Netzwerkalgorithmen

Sommersemester 2022

Stefan Näher
Universität Trier
naeher@uni-trier.de

Vorlesung 9

Maxflow: Der Preflow-Push Algorithmus

Preflow/Excess

1. Ein **Preflow** ist eine Funktion $x : E \longrightarrow \mathbb{R}_0^+$ mit
   a) $0 \leq x_{ij} \leq u_{ij}$
   b) $\sum_{(j,i)\in E} x_{ji} - \sum_{(i,k)\in E} x_{ik} \geq 0$

2. $e(i) = \sum_{(j,i)\in E} x_{ji} - \sum_{i,k)\in E} x_{ik}$
   heißt **Überschuss** oder Excess des Knotens i.

3. Ein Knoten $i \in V \setminus \{s, t\}$ mit $e(i) > 0$ ist **aktiv**.

Maxflow: Der Preflow-Push Algorithmus

Distanz-Funktion

Sei x ein Preflow und $G(x)$ das entsprechende Restnetzwerk.

Die Funktion $d : V \longrightarrow \mathbb{N}$ ist eine **Distanz-Funktion**, wenn
1. $d(t) = 0$
2. $d(i) \leq d(j) + 1$ für alle Kanten (i, j) in $G(x)$.

Eigenschaften
- $d(i)$ ist untere Schranke für exakte Distanz $dist(i, t)$
- $d = 0$ (Nullfunktion) ist eine gültige Distanzfunktion

Eine Kante (i, j) ist **admissable**, wenn
$d(i) = d(j) + 1$

Der generische Preflow-Push Algorithmus

// Initialisierung

1. **forall** $i \in V$ **do**
2. $\quad d(i) \leftarrow 0;$
3. $\quad e(i) \leftarrow 0;$
4. **od**

5. **forall** $(i, j) \in E$ **do**
6. $\quad x_{ij} \leftarrow 0;$
7. **od**

// saturiere alle aus s ausgehenden Kanten

5. **forall** $j \in V$ mit $(s, j) \in E$ **do**
6. $\quad x_{sj} \leftarrow u_{sj}$;
7. $\quad e(j) \leftarrow e(j) + x_{sj}$;
8. **od**
9. $d(s) \leftarrow n$; // und hebe s auf Level n

// Haupt-Schleife

9. **while** es existiert ein aktiver Knoten **do**
10. $\quad$ wähle einen solchen Knoten i;
11. $\quad$ PUSH/RELABEL(i)
12. **od**

PUSH/RELABEL

1. PUSH/RELABEL(i)
2. **if** i hat eine ausgehende admissible edge (i,j) in $G(x)$ **then** // push
3. wähle eine solche Kante (i,j);
4. $\delta \leftarrow \min\{\, e(i), r_{ij} \,\}$;
5. $x_{ij} \leftarrow x_{ij} + \delta$;
6. $e(i) \leftarrow e(i) - \delta$;
7. $e(j) \leftarrow e(j) + \delta$;
8. **else** // relabel
9. $d(i) \leftarrow \min\{\, d(j) \mid (i,j) \text{ in } G(x) \,\} + 1$;
10. **fi**

Wir unterscheiden 2 Arten von Push-Operationen

1. **Saturierende Push-Operationen**

$$e(i) >= r_{ij}$$

Der Excess im Knoten i ist mindestens so hoch wie die Restkapazität.
Nach dem Push ist die Kante saturiert ($r_{ij} = 0$).

2. **Nicht-saturierende Push-Operationen**

$$e(i) < r_{ij}$$

Der Excess reicht nicht aus, um die Kante zu saturieren.
Nach dem Push ist $e(i) = 0$ (d.h. i ist nicht mehr aktiv).

Lemma 1

Falls $e(i) > 0$ (d.h. i ist aktiv), dann exisitiert ein Pfad von i nach s in $G(x)$.

Beweis:

Übung

Lemma 2
Für alle Knoten $i \in V$ gilt $d(i) < 2n$.

Beweis:
Wenn ein Knoten i relabeled wird, gilt $e(i) > 0$.
Nach Lemma 1 existiert dann ein Pfad von i nach s in $G(x)$.
Ein solcher Pfad P hat maximal Länge $n - 1$ (das gilt für jeden Pfad).

Dann gilt:
$d(s) = n$ und $d(u) \leq d(v) + 1$ für jede Kante (u, v) auf dem Pfad P.

Daraus folgt: $d(i) \leq d(s) + |P| < 2n$.

Lemma 3
Für jeden einzelnen Knoten i werden maximal $2n$ *Relabel*-Operationen durchgeführt.

Beweis:
Bei jeder *Relabel*-Operation für i erhöht sich $d(i)$ um mindestens 1.
Dann folgt aus Lemma 2, dass jedes Dist-Label $d(i)$ höchstens $2n$ mal erhöht wird, d.h. jeder einzelne Knoten i wird maximal $2n$ mal relabeled.

Lemma 4

Es werden insgesamt maximal $2n^2$ *Relabel*-Operationen durchgeführt.

Beweis:

Nach Lemma 2 wird jeder Knoten maximal $2n$ mal relabeled.

Also ist die Gesamtzahl aller Relabel-Operationen höchstens $2n^2$.

Lemma 5
Es werden maximal nm *saturierende* Push-Operationen durchgeführt.

Beweis:
Zwischen jeweils 2 aufeinanderfolgenden *saturierenden* Push-Operationen über eine Kante (i, j) muss das Distanz-Label $d(i)$ des Knotens i um mindestens 2 erhöht worden sein.
Warum ? $\longrightarrow$ **Übung**

Da (nach Lemma 2) $d(i) < 2n$ kann jede einzelne Kante (i, j) maximal n mal saturiert werden.

Daraus folgt das Lemma.

Lemma 6
Es werden maximal $\mathcal{O}(n^2 m)$ *nicht-saturierende* Push-Operationen durchgeführt.

Beweis:
Wir verwenden eine Potentialfunktion Φ.
Sei I die Menge der aktiven Knoten d.h. $e(i) > 0$ für alle $i \in I$.
Dann definiere $\Phi = \sum_{i \in I} d(i)$ (Summe aller Distanz-Labels von aktiven Knoten).

Da $|I| \leq n$ und $d(i) < 2n$ gilt immer: $\Phi \leq 2n^2$.

1. Am Anfang gilt: $\Phi \leq 2n^2$

2. Bei Termination des Algorithmus gilt $\Phi = 0$, da dann $I = \emptyset$.

Bei einer PUSH/RELABEL(i) Operation kann man 2 Fälle unterscheiden:

Fall 1:
Es gibt keine aus i ausgehende *admissible* Edge in $G(x)$

Dann wird das Distanz-Label $d(i)$ um ein $\epsilon \geq 1$ erhöht.

Diese Operation erhöht also Φ um höchstens ϵ.
wieso höchstens ?.

Insgesamt kann das Distanz-Label eines Knotens um maximal $2n$ erhöht werden. Damit beträgt die maximale Erhöhung von Φ aufgrund von Relabel-Operationen höchstens sn^2.

Fall 2:
Es gibt eine aus i ausgehende *admissible* Edge in $G(x)$.

Der Algorithmus führt also eine PUSH-Operation über diese Kante (i, j) aus. Diese kann saturierend oder nicht-saturierend sein.

Fall 2.1: Eine saturierende PUSH-Operation
könnte die Anzahl der aktiven Knoten um 1 erhöhen (warum ?);

In diesem Fall erhöht sich das Potential Φ um $d(j) \leq 2n$. Also ist der Gesamtbeitrag aller saturierenden Push-Operationen zu Φ maximal $2n^2m$, da insgesamt höchstens nm saturierende Pushes ausgeführt werden (Lemma 5).

Fall 2.1: Eine nicht-saturierende PUSH-Operation
Behauptung: Eine solche Operation vermindert Φ um mindestens 1.

Beweis:
Nach einem nicht-saturierenden Push ist i nicht mehr aktiv (warum ?) Dies vermindert Φ um $d(i)$.

Falls j vor der Operation nicht aktiv war (d.h. $e(j) = 0$) wird Φ gleichzeitig um $d(j) = d(i) - 1$ ((i, j) admissible) erhöht.

Damit vermindert also eine nicht-saturierende Push-Operation das Potential Φ um mindestens 1.

Zusammenfassung

1. Der Anfangswert von Φ ist höchstens $2n^2$ und am Ende ist $\Phi = 0$.

3. Die maximal mögliche Erhöhung von Φ beträgt

$$2n^2 \text{ (Fall 1)} \; + \; 2n^2m \text{ (Fall 2.1)}$$

4. Jede nicht-saturierende Push-Operation (2.2) vermindert Φ um mindestens **1**.

Daraus folgt, dass der Algorithmus maximal

$$2n^2 + 2n^2 + 2n^2m = \mathcal{O}(n^2m)$$

nicht-saturierende Push-Operationen ausführen kann.

Satz:
Der generische Prefow-Push Algorithmus hat Laufzeit $\mathcal{O}(n^2 \cdot m)$

Beweis:
Die Laufzeit wird dominiert von den Kosten der nicht-saturierenden Push-Operationen, diese betragen $\mathcal{O}(n^2 \cdot m)$ (Lemma 6).

Bemerkungen:

1. Die Laufzeit ist **streng polynomiell**.

2. Dieser **generische** Algorithmus hat Spielräume für Variationen
   (Auswahl des aktiven Knotens und einer admissible Edge beim Push).

3. Dies führt zu effizienteren Versionen.
   $\rightarrow$ **nächste Vorlesung**