Algorithmentechnik — Vorlesung am 15. Dez. 2005

http://i11www.ira.uka.de/teaching/WS_0506/algotech

Steffen Mecke (mecke@ira.uka.de)

WS 05/06

Goldberg/Tarjan

- Definitionen
- Der Algorithmus
- Korrektheit
- Laufzeit
- Mehrmaschinenscheduling

Äquivalente Formulierung

- Ein Fluss ist eine Abbildung $f : V \times V \to \mathbb{R}$ mit
  - *Kapazitätsbedingung*

$$\forall (v,w) \in V \times V \quad f(v,w) \leq c'(v,w) \ , \tag{1}$$

  - *Antisymmetrie-Forderung*

$$\forall (v,w) \in V \times V \quad f(v,w) = -f(w,v) \quad \text{und} \tag{2}$$

  - *Flußerhaltungsbedingung*

$$\forall v \in V \setminus \{s,t\} \quad \sum_{u \in V} f(u,v) = 0 \ . \tag{3}$$

Zur Vereinfachung wird c statt c' verwendet.

Präfluss

Definition

Ein *Präfluss* ist eine Abbildung $f : V \times V \to \mathbb{R}$, welche die Kapazitätsbedingung und die Antisymmetrie-Forderung erfüllt sowie

$$\forall v \in V \setminus \{s\} \quad \sum_{u \in V} f(u, v) \geq 0 \ . \tag{4}$$

Die Bedingung besagt, dass für alle Knoten $v \in V \setminus \{s\}$ mindestens soviel Fluss hineinfließt wie auch hinausfließt.

Definition

Sei f ein Präfluss. Für $v \in V$ heißt der Wert

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

Flussüberschuss, und die Abbildung $r_f : E' \to \mathbb{R}$ mit

$$\forall (u, v) \in E' \quad r_f(u, v) := c(u, v) - f(u, v)$$

heißt *Restkapazität*.

Definition

Eine Kante $(v, w) \in E'$ heißt *Residualkante* bezüglich Präfluss f, falls $r_f(v, w) > 0$. Der *Residualgraph* zu f ist gegeben durch $D_f(V, E_f)$ mit

$$E_f := \{(v, w) \in E' |\ r_f(v, w) > 0\}.$$

Bemerkung: Auf Residualkanten kann der Fluss erhöht werden. Das bedeutet dass entweder der „echte" Fluss entweder erhöht wird (falls $0 < f(u, v) \leq c(u, v)$) oder Fluss „zurückgeschoben" wird (falls $f(v, u) = -f(u, v) < 0 \leq c(v, u)$).

Definition

- Eine Abbildung $\text{dist}\colon V \to \mathbb{N}_0 \cup \{\infty\}$ heißt *zulässige Markierung* bezüglich eines Präflusses f, falls gilt:

$$\text{dist}(s) = |V|, \quad \text{dist}(t) = 0$$

  und für alle $v \in V \setminus \{t\}$, falls $(v, w) \in E_f$,

$$\text{dist}(v) \leq \text{dist}(w) + 1 \enspace .$$

- Ein Knoten $v \in V$ heißt *aktiv* im Laufe des Algorithmus, wenn

$$v \in V \setminus \{s, t\}, \; e(v) > 0 \text{ und } \text{dist}(v) < \infty.$$

Invarianten des Algorithmus

- f ist stets ein Präfluss.
- dist ist stets eine zulässige Markierung (insbes. $\text{dist}(s) = |V|$).
- Falls $dist(v) < |V|$, so ist $\text{dist}(v)$ untere Schranke für den Abstand von v zu t in D_f.
- Falls $dist(v) > |V|$, so ist $\text{dist}(v) - |V|$ untere Schranke für den Abstand von v zu s in D_f und t von v in D_f nicht erreichbar.

Der Algorithmus

1. Für $(v, w) \in V \times V$ setze $f(v, w) := 0$ und $r_f(v, w) := c(v, w)$.
2. Setze $\text{dist}(s) := |V|$.
3. Für $v \in V \setminus \{s\}$ setze:
   3.1 $\text{dist}(v) := 0$.
   3.2 $f(s, v) := c(s, v)$, $r_f(s, v) := 0$.
   3.3 $f(v, s) := -c(s, v)$, $r_f(v, s) := c(v, s) - f(v, s)$.
   3.4 $e(v) := c(s, v)$.
4. Solange es einen aktiven Knoten gibt:
   4.1 Wähle einen aktiven Knoten v aus.
   4.2 Führe für v eine zulässige Operation Push oder Relabel aus.

Push und Relabel

- Push(v, w) ist zulässig, falls v aktiv ist und $r_f(v, w) > 0$ und $\mathrm{dist}(v) = \mathrm{dist}(w) + 1$ gilt
  1. $\Delta := \min(e(v), r_f(v, w))$.
  2. $f(v, w) := f(v, w) + \Delta$, $f(w, v) := f(w, v) - \Delta$.
  3. $r_f(v, w) := r_f(v, w) - \Delta$, $r_f(w, v) := r_f(w, v) + \Delta$.
  4. $e(v) := e(v) - \Delta$, $e(w) := e(w) + \Delta$.

- Relabel(v) ist zulässig, falls v aktiv ist und für alle w mit $r_f(v, w) > 0$ gilt, dass $\mathrm{dist}(v) \leq \mathrm{dist}(w)$.

  $$\mathrm{dist}(v) := \begin{cases} \infty & \text{, falls } \{w \mid r_f(v, w) > 0\} = \emptyset \;, \\ \min\{\mathrm{dist}(w) + 1 \mid r_f(v, w) > 0\} & \text{sonst} \;. \end{cases}$$

Lemma

Während des Ablaufs des Algorithmus von Goldberg und Tarjan ist f stets ein Präfluss und dist stets eine bezüglich f zulässige Markierung.

Lemma

Sei f ein Präfluss auf D, die Funktion dist eine bezüglich f zulässige Markierung auf V und $v \in V$ ein aktiver Knoten. Dann ist entweder eine PUSH*-Operation von v oder eine* RELABEL*-Operation von v zulässig.*

Lemma

Sei f ein Präfluss und dist bezüglich f zulässig. Dann ist t im Residualgraph D_f von s aus nicht erreichbar (es gibt also keinen gerichteten s-t-Weg in D_f).

Satz

Falls der Algorithmus von Goldberg und Tarjan terminiert und am Ende alle Markierungen endlich sind, dann ist der konstruierte Präfluss ein Maximalfluss im Netzwerk $(D; s, t; c)$.

Lemma

Sei f ein Präfluss auf D. Wenn für v gilt, dass $e(v) > 0$, so ist s in D_f von v aus erreichbar.

Lemma

Während des gesamten Algorithmus gilt

$$\forall v \in V \quad dist(v) \leq 2|V| - 1 \enspace .$$

Lemma

Während des Algorithmus werden höchstens $2|V| - 1$ Operationen Relabel *pro Knoten ausgeführt. Die Gesamtzahl der* Relabel*-Operationen ist also höchstens $2|V|^2$.*

Definition

Eine Operation $\textsc{Push}(v, w)$ heißt *saturierend*, wenn hinterher $r_f(v, w) = 0$ gilt. Ansonsten heißt $\textsc{Push}(v, w)$ *nicht saturierend*.

Lemma

Während des Algorithmus werden höchstens $2|V|\,|E|$ *saturierende* $\textsc{Push}$ *ausgeführt.*

Lemma

Während des Algorithmus werden höchstens $4|V|^2|E|$ *nicht saturierende* $\textsc{Push}$ *ausgeführt.*

Satz

Der Algorithmus von Goldberg und Tarjan terminiert nach spätestens $O(|V|^2|E|)$ *Ausführungen zulässiger* $\textsc{Push}$*- oder* $\textsc{Relabel}$*-Operationen.*

Verbesserungen

Durch geschickte Auswahl der Push- und Relabel-Operationen kann bessere Laufzeit erreicht werden:

- **FIFO-Implementation**: Die aktiven Knoten werden entsprechend der Reihenfolge „first-in-first-out" gewählt. Dies führt zu $O(|V|^3)$ (Goldberg 1985, Shiloach und Vishkin 1982). Mit „Dynamischen Bäumen" sogar $O(|V|\,|E| \log \frac{|V|^2}{|E|})$ (Goldberg und Tarjan 1988).
- **Highest-Label-Implementation:** Für Push wird unter den aktiven Knoten derjenige mit höchstem Wert von dist gewählt. Dies führt zu $O(|V|^2|E|^{\frac{1}{2}})$ (Cheriyan und Motvekwani 1989).
- **Excess-Scaling-Implementation:** Für Push(v, w) wird die Kante (v, w) so gewählt, dass $e(v)$ „geeignet groß" und $e(w)$ „geeignet klein" ist. Dies führt zu $O(|E| + |V|^2 \log C)$, wobei $C := \max_{(u,v)} c(u, v)$ ist (Ahuja und Orlin 1989).

Anwendungsbeispiel: Mehrmaschinenscheduling

Gegeben:

- Eine endliche Menge von *Aufträgen* (*Jobs*) $j \in J$
- Für jede Aufgabe j eine Bearbeitungszeit $p_j \in \mathbb{R}_0^+$
- eine früheste Anfangszeit $r_j \in \mathbb{R}_0^+$ und
- eine Deadline $d_j \geq r_j + p_j$
- *M Maschinen*.

Jede Maschine kann zu einem Zeitpunkt nur einen Job bearbeiten, und jeder Job kann zu einem Zeitpunkt nur von einer Maschine bearbeitet werden. Jobs können allerdings unterbrochen werden und später auf derselben oder einer anderen Maschine weiterbearbeitet werden.

Anwendungsbeispiel: Mehrmaschinenscheduling(Forts.)

Gesucht: Eine Bearbeitungsreihenfolge der Jobs auf den Maschinen, die alle Bedingungen erfüllt, sofern eine solche existiert.

Vorgehen:

- Ordne die r_j und d_j für alle $j \in J$ in nichtabsteigender Reihenfolge.
- Bezeichne die höchstens $2|J| - 1$ paarweise disjunkten Zeitintervalle zwischen diesen Zeitpunkten mit T_{kl} für $[k, l)$ bezeichnet.

Umwandlung in Flussproblem

- $|J|$ Knoten für die Jobs.
- Je einen Knoten für jedes T_{kl}
- Quelle s, Senke t
- Kanten (s, j) mit $c(s, j) = p_j$
- Kanten (T_{kl}, t) mit $c(T_{kl}, t) = M(l - k)$
- Kanten (j, T_{kl}) mit $c(j, T_{kl}) = l - k$, falls $r_j \leq k$ und $d_j \geq l$

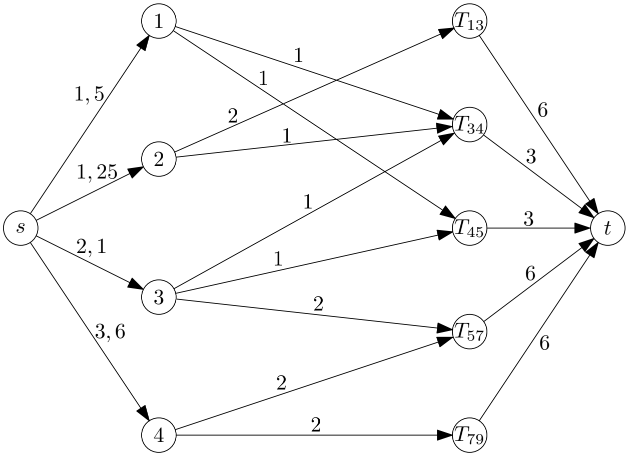
Eine zulässige Bearbeitungsreihenfolge existiert genau dann, wenn für einen Maximalfluss f in D gilt, dass $w(f) = \sum_{j \in J} p_j$.

Beispiel

Wir wählen $M := 3$, $J := \{1, 2, 3, 4\}$ und

$$
\begin{array}{lll}
p_1 := 1.5 \ , & r_1 := 3 \ , & d_1 := 5 \ , \\
p_2 := 1.25 \ , & r_2 := 1 \ , & d_2 := 4 \ , \\
p_3 := 2.1 \ , & r_3 := 3 \ , & d_3 := 7 \ , \\
p_4 := 3.6 \ , & r_4 := 5 \ , & d_4 := 9 \ .
\end{array}
$$

- Zeitpunkte $1, 3, 4, 5, 7, 9$
- Intervalle T_{13}, T_{34}, T_{45}, T_{57}, T_{79}.


1
2
3
4
s
t
T_{13}
T_{34}
T_{45}
T_{57}
T_{79}
1, 5
1, 25
2, 1
3, 6
1
1
2
1
1
1
2
2
2
6
3
3
6
6