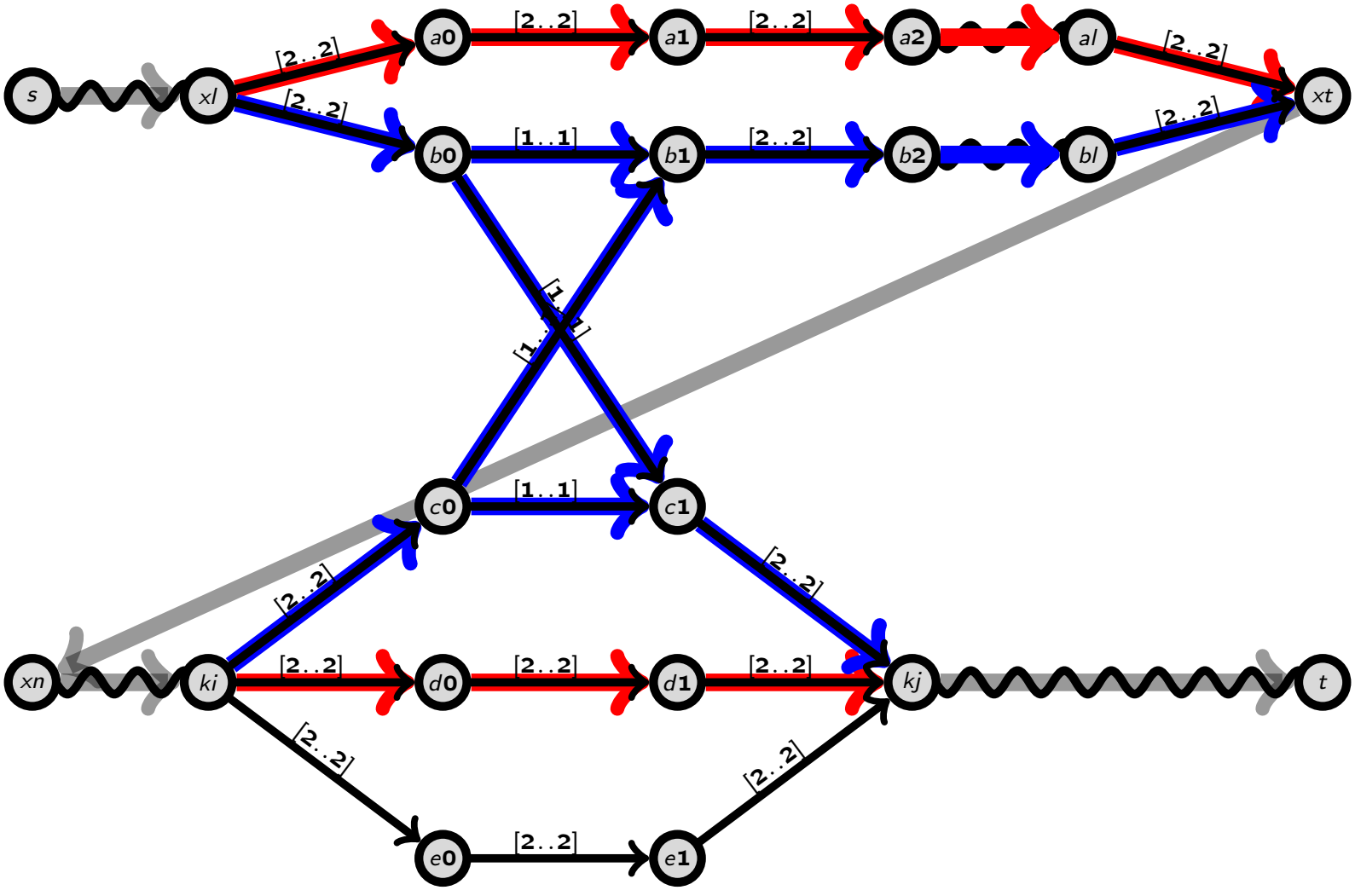
Erste zwei Klauseln k_0 und k_1



Anpassung: Variable x_l in Klausel k_i



Alternative Anpassung: Variable x_l in Klausel k_i



Konstruktion

1. Für jede Variable konstruiere einen Baustein, wie oben.
   - Der obere Zweig entspricht der Variablen selber.
   - Der untere Zweig entspricht der negierten Variablen.
2. Für jede Klausel konstruiere einen Baustein, wie oben.
   - Der erste Zweig entspricht dem ersten Literal in der Klausel.
   - Die weiteren Zweige dem zweiten und dem dritten Literal in der Klausel.
3. Hänge alle Bausteine für die Variablen und Klauseln hintereinander.
4. Für jedes Auftreten eines Literals in einer Klausel mache die obige Anpassung.
5. Falls es eine Belegung der Variablen gibt, die die Formel erfüllt, dann:
   - geht ein Fluss von 2 durch jeweils den Zweig, der der Belegung der Variablen entspricht.

Motivation

- Benutzung der Kanten (Transportwege) im allgemeinen nicht umsonst.
- Daher sollten wir die Kosten minimieren.
- Kosten sind minimal, wenn der Fluss Null ist.
- Daher suchen wir:

  Kostenminimalen Fluss mit Wert W.
- Wichtig: G sollte keine Kreise mit negativen Kosten (Gewichtssumme) enthalten.

Das Flussproblem mit Kosten

Definition (Min-Cost-Flow-Problem)

Eingabe: $G = (V, E, s, t, c, l), W$ mit:

- (V, E) ist ein gerichteter Graph $(n = |V|, m = |E|)$
- $s, t \in V$ mit $s \neq t$
- $c : E \mapsto \mathbb{N}^+$
- $l : E \mapsto \mathbb{Z}$ und $W \in \mathbb{N}$

Ausgabe: $f : E \mapsto \mathbb{R}_0^+$ mit:

- $\forall e : f(e) \leqslant c(e)$.
- $\forall v \in V \setminus \{s, t\} : \sum_{(a,v)\in E} f((a, v)) = \sum_{(v,a)\in E} f((v, a))$

Ziel: Bestimme Fluss f $w(f) = W$ und
minimalen $l(f) = \sum_{e\in E} f(e) \cdot l(e)$.
D.h. $l(f) = min\{l(g) \mid g$ ist Fluss mit $w(g) = W\}$.

Erste Ideen

Man könnte versuchen direkt, einen solchen kostenminimalen Fluss zu bestimmen. Aber es erscheint einfacher, zuerst einen Fluss mit korrektem Flusswert zu bestimmen und dann von diesem ausgehend den kostenminimalen Fluss. Das wird im O-Kalkül wohl zu keinem Verlust führen.

Daher wird im Folgenden zuerst gezeigt, wie man einen Fluss mit gegebenen Wert bestimmt, um diesen dann zu verbessern.

Im zweiten Schritt werden dann Verbesserungen betrachtet. Man könnte – das ist der erste Versuch – zwei Wege zwischen Quelle und Senke betrachten. Wenn es diese gibt, so können wir die Kosten verbessern, falls wir auf dem teueren Weg Fluss entfernen und dafür auf den billigeren Weg diesen Fluss neu legen.

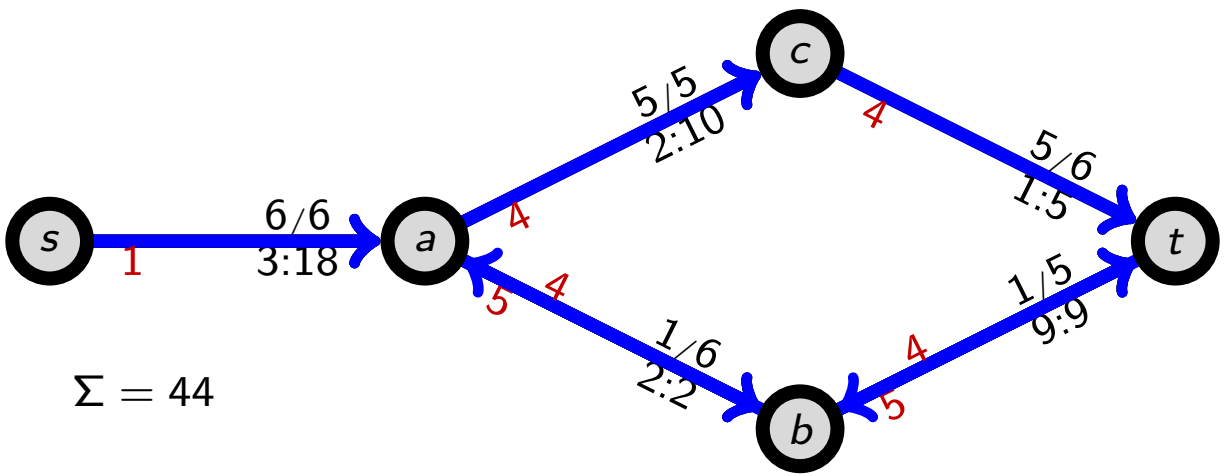
Damit sieht man: Es können einfach Kreise gesucht werden, die die Kosten verbessern. Man sieht aber auch im Folgenden, daß bei einer Suche nach beliebigen Kreisen eine pseudopolynomielle Laufzeit erhalten.

Beobachtungen

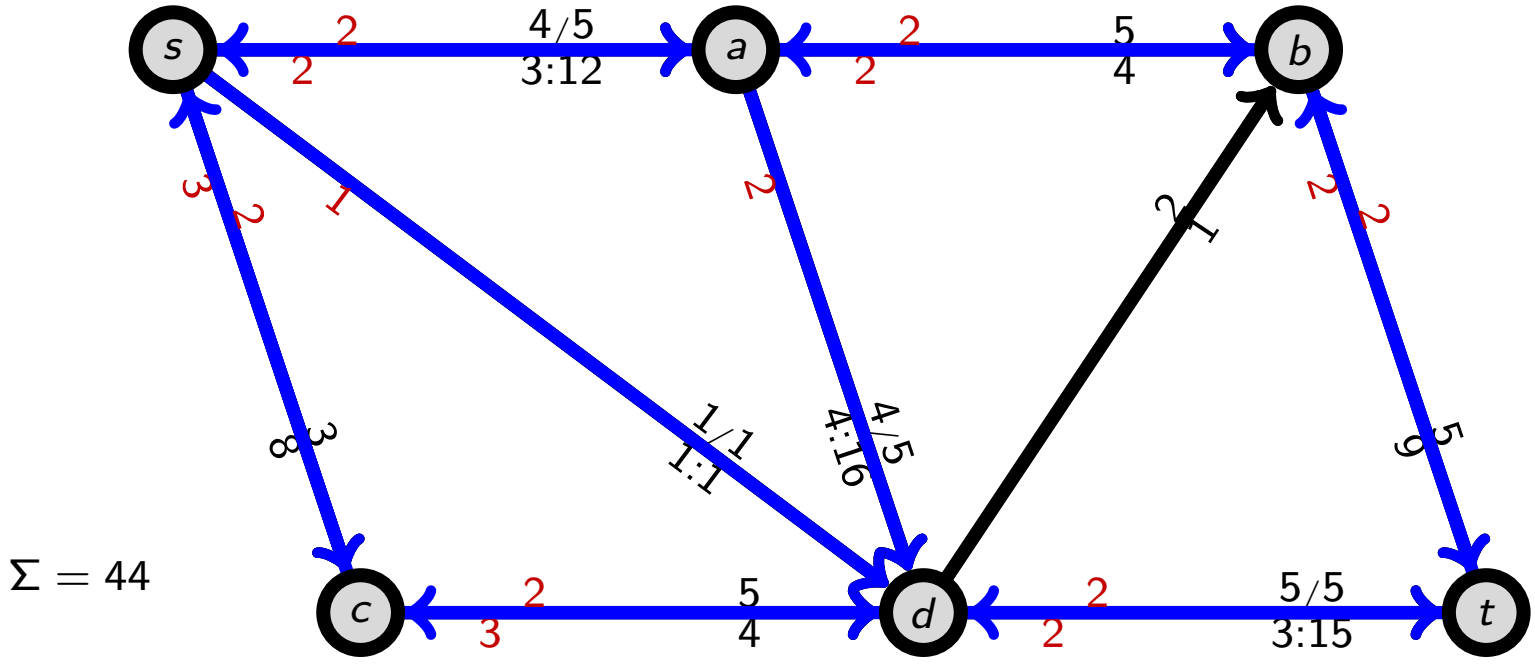
- Falls $W = 1$, so entspricht das dem kürzesten Wege Problem.
- Falls $l(e) = 0$ für alle $e \in E$, dann können die obigen Algorithmen leicht zur Lösung des Problems adaptiert werden:
  1. Falls erstmalig $w(f) \geqslant W$ gilt, dann breche ab.
  2. Sei p der Wert der letzten Erweiterung.
  3. Ersetze den letzten Fluss durch eine Fluss mit dem Wert $p - (w(f) - W)$.
- Alternativ kann auch wie folgt vorgegangen werden:
  1. Erzeuge neue Quelle s'.
  2. Füge Kante $e' = (s', s)$ mit $c(e') = W$ hinzu.
  3. Damit wird der maximale Fluss durche W begrenzt.

Idee (am Beispiel)

- Kantenbeschriftung:
  - oben: Fluss/MaxFluss und
  - darunter Kosten:Aktuelle Kosten
- Bestimme erst einen Fluss (mit Wert $W = 6$).
- Der Fluss muss nicht kostenoptimal sein.
- Verbessere die Kosten.



Beispiel ($W = 5$)

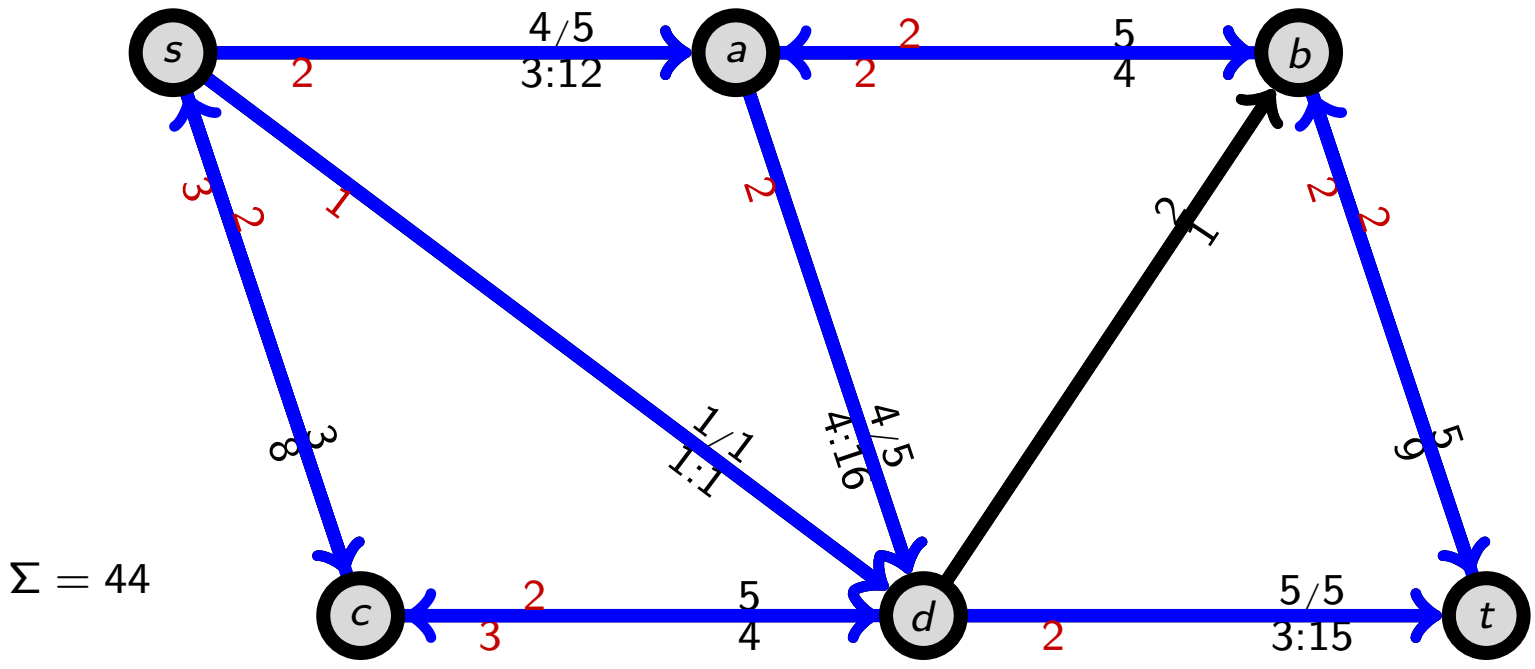


$\Sigma = 44$

Idee

- Bestimme einen beliebigen Fluss f mit $w(f) = W$.
- Verbessere schrittweise die Kosten des Flusses:
  - Annahme: es gibt Fluss f' mit $l(f') < l(f)$ und $w(f') = w(f)$.
  - Dann gibt es einen Unterschied zwischen f und f'.
  - Betrachte diesen Unterschied.
  - Das muss ein zyklischer Fluss sein, d.h. ein Fluss ohne Quelle und Senke.
    - Für f' und f gilt die Flusserhaltung.
    - Und $f_{out}(s) = f'_{out}(s) = f_{in}(t) = f'_{in}(t)$.
    - Damit gilt: Falls $f(a, b) \neq f'(a, b)$, dann gibt es $c \in V \setminus \{a, b\}$ mit: $f(a, c) \neq f'(a, c)$.
    - Damit gibt es mindestens einen Kreis mit Fluss g über Kanten e mit $f(e) \neq f'(e)$.
  - Dieser zyklische Fluss besteht aus einer Summe von Kreisen.
  - Einer dieser Kreise muss die Kosten für f' verbessern.
- Idee: suche diese verbessernden Kreise.

Beispiel mit Kreisen ($W = 5$)



$\Sigma = 44$

Frage: Warum eine Suche nach Kreisen?

- Ein verbessernder Kreis entspricht zwei Wegen:
  - Ein Weg, der gelöscht wird,
  - Ein Weg, der hinzugefügt wird.
  - Die Differenz der Wege ist der Kreis.
- Man könnte also auch nach Wegen suchen.
- Vorteil bei Kreisen:
  - Gewinn entspricht direkt den Kosten des Kreises.
  - Einfachere Algorithmen bei sich ändernden Kosten.
    - Optimale Lösung bekannt.
    - "Anbieter" verändert die Kosten einer Kante.
    - Passe bisherige Lösung durch das Suchen von Kreisen an.
  - Die folgenden Beweise werden dadurch einfacher.

Definitionen

- Gegeben $G = (V, E, s, t, c, l)$ und Restnetzwerk $G_f = (V', E', s, t, c', l')$.
- $c'(e) = c(e)$ für $e \in E \cap E'$
- $c'(a, b) = c(b, a)$ und $l'(a, b) = -l(b, a)$ für $(a, b) \in E' \setminus E$
- f' ist eine Zirkulation, gdw.: $\forall v \in V : f'_{in}(v) = f'_{out}(v)$.
- Wert einer Zirkulation f' über Schnitt (S, T):

$$f'(S, T) = \sum_{(v,w) \in E, v \in S, w \in T} f'(v, w) - \sum_{(w,v) \in E, v \in S, w \in T} f'(w, v).$$

- $w(f') = f'(\{s\}, V \setminus \{s\})$.

Lemma

Es gilt:

- $w(f') = 0$ *und*
- *für jeden Schnitt* (S, T) *gilt:* $f'(S, T) = 0$.

Beweis: Flusserhaltung und Induktion über Größe von S.

Zirkulation und Kostenminimalität

Lemma

Falls f nicht kostenminimal ist, dann gibt es einen Kreis C in G_f und f hat auf C negative Kosten.

Beweis:

- Sei f^* Fluss auf G mit $l(f^*) < l(f)$.
- Damit unterscheiden sich f^* und f.
- Für alle $e \in E$ setze $f'(e) = f^*(e) - f(e)$.
- f' ist dann zyklischer Fluss.
- f' wird durch höchstens $m' < m$ viele Kreisflüsse f'_i $(1 \leqslant i \leqslant m')$ gebildet.
- Damit gilt: $l(f') = l(f^*) - l(f) = \sum_{1 \leqslant i \leqslant m'} l(f'_i)$
- Damit existiert j mit $l(f'_j) < 0$.

Algorithmus (Min-Cost-Flow)

Theorem

Ein Fluss f ist kostenminimal, wenn G_f keinen Kreis mit negativen Kosten enthält.

1. Eingabe: $G = (V, E, s, t, c, l), W$.
2. Bestimme Fluss f' mit $w(f') = W$.
3. Solange es in G_f einen negativen Kreis C gibt:
   1. $f = f + f'(C)$.

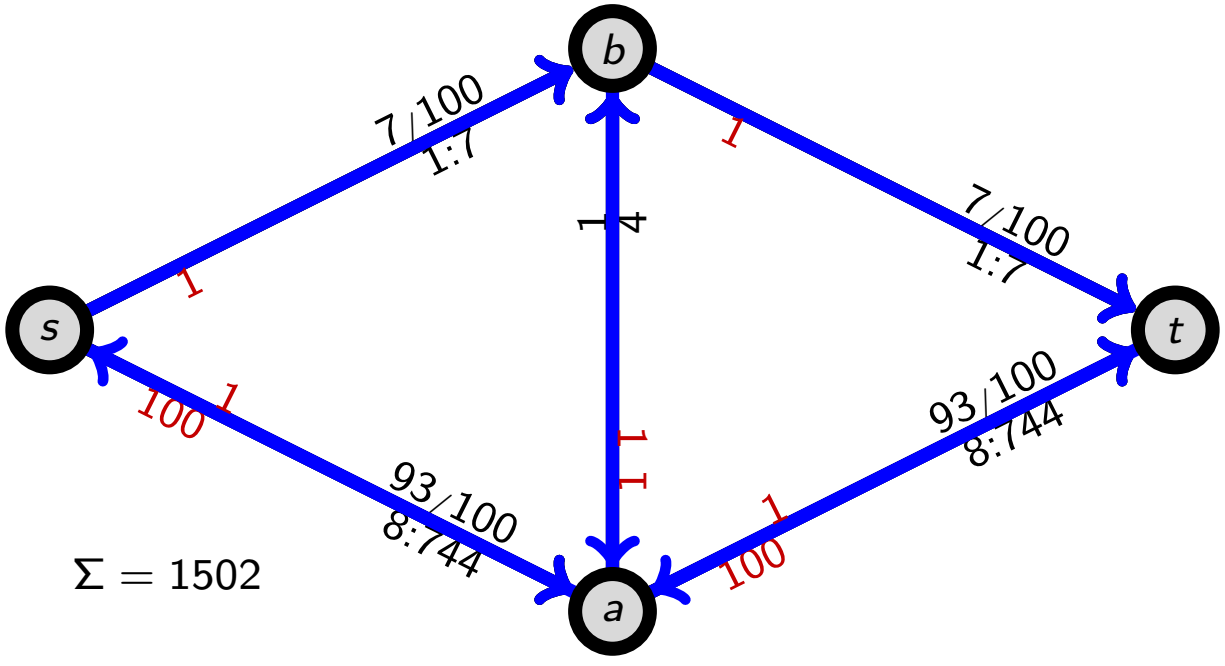
- Es gilt: $w(f + f'(C)) = w(f) + w(f'(C)) = w(f) = W$.
  - D.h. der Wert des Flusses bleibt gleich.
- Es gilt: $l(f + f'(C)) = l(f) + l(f'(C)) < l(f)$.
  - D.h. die Kosten verringern sich.

Theorem

Der obige Algorithmus bestimmt einen kostenminimalen Fluss.

Beispiel zur Laufzeit ($W = 100$)

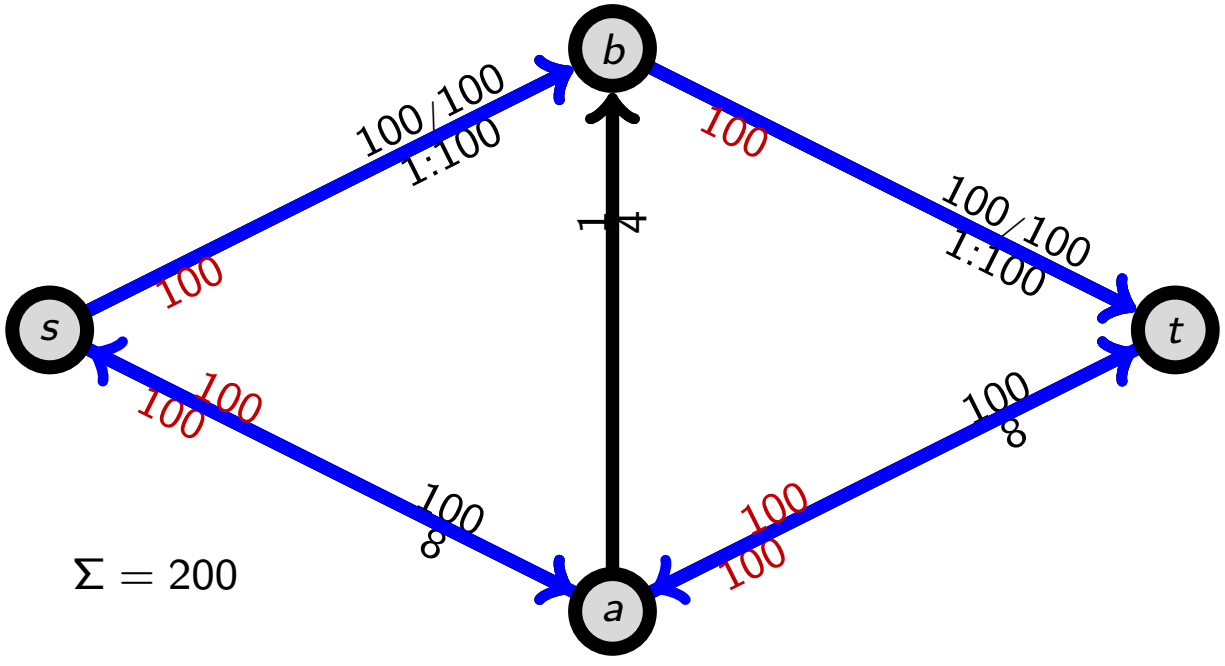
Bei schlecht gewählten Kreisen kann die Laufzeit sehr lang werden.



Hier geben wir auf.

Nochmal Beispiel zur Laufzeit ($W = 100$)

Bei gut gewählten Kreisen kann die Laufzeit ggf. besser sein.



Suche nach guten verbessernden Kreise

Das vorhegehende Beispiel hat gezeigt, daß bei unstrukturierter Suche Kreise gefunden werden können, die nur eine kleine Verbesserung erzeilen. Daher sollten nun Kreise gefunden werden, die eine möglichst grosse Verbesserung der Kosten erreichen.

Man könnte also einen Kreis mit minimalen Kosten suchen. Dieser könnte dann aber auch viele Kanten, beinhalten. In den bisherigen Beweisen haben wir gesehen, daß die Laufzeit dadurch klein wurde, indem wir die Nutzungshäufigkeit der Kanten zu minimieren versuchen.

Daher erscheint es sinnvoll, Kanten mit hoher Kostenverbesserung zu bevorzugen. Nun müssen wir aber gleichzeitig nach kostenverbessernden Kreisen suchen. Zusammengefasst heißt das: Suche nach einem Kreis, wo die durchschnittlichen Kantenkosten minimal sind (d.h. die durchschnittliche Kostenverbesserung pro Kante ist maximal).

Laufzeit und Verbesserung selbiger

- Der bisherige Algorithmus hat in obiger Form eine pseudopolynomielle Laufzeit.
- Dies kann aber verbessert werden:
  - Wähle kostengünstige Kreise.
  - Wähle Kreise über kostengünstige Kanten.
  - Also unabhängig von der Kreislänge.
- Dazu werden die Kreise gewählt, die die durchschnittlichen Kantenkosten minimieren.
- Setze:
$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$
- Setze weiter: $\mu(f) = -\bar{l}(C)$.
- Falls C negative Kosten hat, so ist $\mu(C)$ positiv.
- $\mu(f)$ gibt also die Verbesserung eines Kreisflusses an.

Mean-Algorithmus (Min-Cost-Flow)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$
$$\mu(f) = -\bar{l}(C)$$

1. Eingabe: $G = (V, E, s, t, c, l), W$.
2. Bestimme Fluss f mit $w(f) = W$.
3. Solange es in G_f einen negativen Kreis C gibt,
   1. Wähle C mit $\bar{l}(C)$ minimal.
      1. Bestimme maximalen zyklischen Fluss f' auf C.
      2. $f = f + f'(C)$.

- Es gilt: $w(f + f'(C)) = w(f) + w(f'(C)) = w(f) = W$.
- Es gilt: $l(f + f'(C)) = l(f) + l(f'(C)) < l(f)$.

Überblick

Es sind im Folgenden noch drei Probleme zu lösen:

- Wie wird eigentlich ein Min-Mean-Kreis bestimmt?
- Wie viel Laufzeit ist notwendig einen Min-Mean-Kreis zu bestimmen?
- Wie oft ist es notwendig, so einen Min-Mean-Kreis zu bestimmen?

Wir wählen das folgende Vorgehen:

Zuerst bestimmen wir, wie oft eine solcher Min-Mean-Kreis bestimmt wird. Dazu zeigen wird, daß nachdem n mal ein Min-Mean-Kreis bestimmt wurde, die Kosten um einen Faktor von $1 - 1/n$ gesunken sein müssen. Daher die Kosten eines Min-Mean-Kreises mindestens $1/n$ sind, werden also höchstens $m \cdot \log(nL)$ Berechnungen eines Min-Mean-Kreis gemacht.

Das Bestimmen eines Min-Mean-Kreis erfolgt in zwei Schritten: Mittels Dynamischer Programmierung wird der Wert des Min-Mean-Kreis berechnet. Mit Hilfe einer Potentialfunktion werden die Kosten der Kanten eines Min-Mean-Kreis zu Null.

Überblick zum Beweis

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$
$$\mu(f) = -\bar{l}(C)$$
$$1 - x \leqslant e^{-x}, x = 1/n$$

- Teile Iterationen in Phasen auf.
- Bestimme die Anzahl der Phasen.
- Bestimme die Anzahl der Iterationen pro Phase.
  - Zeige dies unter Verwendung einer besonderen Annahme.
  - Verändere l so, dass Annahme immer gilt und Algorithmus analog vorgeht.
  - Bestimme dazu Potential der Knoten, und addiere dies zu den Kantenkosten.
- Bestimme Laufzeit für eine Iteration.
- Zeige, wie man einfach einen Min-Mean-Kreis C findet (Dynamisches Programmieren).
  - Bestimme dazu vorab den Wert von C.
  - Dann wird Suche nach C einfach (passe Kosten an).
- Beachte für verbessernde Kreise C gilt: $\bar{l}(C) \leqslant -1/n$.

Idee zur Anzahl der Berechnungen eines Min-Mean-Kreises

Ziel ist es zu zeigen, daß höchstens $m \cdot \log(nL)$ Berechnungen eines Min-Mean-Kreis gemacht werden. Dazu teilen wir die Berechnungen in Phasen ein. Die erste Phase endet, wenn die Kosten des Flusses um einen Faktor von $1 - 1/n$ gesunken sind. Analog endet jede weitere Phase, wenn wieder die Kosten des Flusses um einen Faktor von $1 - 1/n$ gesunken sind.

Damit kann dann direkt gezeigt werden, daß die Anzahl der Phasen durch $n \ln(nL) + 1$ beschänkt ist.

In einem zweiten Schritt wird gezeigt, daß in einer Phase höchstens m Berechnungen eines Min-Mean-Kreis gemacht werden. Dazu nehmen wir erst einmal an, daß – alle Kanten des Graphen als Einzelkante betrachtet – keine Kostenverbesserung haben, die besser ist als die des Min-Mean-Kreises. Da in dieser Situation jeweils mindestens eine Kante entfernt (und in Gegenrichtung ohne Kostengewinn eingesetzt) wird. Wir werden im Folgenden mittels einer einfachen Rechnung sehen, daß damit die Phasenlängen durch m beschränkt sind.

Im letzten Teil wird mittels Potentialfunktion die Eingabe transformiert, so daß die oben dargestellte Annahme weiter gilt. Dabei werden für die Kosten für die Kreise nicht verändert. Damit ändert sich auch nicht das Verhalten des Algorithmuses.

Laufzeit (Aufteilung in Phasen)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$
$$\mu(f) = -\bar{l}(C)$$
$$1 - x \leqslant e^{-x}, x = 1/n$$

- Sei f der Fluss zu Beginn der i-ten Phase.
- Die Phase i endet, falls ein Fluss g gefunden wird mit:

$$\mu(g) \leqslant (1 - 1/n) \cdot \mu(f) \text{ oder } \mu(g) \leqslant 0.$$

- Falls $\mu(g) \leqslant 0$ terminiert der Algorithmus.
- Sei μ_0 der Wert von μ zu Beginn der ersten Phase.
- Sei μ_i der Wert von μ am Ende der i-ten Phase ($1 \leqslant i \leqslant T$).
- Damit gilt:

$$\mu_i \leqslant (1 - 1/n) \cdot \mu_{i-1} \leqslant \frac{\mu_{i-1}}{e^{1/n}}.$$

- Weiter gilt: $\mu_0 \leqslant L = \sum_{e \in E} |l(e)|$
- Wegen der Ganzzahligkeit gilt: $\mu_{T-1} \geqslant 1/n$.
- Es folgt:

$$T - 1 \leqslant \log_{e^{1/n}}(nL) = \frac{\ln(nL)}{\ln(e^{1/n})} = n \ln(nL)$$

- Und: $T \leqslant n \ln(nL) + 1$.

Laufzeit (Iterationen pro Phase)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$
$$\mu(f) = -\bar{l}(C)$$

- Zeige im Folgenden:
  - Die Phase (und der Algorithmus) terminiert nach spätestens m Iterationen, oder
  - Die nächste Phase startet nach spätestens $m-1$ Iterationen.
    - D.h. war der initiale Fluss f zu Beginn der Phase,
    - dann ist ein Fluss g nach spätestens $m-1$ Iterationen erreicht mit:
    - $\mu(g) \leqslant (1 - 1/n) \cdot \mu(f)$.
- Vorgehen:
  - Zeige Behauptung unter der Annahme: $\forall e \in E(G_f) : l(e) \geqslant -\mu(f)$.
  - Zeige Behauptung: Verändere dann l, so dass die Annahme immer gilt.

Laufzeit (Iterationen pro Phase)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$

$$\mu(f) = -\bar{l}(C)$$

$$\forall e \in E(G_f) : l(e) \geqslant -\mu(f)$$

- Typ 1 Iteration: Kreis C enthält nur Kanten mit negativen Kosten.
- Typ 2 Iteration: Kreis C enthält mindestens eine Kante mit positiven Kosten.
- Bei jeder Typ 1 Iteration wird mindestens eine Kante saturiert und entfernt.
- Alle dabei neu entstehenden Kanten haben positive Kosten (andere Richtung).
- Nach spätestens m konsekutiven Typ 1 Iterationen terminiert das Verfahren.
- Wenn also die Phase mehr als m Iterationen hat, folgt nach spätestens $m - 1$ Iterationen eine Typ 2 Iteration.
- Wir zeigen nun, dass dieser Fall unter der Annahme nicht auftritt.

Laufzeit mit Annahmen (Iterationen pro Phase)

$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$, $\mu(f) = -\bar{l}(C)$, **Annahme:** $\forall e \in E(G_f) : l(e) \geqslant -\mu(f)$

- Sei g der Fluss vor der ersten Typ 2 Iteration.
- Die Annahme gilt weiter (keine neuen Kanten mit negativen Kosten):

$$\forall e \in E(G_g) : l(e) \geqslant -\mu(f)$$

- Sei C der Min-Mean-Kreis in G_g und H die Kanten mit negativen Kosten in C.
- Damit gilt:

$$\mu(g) = \sum_{e \in C} \frac{-l(e)}{|C|} \leqslant \sum_{e \in H} \frac{-l(e)}{|C|} \leqslant |H| \cdot \frac{\mu(f)}{|C|}$$

- Wegen $|H| \leqslant |C| - 1$ folgt:

$$|H|/|C| \leqslant 1 - 1/|C| \leqslant 1 - 1/n$$

- Damit: $\mu(g) \leqslant (1 - 1/n) \cdot \mu(f)$.
- Widerspruch: sind schon am Ende der Phase.
- Also gibt es in der Phase keine Typ 2 Iterationen.
- Falls die Annahme gilt, so endet die Phase nach $m - 1$ Typ 1 Iterationen.

Potentialfunktion

Eine Potentialfunktion bestimmt das Potential eines Knotens. D.h. wie "gut" ist der Knoten für die Lösung des Problems Das ist natürlich vom Problem und der Anwendung abhängig. Wir werden beim Matching Problem auch noch eine weitere Anwendung einer Potentialfunktion sehen.

Hier wird für einen Knoten v als Potential gewählt: die Kosten eines minimalen Kantenzuges, der in v endet. Für einen Knoten des Min-Mean-Kreis ist damit dieser Kreis ein wichtiger Teil seines Potentials.

Ein Knoten v addiert sein Potential auf die Kosten jeder seiner ausgehenden Kanten und subtrahiert die Kosten jeder seiner eingehenden Kanten. Damit bleiben die Kosten auf Kreisen unverändert.

Weiterhin sieht man, das diese Potentialfunktion mittels dynamischer Programmierung effizient bestimmt werden kann.

Analog werden wir im Folgenden sehen, daß dann auch der Wert des Min-Mean-Kreis mittels dynamischer Programmierung, unter Verwendung der Zwischenergebnisse von der Berechnung der Knotenpotentale, effizient bestimmt werden kann.

Laufzeit (Erzwinge die Annahme)

$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$, $\mu(f) = -\bar{l}(C)$, **Annahme:** $\forall e \in E(G_f) : l(e) \geqslant -\mu(f)$

- Wir sorgen dafür, dass die Annahme erfüllt werden kann:
- Wir verändern l geeignet.
- Verhalten des Algorithmus sollte unverändert sein.
- Sei $p : V \mapsto \mathbb{Z}$ ein Potential für die Knoten.
- Setze für alle $e = (v, w) \in E(G_f)$ setze: $l'(e) = l(e) + p(v) - p(w)$.
- Für $\overline{e} = (w, v)$ gilt:

$$\begin{aligned} l'(\overline{e}) &= l(\overline{e}) + p(w) - p(v) \\ & \quad -l(e) + p(w) - p(v) = -l'(e) \end{aligned}$$

- Potentiale ändern die Kostensumme auf Kreisen C nicht: $l(C) = l'(C)$.
- Potentiale ändern damit auch nicht den Ablauf des Verfahrens.
- Wir zeigen nun, dass es Potentiale gibt, die die Annahme erzwingen.

Laufzeit (Erzwinge die Annahme durch Potential)

$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$, $\mu(f) = -\bar{l}(C)$, **Annahme:** $\forall e \in E(G_f) : l(e) \geqslant -\mu(f)$

Lemma (Existenz von p)

In $G_f = (V, E_f)$ gelte $\bar{l}(C) \geqslant -\mu$ für jeden Kreis C.
Dann gibt es $p : V \mapsto \mathbb{Z}$ mit:
$l'(e) = l(e) + p(w) - p(v) \geqslant -\mu$ für jede Kante $e \in E_f$.

Beweis:

- Für $e \in E_f$ setze: $l_\mu(e) = l(e) + \mu$
- Für $v \in V$ bestimme Kantenzug P in G_f von beliebigen Knoten w zu v mit minimalen Gewicht. Setze dann $p(v) = l_\mu(P)$.
- Beachte: Das ist wohldefinert, denn aus $\bar{l}(C) \geqslant -\mu$ folgt $l_\mu(C) \geqslant 0$ für beliebigen Kreis C.
- Damit gilt für jede Kante $e = (v, w) \in E_f : p(w) \leqslant p(v) + l_\mu(e)$.
- Es folgt:

$$\begin{aligned} l'(e) &= l(e) + p(v) - p(w) \\ &= l_\mu(e) + p(v) - p(w) - \mu \geqslant -\mu \end{aligned}$$

Laufzeit pro Iteration

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}, \quad \mu(f) = -\bar{l}(C), \quad \forall e \in E(G_f) : l(e) \geqslant -\mu(f)$$

Lemma (Karp, 1978)

Ein Min-Mean-Cycle kann in G_f in Zeit $O(nm)$ gefunden werden.

Beweis:

- Setze für $v \in V$ und $k \in \{0, \ldots, n\}$:
  $d_k(v)$ seien die Kosten des günstigsten Kantenzugs nach v mit genau k Kanten.
- Es gilt $d_0(v) = 0$ und

$$d_{k+1}(v) = \min_{e=(w,v) \in E_f} (d_k(w) + l(e)).$$

- Alle $d_k(v)$ Werte können durch dynamische Programmierung in Zeit $O(nm)$ bestimmt werden.

Wert des Min-Mean-Kreises

Um den Wert des Min-Mean-Kreises zu bestimmen, können wir die bereits berechneten Werte $d_i(v)$ nutzen. Zur Erinnierung, $d_i(v)$ sind die Kosten eines Kantenzugs der Länge i zum Knoten v

Nun können diese Werte nicht direkt zum Bestimmen des Wertes des Min-Mean-Kreises genutzt werden. Wenn wir aber i ausreichend gross (z.B. $i = n$) wählen, dann wird der Min-Mean-Kreises auf dem minimalen Kantenzug zu v auftreten.

Nehmen wir mal an: v ist ein Knoten des Min-Mean-Kreises. Unser Kantenzug P mit dem Wert $d_i(v)$ betritt irgendwann den Min-Mean-Kreises. Also müssen wir den initialen Anteil von P, also ein Pfad maximaler Länge, abziehen. Für ein passendes j ist der Wert des Min-Mean-Kreises $\frac{d_n(v)-d_j(v)}{n-j}$.

Es verbleibt das Minimieren über alle Knoten und das folgende Maximieren über alle j:

$$\min_{v\in V} \max_{j=0}^{n-1} \left(\frac{d_n(v) - d_j(v)}{n-j} \right).$$

Im Folgenden werden wir zeigen, daß diese Formel wirklich den Wert des Min-Mean-Kreises bestimmt und damit auch wieder per dynamischer Programmierung bestimmt werden kann.

Wert des Min-Mean-Kreises

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}, \quad \mu(f) = -\bar{l}(C)$$

Lemma

Der Wert $\bar{l}(C)$ des Min-Mean-Kreises ist:

$$\alpha = \min_{v \in V} \max_{j=0}^{n-1} \left(\frac{d_n(v) - d_j(v)}{n - j} \right)$$

Beweis:

- Sei C ein Min-Mean-Kreis.
- Wenn alle Kantenkosten um δ erhöht werden, bleibt C Min-Mean-Kreis.
- Der Wert von C erhöht sich auch um δ.
- Also können wir alle Kantenkosten um δ verschieben bis $\bar{l}(C) = 0$ gilt.
- Zeige nun: $\alpha = 0$ gilt damit auch.

Wert des Min-Mean-Kreises (Zeige $\alpha \geqslant 0$)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$

$$\mu(f) = -\bar{l}(C)$$

$$\alpha = \min_{v \in V} \max_{j=0}^{n-1} \left(\frac{d_n(v) - d_j(v)}{n-j} \right)$$

- Sei v beliebig und P Kantenzug, der bei v endet und Kosten $d_n(v)$ hat.
- P muss Kreis C beinhalten.
- Sei P' der verbleibende Kantenzug mit j Kanten ohne C.
- Dann hat C $n - j$ Kanten.
- Wegen $l(C) \geqslant 0$ gilt:

$$d_n(v) = l(P) = l(C) + l(P') \geqslant l(P') \geqslant d_j(v).$$

- Für jeden Knoten v gibt es $j \in \{1, \ldots, n-1\}$ mit $d_n(v) \geqslant d_j(v)$.
- Damit $\alpha \geqslant 0$.
- Im Folgenden zeigen wir nun noch $\alpha \leqslant 0$.

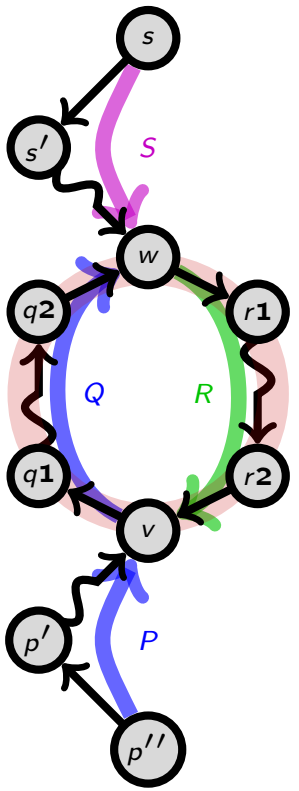
Wert des Min-Mean-Kreises (Zeige $\alpha \leqslant 0$)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$

$$\mu(f) = -\bar{l}(C)$$

$$\alpha = \min_{v \in V} \max_{j=0}^{n-1} \left(\frac{d_n(v) - d_j(v)}{n-j} \right)$$

- Zeige: Es gibt Knoten w mit: $d_n(w) \leqslant d_j(w)$ für alle $j \in \{0, \ldots, n-1\}$
- Sei v Knoten des Min-Mean-Kreises C.
- Sei P kürzester Kantenzug, der bei v endet. O.B.d.A. enthält P keinen Kreis. Sei $p < n$ die Anzahl der Kanten in P
- Sei w der Knoten, der auf C von v aus nach $n - p$ Kanten liegt. Diesen Kantenzug nennen wir Q.
- Sei R der Weg von w zu v auf C mit r Kanten.
- Sei $j \in \{0, \ldots, n-1\}$ und S ein Kantenzug minimaler Länge aus j Kanten, der an w endet.



Wert des Min-Mean-Kreises (Zeige $\alpha \leqslant 0$)

$$\bar{l}(C) = \frac{l(C)}{|C|} = \frac{\sum_{e \in C} l(e)}{|C|}$$

$$\mu(f) = -\bar{l}(C)$$

$$\alpha = \min_{v \in V} \max_{j=0}^{n-1} \left(\frac{d_n(v) - d_j(v)}{n-j} \right)$$

- Dann gilt:

$$d_n(w) \leqslant l(P) + l(Q) = d_p(v) + l(Q).$$

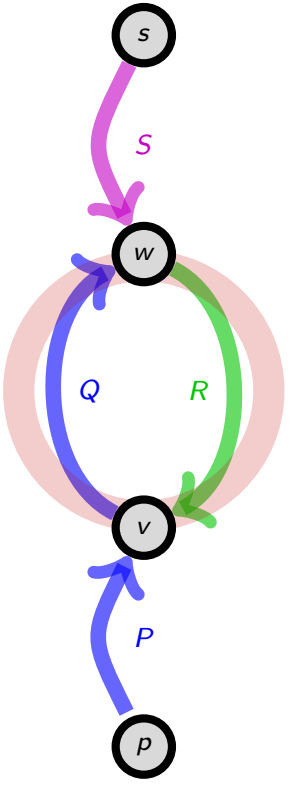
- und

$$d_p(v) \leqslant d_{j+r}(v) \leqslant d_j(w) + l(R).$$

- es folgt:

$$d_n(w) \leqslant d_j(w) + l(R) + l(Q).$$

- Der Kantenzug $Q \odot R$ hat Kosten 0 (wegen $l(C) = 0$).
- Damit gilt: $d_n(w) \leqslant d_j(w)$.



Gesamtverfahren

1. Berechne die $d_k(v)$ Werte und bestimme α.
2. Addiere zu allen Kantenkosten um α.
   - Damit ist der Wert des Min-Mean-Kreises 0.
3. Transformiere die Kantenkosten um den Wert α wie in Lemma "Existenz von p" beschrieben.
   - Die Potentiale ergeben sich aus den Werten $d_k(v)$.
   - Alle Kantenkosten sind nun nicht negativ.
   - Die Kanten des Min-Mean-Cycle haben Wert 0.
4. Lösche alle Kanten mit Wert größer 0.
5. Suche mit Tiefensuche Kreis in den verbleibenden Kanten.

Gesamtlaufzeit

Theorem

Der Mean-Cycle-Algorithmus hat eine Laufzeit von $O(m^2 \cdot n^2 \cdot \log(nL))$.

Beweis, siehe obige Überlegungen:

- $n \log(nL) + 1$ Phasen (L Maximum der absoluten Kantenkosten).
- m Iterationen pro Phase.
- $O(nm)$ Laufzeit pro Iteration.

Theorem

Der Mean-Cycle-Algorithmus hat eine Laufzeit von $O(m^3 \cdot n^2 \cdot \log n)$.

Literatur

- Ahuja, Magnanti, Orlin: Network Flows: Theory, Algorihms, and Applications, Prentice Hall, 1993.
- Cormen, Leiserson, Rives: Introduction to Algorithms, First Edition, MIT Press, 1990.
- Cormen, Leiserson, Rives: Introduction to Algorithms, Second Edition, MIT Press, 2001.
- Ottmann, Widmayer: Algorithmen und Datenstrukturen. BI-Wiss.-Verl. 1990.

Fragen(Flüsse)

- Was ist die Laufzeit der Ford-Fulkerson Methode?
- Wann hat die Ford-Fulkerson Methode die maximale Laufzeit?
- Warum liefert die Ford-Fulkerson Methode den maximalen Fluss?
- Was ist die Laufzeit der Ford-Fulkerson Methode mit Breitensuche?
- Wie ist die Idee des Min-Cut-Max-Flow Theorems?
- Wie wird der Schnitt zu dem maximalen Fluss gefunden?
- Was ist die Idee des Algorithmus von Dinitz?
- Wie funktioniert die Forward-Propagation?
- Wie ist die Laufzeit vom Algorithmus von Dinitz?
- Wie ist die Begründung zur Laufzeit vom Algorithmus von Dinitz?

Fragen(Flüsse)

- Wie bestimmt man Flüsse mit einem Mindestfluss auf den Kanten?
- Wie bestimmt man Flüsse mit einem Mindestfluss auf den Kanten, wo aber auch ein leerer Fluss erlaubt ist?
- Wie ist die Idee der Algorithmen zu kostenminimalen Flüssen?
- Was ist die Laufzeit der Algorithmen zu kostenminimalen Flüssen?
- Gebe die Idee zum Beweis der Laufzeit der Algorithmen zu kostenminimalen Flüssen?

Legende

n : Nicht relevant

g : Grundlagen, die implizit genutzt werden

i : Idee des Beweises oder des Vorgehens

s : Struktur des Beweises oder des Vorgehens

w : Vollständiges Wissen