

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

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

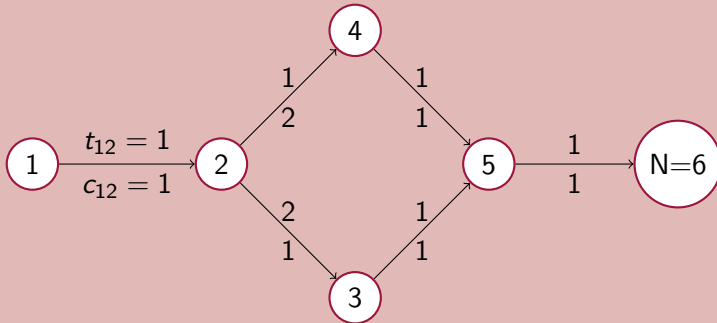
Man weiß $OPT = \min \{c \mid g_n(c) \leq T\}$

- 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:

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

$$k \leq k$$

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

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:

$$c \geq \frac{n-1}{\epsilon}$$

$$c \geq \frac{n-1}{\epsilon}$$

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

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

Das FPTAS

Test für Grenzen von OPT