Netzwerkalgorithmen

Sommersemester 2025

Stefan Näher
Universität Trier
naeher@uni-trier.de

Vorlesung 10

Lemma 1
Falls $e(i) > 0$ (d.h. i ist aktiv), dann exisitiert ein Pfad von i nach s in $G(x)$.

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

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

Lemma 4
Es werden insgesamt maximal $2n^2$ *Relabel*-Operationen durchgeführt.

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

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 $2n^2$.

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**.
   Wichtig: Die Eigenschaften des generischen Algorithmus (Lemma 1-5) bleiben erhalten. Die Anzahl der nicht-saturierenden Pushes wird vermindert.

Der FIFO Preflow-Push Algorithmus

Verfeinerung des generischen ALgorithmus

1. **Knoten-Behandlung** (node examination)

a) Wähle einen aktiven Knoten i d.h. $e(i) > 0$

b) Führe eine Folge von Push-Operationen von i aus durch, bis
- entweder i inaktiv wird ($e(i) = 0$)
- oder i relabeled wird ($e(i) > 0$ und keine admissible Edge (i, j)).

2. **Verwalte die aktiven Knoten** in einer **Schlange** oder **FIFO**-Queue Q.

Der FIFO Preflow-Push Algorithmus

Dann kann der Grundalgorithmus wie folgt modifiziert werden.

1. Intialisierung: $\mathrm{Q} =$ alle aktiven Nachbarn von s.
2. Hauptschleife: **while** $\neg \mathrm{Q}.\mathrm{empty}()$ **do**
3. Wahl eines aktiven Knotens: $\mathrm{i} \leftarrow \mathrm{Q}.\mathrm{top}()$
4. Innere Schleife: Knoten-Behandlung von i
5. Aktivierung eines Knotens: $\mathrm{Q}.\mathrm{append}(\mathrm{i})$

Übungsaufgabe:
Formulieren Sie den kompletten Algorithmus in Pseudo-Code.

Analyse des FIFO Preflow-Push Algorithms

Teile den Ablauf in **Phasen** ein.

1. Phase: Behandlung aller Knoten, die nach der Initialisierung aktiv sind
2. Phase: Behandlung aller Knoten, die sich nach Phase 1 in Q befinden

i-te Phase: Behandlung aller Knoten, die sich nach Phase $i-1$ in Q befinden.

Behauptung: Der Algorithmus führt maximal $2n^2$ Phasen aus.

Analyse des FIFO Preflow-Push Algorithms

Beweis:
Verwende das Potential $\Phi = \max\{d(i) \mid e(i) > 0\}$

1. Nach der Initialisierung gilt: $\Phi = 0$.

2. Betrachte die mögliche Veränderung der Potentialfunktion Φ über eine komplette Phase.

Fall 1:
Der Algorithmus führt mindestens eine Relabel-Operation in dieser Phase aus. Sei δ die maximale Erhöhung eines Dist-Labels in der Phase. Dann kann sich Φ um δ erhöhen. Aus Lemma 4 folgt, dass sich Φ insgesamt (in allen Phasen) um maximal $2nm^2$ erhöhen kann.

Fall 2:
Der Algorithmus führt in der Phase keine Relabel-Operation aus. Dann wird der Excess von allen aktiven Knoten in Q zu Konten mit kleinerem dist-Label verschoben. Dadurch vermindert sich Φ um mindestens 1 warum ?

Insgesamt folgt, dass die Anzahl der Phasen nicht größer als $2n^2$ sein kann.

Laufzeit des FIFO-Algorithmus

Satz 2:
Der FIFO Preflow-Push-Algorithmus hat eine Laufzeit von $\mathcal{O}(n^3)$.

Beweis:

In einer Phase wird jeder Knoten maximal **einmal** behandelt (node examination) und jede Behandlung eines Knotens führt maximal **ein** nicht-saturierendes Push aus (evtl. am Ende).

Da es maximal $2n^2$ Phasen gibt (Behauptung) folgt eine obere Schranke von $n \cdot 2n^2 = \mathcal{O}(n^3)$ für die Anzahl aller nicht-saturierender Push Operationen.

Highest-Label Preflow-Push

Eine Weitere Verfeinerung des Algorithmus

Idee:

Wähle in jeder Iteration einen Knoten mit maximalem Dist-Label.

Analyse der Laufzeit

Satz 3: Der Highest-Label Preflow-Push Algorithmus hat Laufzeit von $\mathcal{O}(n^2\sqrt{m})$

Ohne Beweis:

Der Algorithmus führt maximal $\mathcal{O}(n^2\sqrt{m})$ nicht-saturierende Push-Operationen aus.

Highest-Label Preflow-Push

while $\neg \mathrm{Q}.\mathrm{empty}()$ **do**
  sei i ein aktiver Knoten in Q mit $\mathrm{d}(\mathrm{i})$ maximal
  Behandlung von i
od

Q ist also eine **Priority-Queue** der aktiven Knoten mit den Dist-Labels als Prioritäten.

Aus **Lemma 2** folgt: Die Dist-Labels (also Prioritäten) sind ganze Zahlen aus dem beschränkten Interval $\{ 0, \ldots, 2\mathrm{n} \}$

Übungsaufgabe:
Überlegen Sie sich eine effiziente Implementierung der Priority-Queue Q.