

FPTAS für das Restricted Shortest Path-Problem

Rasmus Diederichsen Sebastian Höffner

Universität Osnabrück

6. Dezember 2015



Inhalt

- ① Das Problem
- ② Exakte Lösung
 - Algorithmus
 - Laufzeit
 - Terminierung
 - Beispiel
- ③ Das FPTAS
 - Test für Grenzen von OPT
 - Laufzeit von des Tests
 - Verbesserte Grenzen von OPT
 - Algorithmus
 - Laufzeit des FPTAS

Problemstellung

Gegeben

- azyklischer Graph $G = (V, E)$
- $(u, v) \in E$ hat Gewicht c und Verzögerung t

Single Source Shortest Path

Berechne vom Startknoten aus alle nach Kosten kürzesten Wege zu allen anderen ▶ Dijkstra

All Pairs Shortest Path

Kürzeste Wege zwischen allen Knotenpaaren ▶ Floyd

Das Problem

Gegeben

- azyklischer Graph $G = (V, E)$
- $(u, v) \in E$ hat gewicht c und Verzögerung t

Restricted Shortest Path

Finde nach Kosten kürzesten Weg von a nach b mit Verzögerung $\leq T$. **NP**-schwer.

Exakte Lösung

Algorithmus

Dynamische Programmierung (ähnlich wie Knapsack). Kanten (i, j) mit $i < j$, da azyklisch.

Algorithmus

$$g_1(c) = 0, \text{ Für } c = 0, \dots, OPT,$$

$$g_j(0) = \infty, \text{ Für } j = 2, \dots, n,$$

$$g_j(c) = \min \left\{ g_j(c-1), \min_{k | c_{kj} \leq c} \{ g_k(c - c_{kj}) + t_{kj} \} \right\}$$

$$\text{Für } j = 2, \dots, n; \ c = 1, \dots, OPT$$

Exakte Lösung

Laufzeit

$$g_1(c) = 0, \text{ Für } c = 0, \dots, OPT,$$

$$g_j(0) = \infty, \text{ Für } j = 2, \dots, n,$$

$$g_j(c) = \min \left\{ g_j(c-1), \min_{k | c_{kj} \leq c} \{ g_k(c - c_{kj}) + t_{kj} \} \right\}$$

$$\text{Für } j = 2, \dots, n; c = 1, \dots, OPT$$

- $\mathcal{O}(OPT \cdot n \cdot \text{Aufwand pro } (c, j))$
 - ▶ Pro (c, j) evtl. alle Vorgänger betrachten
 - ▶ $\mathcal{O}(n^2 OPT) = \mathcal{O}(|E| OPT)$
- Pseudopolynomiell

Exakte Lösung

Terminierung

$$g_1(c) = 0, \text{ Für } c = 0, \dots, OPT,$$

$$g_j(0) = \infty, \text{ Für } j = 2, \dots, n,$$

$$g_j(c) = \min \left\{ g_j(c-1), \min_{k | c_{kj} \leq c} \{ g_k(c - c_{kj}) + t_{kj} \} \right\}$$

$$\text{Für } j = 2, \dots, n; \ c = 1, \dots, OPT$$

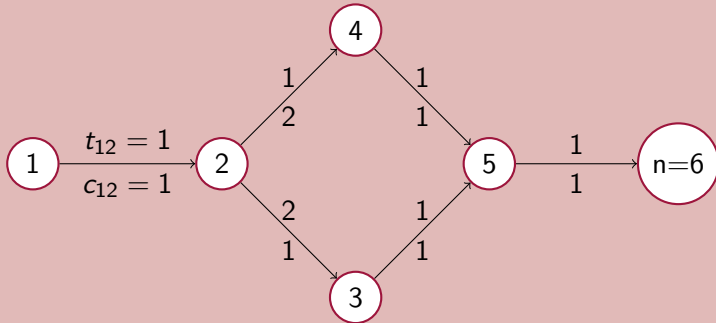
Man weiß $OPT = \min \{ c \mid g_n(c) \leq T \}$

- Setze OPT , sobald erstes c mit $g_n(c) \leq T$ gefunden.

Exakte Lösung

Beispiel

Autobahn? Oder doch lieber Landstraße?



Exakte Lösung

Beispiel

$j \backslash c$	0	1	2	3	4	5
1	0	0	0	0	0	0
2	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞

Exakte Lösung

Beispiel

$j \backslash c$	0	1	2	3	4	5
1	0	0	0	0	0	0
2	∞	1	∞	∞	∞	∞
3	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	∞

$$g_2(1) = \min \left\{ g_2(0), \min_{k | c_{kj} \leq c} \{ g_k(c - c_{kj}) + t_{kj} \} \right\}$$

$$g_2(1) = \min \{ \infty, \min \{ g_1(1 - 1) + 1 \} \}$$

$$g_2(1) = 1$$

Exakte Lösung

Beispiel

$j \backslash c$	0	1	2	3	4	5
1	0	0	0	0	0	0
2	∞	1	1	1	1	1
3	∞	∞	3	3	3	3
4	∞	∞	∞	2	2	2
5	∞	∞	∞	4	3	3
6	∞	∞	∞	∞	5	4

Das FPTAS

Test für Grenzen von OPT

Wir suchen zunächst ein Verfahren, dass untere und obere Schranken für OPT findet.

- Wunsch-dir-was: Polynomieller Algorithmus $TEST(k)$, sodass

$$TEST_{magic}(k) = \begin{cases} 1 & \text{falls } OPT \geq k \\ 0 & \text{falls } OPT < k \end{cases}$$

- ▶ Binäre Suche auf $0, \dots, UB$
- ▶ Leider **NP**-schwer

Das FPTAS

Test für Grenzen von OPT

$TEST_{magic}(k)$ kann nicht existieren, also schwächer:

Eigenschaften von $TEST(k)$

$$TEST(k) = \begin{cases} 1 & \text{falls } OPT \geq k \\ 0 & \text{falls } OPT < k(1 + \epsilon) \end{cases}$$

Das FPTAS

Test für Grenzen von OPT

$TEST_{magic}(k)$ kann nicht existieren, also schwächer:

$TEST(k)$

- Skaliere und runde Kantengewichte als $\hat{c}_{ij} = \left\lfloor \frac{c_{ij}(n-1)}{k\epsilon} \right\rfloor$
- Wende exakten Algorithmus an, bis $g_n(c) \leq T$ gefunden ist oder $c \geq \frac{n-1}{\epsilon}$.

Das FPTAS

Test für Grenzen von *OPT*

TEST(*k*) erfüllt seinen Zweck:

$$c < \frac{n-1}{\epsilon} \rightarrow \text{Es gibt Pfad} < k(1 + \epsilon)$$

$$k \leq k$$

$$\frac{k\epsilon}{n-1} \frac{n-1}{\epsilon} \leq k$$

Durch Einsetzen folgt:

$$\frac{k\epsilon}{n-1} c < k$$

$$\frac{k\epsilon}{n-1} c + k\epsilon < k + k\epsilon$$

$$\frac{k\epsilon}{n-1} c + k\epsilon < k(1 + \epsilon)$$

Das FPTAS

Test für Grenzen von OPT

$TEST(k)$ erfüllt seinen Zweck:

$c \geq \frac{n-1}{\epsilon} \rightarrow$ Jeder T -Pfad hat Länge $\geq k$

$$\begin{aligned} c &\geq \frac{n-1}{\epsilon} \\ c \frac{k\epsilon}{n-1} &\geq \frac{k\epsilon}{n-1} \frac{n-1}{\epsilon} \\ c \frac{k\epsilon}{n-1} &\geq k \end{aligned}$$

Das FPTAS

Test für Grenzen von *OPT*

Test-Algorithmus

```
1  Setze  $c \leftarrow 0$ 
2  Für alle  $(i,j) \in E$ :
3      Falls  $c_{ij} > k$ , entferne  $(i,j)$ 
4      Sonst  $c_{ij} \leftarrow \lfloor c_{ij}(n-1)/k\epsilon \rfloor$ 
5
6  Falls  $c \geq (n-1)/\epsilon$ , return true
7  Sonst:
8      Wende Algorithmus B an und berechne  $g_j(c)$  für
           $j = 2, \dots, n$ 
9      Falls  $g_n(c) \leq T$ , return false
10     Sonst:
11         Setze  $c \leftarrow c + 1$ 
12         Goto Zeile 6
```

Das FPTAS

Laufzeit von des Tests

- Runden: in $\mathcal{O}(\log n)$ durch binäre Suche, falls nach oben beschränkt ▶ $\mathcal{O}\left(|E| \log \frac{n-1}{\epsilon}\right)$
- Exakter Algorithmus führt $\leq \frac{n-1}{\epsilon}$ Iterationen durch, $\mathcal{O}(|E|)$ pro Iteration
 - ▶ Insgesamt $\mathcal{O}\left(|E| \log \frac{n-1}{\epsilon} + |E| \frac{n-1}{\epsilon}\right) = \mathcal{O}\left(|E| \frac{n-1}{\epsilon}\right)$

Das FPTAS

Verbesserte Grenzen von OPT

Wir benötigen obere und untere Schranken $LB \leq OPT \leq UB$.

Lower Bound

- $LB = 1$
- $LB =$ kürzester Pfad nach Kosten

Das FPTAS

Verbesserte Grenzen von OPT

Wir benötigen obere und untere Schranken $LB \leq OPT \leq UB$.

Upper Bound

- $UB = \sum (n - 1)$ längste Kanten
- $UB =$ Kosten schnellster Pfad von 1 nach n .

Das FPTAS

Algorithmus

Approximationsschema-Algorithmus

```
1   $UB := \sum (n-1)$  größte Kosten
2   $LB := 1$ 
3
4  Falls  $UB \leq 2LB$ , Goto Zeile 11
5  Sonst:
6       $k := \sqrt{UB \cdot LB}$ 
7      Falls  $TEST(k) == \text{true}$ ,  $LB := k$ 
8      Sonst  $UB := k(1 + \epsilon)$ 
9      Goto Zeile 4
10
11 Setze  $c_{ij} \leftarrow c_{ij}(n-1)/LB\epsilon$ 
12 Berechne optimale Lösung
```

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$
- Rundung der Kantenkosten: Laufzeit letzte Anwendung des exakten Algorithmus' ist $\mathcal{O}\left(|E|OPT\frac{(n-1)}{LB\epsilon}\right)$

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$
- Rundung der Kantenkosten: Laufzeit letzte Anwendung des exakten Algorithmus' ist $\mathcal{O}\left(|E|OPT\frac{(n-1)}{LB\epsilon}\right)$
- $OPT \leq 2LB$: $\mathcal{O}(|E|2LB\frac{(n-1)}{LB\epsilon}) = \mathcal{O}(|E|2\frac{(n-1)}{\epsilon}) = \mathcal{O}(|E|\frac{n-1}{\epsilon})$

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$
- Rundung der Kantenkosten: Laufzeit letzte Anwendung des exakten Algorithmus' ist $\mathcal{O}\left(|E|OPT\frac{(n-1)}{LB\epsilon}\right)$
- $OPT \leq 2LB$: $\mathcal{O}(|E|2LB\frac{(n-1)}{LB\epsilon}) = \mathcal{O}(|E|2\frac{(n-1)}{\epsilon}) = \mathcal{O}(|E|\frac{n-1}{\epsilon})$
- Laut Hassin $\log \log \frac{UB}{LB}$ Tests, bis $\frac{UB}{LB} \leq 2 \pm \epsilon$

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$
- Rundung der Kantenkosten: Laufzeit letzte Anwendung des exakten Algorithmus' ist $\mathcal{O}\left(|E|OPT\frac{(n-1)}{LB\epsilon}\right)$
- $OPT \leq 2LB$: $\mathcal{O}(|E|2LB\frac{(n-1)}{LB\epsilon}) = \mathcal{O}(|E|2\frac{(n-1)}{\epsilon}) = \mathcal{O}(|E|\frac{n-1}{\epsilon})$
- Laut Hassin $\log \log \frac{UB}{LB}$ Tests, bis $\frac{UB}{LB} \leq 2 \pm$
- Wurzeln evtl. teuer, es reicht aber $\log \log \frac{UB}{LB} \pm$

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$
- Rundung der Kantenkosten: Laufzeit letzte Anwendung des exakten Algorithmus' ist $\mathcal{O}\left(|E|OPT\frac{(n-1)}{LB\epsilon}\right)$
- $OPT \leq 2LB$: $\mathcal{O}\left(|E|2LB\frac{(n-1)}{LB\epsilon}\right) = \mathcal{O}\left(|E|2\frac{(n-1)}{\epsilon}\right) = \mathcal{O}\left(|E|\frac{n-1}{\epsilon}\right)$
- Laut Hassin $\log \log \frac{UB}{LB}$ Tests, bis $\frac{UB}{LB} \leq 2 \pm$
- Wurzeln evtl. teuer, es reicht aber $\log \log \frac{UB}{LB} \pm$
- Einzelne Aufrufe von $TEST$: $\mathcal{O}\left(|E|\frac{n-1}{\epsilon}\right)$

Das FPTAS

Laufzeit des FPTAS

- Wie für $TEST$ kann Abweichung vom OPT nicht größer als $k\epsilon$ sein, hier mit $k = LB$, die Abweichung ist also $\leq OPT\epsilon$
- Rundung der Kantenkosten: Laufzeit letzte Anwendung des exakten Algorithmus' ist $\mathcal{O}\left(|E|OPT\frac{(n-1)}{LB\epsilon}\right)$
- $OPT \leq 2LB$: $\mathcal{O}(|E|2LB\frac{(n-1)}{LB\epsilon}) = \mathcal{O}(|E|2\frac{(n-1)}{\epsilon}) = \mathcal{O}\left(|E|\frac{n-1}{\epsilon}\right)$
- Laut Hassin $\log \log \frac{UB}{LB}$ Tests, bis $\frac{UB}{LB} \leq 2 \pm$
- Wurzeln evtl. teuer, es reicht aber $\log \log \frac{UB}{LB} \pm$
- Einzelne Aufrufe von $TEST$: $\mathcal{O}\left(|E|\frac{n-1}{\epsilon}\right)$
- Insgesamt $\mathcal{O}\left(\log \log \frac{UB}{LB} \cdot \left(|E|\frac{n-1}{\epsilon} + \log \log \frac{UB}{LB}\right)\right)$

Fin.