3.3 Das eingeschränkte Systemoptimum

Motivation:

Vermeide Preis der Anarchie $\frac{UE}{SO}$

Vermeide Unfairness des SO (einzelne bekommen "lange" Routen)

Unfairness einer Routenzuweisung f

$$= \max_k \frac{\max \{ \tau_P(f) \mid P \in \mathcal{P}_k \}}{\text{Fahrzeit } L_k(UE) \text{ im } UE}$$

Bemerkung: unfairness $(UE) = 1$

unfairness $(SO) \to \infty$ ($\S 1$)

Daher Idee (Jahn, M., Schulz, Stier 05)

Lasse nur Wege zu, deren Fahrzeit bzgl. UE-Fahrzeiten nur wenig über $L_k(f)$ liegen,

d.h. $\mathcal{P}_k^\varepsilon = \{ P \in \mathcal{P}_k \mid \sum_{a \in P} \tau_a(UE_a) \leq (1+\varepsilon) L_k(UE) \}$

⇓

Herleitung des Frank-Wolfe Algorithmus wie bisher, nur mit Menge $\mathcal{P}_k^\varepsilon$ statt $\mathcal{P}_k$

↑ hängt nicht vom aktuellen Fluss f ab

d.h. Nebenbedingungen sind immer $\begin{cases} \sum_{P \in \mathcal{P}_k^\varepsilon} f_P = d_k & k \in C \\ f_P \geq 0 \end{cases}$

⇒ Berechnung der Abstiegsrichtung führt auf die Teilaufgabe

Berechne pro Commodity $k \in C$ einen kürzesten Weg bzgl $c_a^l(x_a^l)$ aus P_k^ε

↓

Constrained shortest path Problem (CSP)

Gegeben: Digraph $G = (V, A)$, $s, t \in V$

2 Kantenbewertungen τ_a "Zeit"

l_a "Länge", Schranke L

Aufgabe: Berechne kürzesten s,t Weg P bzgl. τ_a

mit $l(P) = \sum_{a \in P} l_a \leq L$

Leider Komplexitätssprung: CSP ist (schwach) NP-schwer

genaueres Studium in § 4

§4 Constrained shortest paths

4.1 Komplexität

4.1 PROPOS. Das CSP ist schwach NP-vollständig

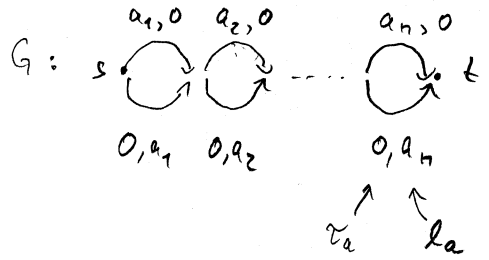
Beweis: Reduktion von PARTITION:

Geg. Zahlen $a_1, \dots, a_n$ mit $\sum a_i = 2b$

Frage: Gibt es Indexmenge I mit $\sum_{i \in I} a_i = b$?

Sei I Instanz von PARTITION.

Konstruiere daraus Instanz I' von CSP:



Frage: Gibt es s,t Weg P mit $\tau(P) \leq b$, $l(P) \leq b$

Offenbar ist die Antwort auf diese Frage ja

$\Leftrightarrow$ $\exists$ Indexmenge I mit $\sum_{i \in I} a_i = b$ □

4.2 PROPOSITION: CSP kann in pseudopolynomialer Zeit gelöst werden.

Beweis: analog zu SUBSET SUM (Verallgemeinerung von PARTITION) aus ADM I □

4.2 Lösungsansätze

1) Bearley & Christofides 89 Brand & Bound + Lagrange Relaxation

2) Labeling Algorithmus (erweiterter Dijkstra)
   Verwandschaft zu Pareto-optimalen Wegen

3) Genetisch (Telekom + Ziegelmann 2000) → § 4.4

4.2.1 Ansatz von Bearley & Christofides

Brand & Bound Baum ≙ wie bei Telekom Problem aus ADM II
≙ IP-Formulierung Festlegen von Kanten

Lagrange Relaxation:

min $\tau(P)$

so daß $l(P) \leq L$] in Zielfkt

P ist Weg von s nach t

$$(LR_\mu) \quad \min \underbrace{\tau(P) + \mu(l(P) - L)}_{=: \Lambda(\mu),\ \mu \in \mathbb{R}_+^1}$$

$$\sum_{a \in P} (\tau_a + \mu l_a) - \underbrace{\mu L}_{\text{Konstant für festes } \mu}$$

↑

Kürzester Weg bzgl. neuer Kantenbewertung $\tau_a + \mu l_a$

Sei P^* opt. Lösung des CSP und $\tau^* := \tau(P^*)$

Dann gilt (1) $\Lambda(\mu) \leq \tau^* \quad \forall \mu \geq 0$

(2) $\Lambda^* := \max_{\mu \geq 0} \Lambda(\mu) \leq \tau^*$

(3) Ist P eine Optimallösung von (LR_μ) und $l(P) \leq L$ $\Rightarrow$ P opt für CSPP

$\tau^* - \Lambda^*$ heißt <u>Dualitätslücke</u>

<u>Subgradientenverfahren</u> analog zu ADM II, § 7.4

Ist P_{μ_0} kürzester Weg in (LR_{μ_0}), so ist $l(P_{\mu_0}) - L$ (verletzte Restr.) ein Subgradient in μ_0 von $\Lambda(\mu)$

Gehe Schritt in dieser Richtung

$$\mu^{neu} = \mu^{alt} + \Theta\,(l(P_{\mu_0}) - L)$$

$\downarrow$

neue Kantenbewertung

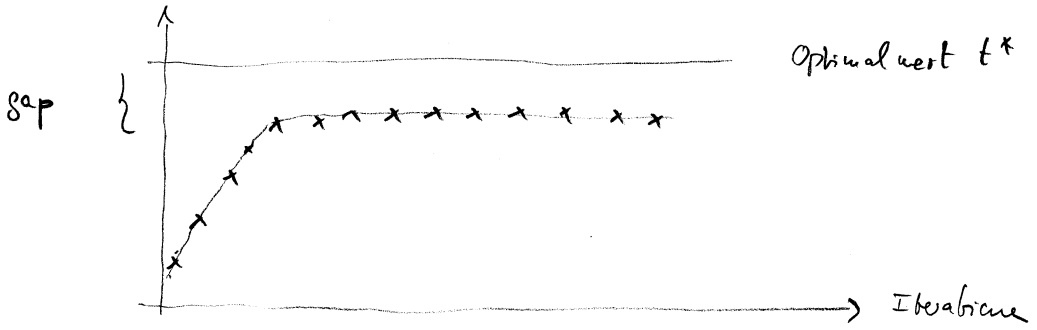
$$\tau_a + \mu^{neu} l_a$$

Unterschiede in Kantenbewertung gerade $(\mu^{neu} - \mu^{alt})\, l_a$

$\uparrow$

dieselbe Zahl für alle Kanten

typischer Verlauf



unbefriedigend

4.2.2 Pareto-optimale Wege

Geg. $G = (V, A)$ $\quad s, t \in V$

Für jede Kante a ein r-dim Bewertungsvektor $\lambda(a) = \begin{pmatrix} \lambda_1(a) \\ \vdots \\ \lambda_r(a) \end{pmatrix} \geq 0$

Länge $\lambda(P)$ eines Weges P ist

$$\lambda(P) = \sum_{a \in P} \lambda(a) = \begin{pmatrix} \sum_{a \in P} \lambda_1(a) \\ \vdots \\ \sum_{a \in P} \lambda_r(a) \end{pmatrix} =: \begin{pmatrix} \lambda_1(P) \\ \vdots \\ \lambda_r(P) \end{pmatrix}$$

↑ hier unterschiedliche Operationen möglich

Σ, max, min ....

[allgemein: Semiringe]

Gesucht: alle Pareto-optimalen s,t Wege

P heißt Pareto-optimal

$:\Leftrightarrow$ $\nexists$ Weg P' mit $\lambda(P') \leq \lambda(P)$ und $\lambda_i(P') < \lambda_i(P)$ für ein i

↑ Komponentenweise Ordnung

(keine Dominanz)

Anwendungen: Autowahrheit: Zeit, Entfernung, verbrauchte Strecken als Kriterien

Bahn Zeit, Kosten, # Umstiege

Der Dijkstra Algorithmus für Pareto-optimale Wege

zunächst Wh: Dijkstra für kürzeste Wege ($\lambda_a \geq 0$)

- berechnet für jeden Knoten v eine Distanz $d[v]$
  $\hat{=}$ Länge kürzester Weg von s zu v
  + einen Knoten Vorgänger[v], der Vorgänger von v auf kürzestem Weg von s zu v ist
- Werte $d[v]$, Vorgänger[v] sind zunächst vorläufig
  Im Lauf des Algorithmus werden Knoten markiert, für markierte Knoten sind diese Werte endgültig

Initialisierung:

$$d[v] := \begin{cases} 0 & v = s \\ \lambda(s,v) & \text{falls Kante } (s,v) \text{ exist.} \\ \infty & \text{sonst} \end{cases}$$

$$\text{Vorgänger}[v] := \begin{cases} s & \text{falls Kante } (s,v) \text{ exist} \\ \infty & \text{sonst} \end{cases}$$

nun s markiert

Hauptschleife

while ∃ unmarkierten Knoten do

  wähle unmarkierten Knoten v mit kleinstem d[v]

  markiere v

  for alle Kanten (v,w) mit unmarkiertem w do

    if $d[v] + \lambda_{(v,w)} < d[w]$ then

      $d[w] := d[v] + \lambda_{(v,w)}$

      Vorgänger[w] := v

    end if

  end for

end while

Variation des Dijkstra für Pareto-optimale Wege [Theune 95]

statt d[v] r-dim Vektoren $d[v] = \begin{pmatrix} d_1[v] \\ \vdots \\ d_r[v] \end{pmatrix}$ $d_k[v]$ bzgl λ_k

In jedem Knoten mehrere d[v] möglich,

≙ bisher ermittelten Pareto-optimalen s,v-Wege

Initialisierung

$$d[v] := \begin{cases} \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} & \text{falls } v = s \\ \lambda_{(s,v)} & \text{falls } (s,v) \text{ Kante} \\ \begin{pmatrix} \infty \\ \vdots \\ \infty \end{pmatrix} & \text{sonst} \end{cases}$$

Markierung Knoten ⟶ Markierung von Vektoren d[v]

Regel: Wähle lexikographisch kleinsten unmarkierten Vektor d[v] als nächsten zu markierenden Vektor

Aktualisierung des $d[w]$:

Betrachte zu gewähltem $d[v]$ alle Kanten (v,w)

nehme $d[v] + \lambda_{(v,w)}$ zu den bereits in w abgespeicherten Vektoren hinzu und streiche nicht Pareto-optimale und aktualisiere ggf. Vorgänger($d[v]$)

Anfangs nur $d[s]$ markiert

Hauptschleife

<u>while</u> $\exists$ unmarkierten Vektor <u>do</u>

wähle lexikographisch kleinsten unmarkierten Vektor $d[v]$

markiere diesen Vektor

Sei v der zugehörige Knoten

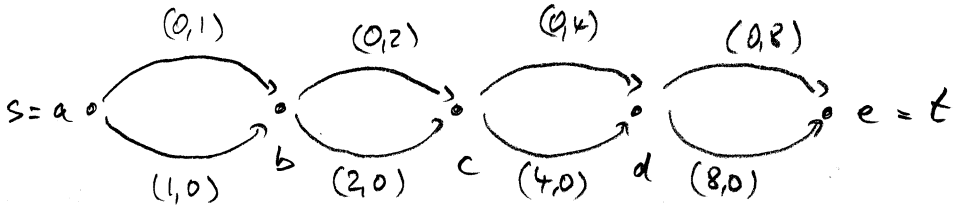
<u>for</u> alle Kanten (v,w) <u>do</u>

nehme $d[w] := d[v] + \lambda(v,w)$ den Vektoren von w hinzu;

streiche nicht Pareto-optimale bei w;

falls $d[w]$ nicht gestrichen wird, so setze Vorgänger$(d[w]) := d[v]$

Beispiel:



Pareto Optima:

a	b	c	d	e
(0,0)	(0,1)	(0,3)	(0,7)	(0,15)
	(1,0)	(2,1)	(4,3)	(8,7)
		(1,2)	(1,6)	(1,14)
		(3,0)	(5,2)	(9,6)
			(2,5)	(2,13)
			(6,1)	(10,5)
			(3,4)	(3,12)
			(7,0)	(11,4)
				(4,11)
				(12,3)
				(5,10)
				(13,2)
				(6,9)
				(14,1)
				(7,8)
				(15,0)
2^0	2^1	2^2	2^3	2^4

exponentielles Wachstum $\Rightarrow$ exponentielles Algorithmus

Korrektheit analog zu normalem Dijkstra per Induktion:
nach # Markierungen:

> Sobald Vektor markiert, gehört er zur Menge der Pareto-Optima zum zugehörigen Knoten