§3 Der Algorithmus von Frank-Wolfe für die (konvexe) Optimierung mit linearen Nebenbedingungen (1956)

3.1 Der allgemeine Fall:

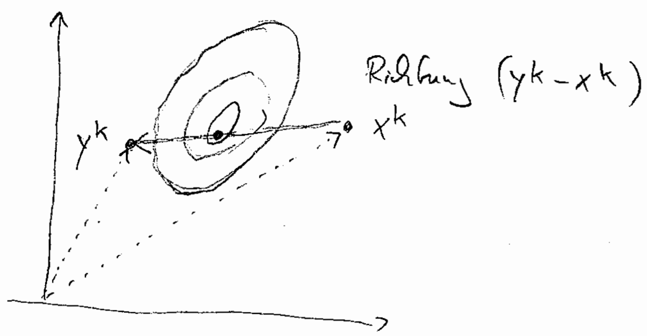
$$\min f(x) \quad \text{unter} \quad Ax = b \quad x \geq 0$$

Algorithmus ist iterativ und erzeugt eine Folge von Punkten $x^0, x^1, \ldots$ wobei x^{k+1} wie folgt aus x^k berechnet wird

(1) Bestimme eine Optimallösung y^k des LP

$$\begin{aligned} \min \quad & \nabla_x f(x^k)^T x \\ \text{unter} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

(2) Wähle x^{k+1} als Minimierer von $f(x)$ auf dem Intervall $[x^k, y^k]$ "Line Search"



3.1 SATZ: Sei f stetig diffbar, konvex und sei

$$X := \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\} \text{ beschränkt}$$

Dann enthält die Folge $x^0, x^1, \ldots$ eine konvergente Teilfolge und jede solche Teilfolge konvergiert gegen ein globales Minimum von f auf X

Vollständiger Beweis siehe z.B. Skript ADM III vom SS 2006

Buch: Dimitris Bertsekas
Nonlinear Programming, 2nd Edition
Athena Scientific, 1999

David G. Luenberger, Yinyu Ye
Linear and Nonlinear Programming
Springer, 2008

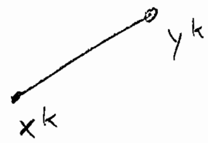
Es bleiben 2 Fragen:

(A) Wie macht man die Line Search?

(B) Wann bricht man mit der Folge der x^k ab?

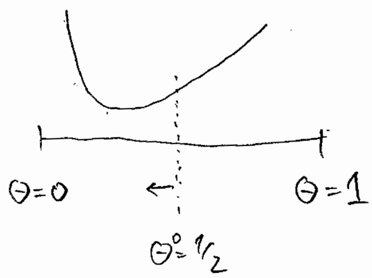
zu (A) Line Search

eine Möglichkeit: binäre Suche auf



$\hat{=}$ Minimierung einer 1-dimensionalen konvexen stetigen Fkt.

$$g(\theta) = f(x^k + \theta(y^k - x^k))$$



dazu muss iterativ die Ableitung (in θ)

$$\frac{dg}{d\theta}(\theta) = \nabla_x f(x^k + \theta(y^k - x^k))^T (y^k - x^k)$$

ausgewertet werden

Ist sie > 0 so $\theta_{i+1} := \theta_i - \frac{1}{2^{i+1}} \|x^k - y^k\|$

Ist sie < 0 so $\theta_{i+1} := \theta_i + \frac{1}{2^{i+1}} \|x^k - y^k\|$

Ist sie 0 oder das verbleibende Intervall $(1/2)^i \|x^k - y^k\|$ klein genug, so STOP

$\Rightarrow$ Abbruch nach i Iterationen mit $(1/2)^i \|x^k - y^k\| < \varepsilon$

$\Rightarrow$ $i \approx \log\left(\frac{1}{\varepsilon} \cdot \|x^k - y^k\|\right)$ viele Auswertungen von ∇f

$\uparrow$

unter Umständen teuer (nicht bei SO)

daher auch andere Methoden betrachtet (z.B. Methode von Armijo, Numerische Mathematik)

zu (B) Abbruchkriterium

$$f \text{ konvex} \underset{\text{diffbar}}{\Longleftrightarrow} f(z) \geq f(x) + \nabla f(x)^T (z - x) \quad \forall x, z \qquad (*)$$

x^* optimal $\underset{\substack{\uparrow \\ (*) \text{ mit } z = x^*}}{\Rightarrow}$ $f(x^*) \geq f(x^k) + \nabla f(x^k)^T (x^* - x^k) \quad \forall\, k$

$\llcorner$ $\{x^k\}$ Folge des Frank-Wolfe-Algo

$\geq f(x^k) + \nabla f(x^k)^T (y^k - x^k) \quad \forall\, k$

$\llcorner$ y^k minimiert $\nabla f(x^k)^T x$

$$\Rightarrow \underbrace{f(x^k) - f(x^*)}_{\substack{\text{verbleibende} \\ \text{"Optimalitätslücke"}}} \leq \underbrace{\left| \nabla f(x^k)^T (y^k - x^k) \right|}_{\text{im Algorithmus bekannte Größen}}$$

$\Rightarrow$ Optimalitätslücke kann im Verfahren kontrolliert werden

$\Rightarrow$ Abbruch kann entsprechend gestaltet werden

3.2 Die Spezialisierung auf das Systemoptimum (SO) und das User-Equilibrium (UE)

Im (SO) haben wir das Minimierungsproblem

$$\min \; C(x(f)) = \sum_{a\in A} x_a(f)\cdot \tau_a(x_a(f))$$

$$\text{mit} \quad x_a(f) = \sum_{P\ni a} f_P \qquad f \text{ Pfadfluss} \quad x \text{ Kantenfluss}$$

$$\sum_{P\in \mathcal{P}_k} f_P = d_k \quad \forall k \in C'$$

$$f_P \geq 0$$

$$\frac{\partial C}{\partial f_P}(x(f)) \; \Big[= \frac{\partial}{\partial f_P} \sum_{a\in A} \underbrace{x_a(f)\cdot\tau_a(x_a(f))}_{=: c_a(x_a(f))}$$

vgl. § 2

$$= \sum_{a\in A} \frac{dc_a}{dx_a}(x_a)\cdot \underbrace{\frac{\partial x_a}{\partial f_P}(f)}_{= 1 \text{ wenn } a\in P \text{ und } 0 \text{ sonst}} \qquad \text{Kettenregel} \Big] \; §2$$

$$(*) \quad = \sum_{a\in P} \frac{dc_a}{dx_a}(x_a)$$

$$\Big[= \sum_{a\in P} \underbrace{\left(x_a\cdot\tau_a'(x_a) + \tau_a(x_a)\right)}_{c_a'(x_a) \text{ aus } §1} \Big]$$

$$= \text{Länge des Weges } P \text{ bzgl. } \; c_a'(x_a)$$

Also ist $\nabla_x C(\bar{x})^T\cdot x = \sum_{a\in A} \frac{dc_a}{dx_a}(\bar{x}_a)\cdot x_a = \sum_{a\in A} \frac{dc_a}{dx_a}(\bar{x}_a) \sum_{P\ni a} f_P$

$$= \sum_P \sum_{a\in P} \frac{dc_a}{dx_a}(\bar{x}_a)\cdot f_P \;\; \overset{(*)}{=} \;\; \sum_P \frac{\partial C}{\partial f_P}(\bar{f})\cdot f_P$$

Summation vertauschen

$$= \nabla_f C_1(\bar{f})^T \cdot f$$

Also $\nabla_f C_1(\bar{f})^T f = \nabla_x C(\bar{x})^T \cdot x$ mit $\bar{x} = \bar{x}(\bar{f})$

$\uparrow$ In Pfad-formulierung $\qquad \uparrow$ In Kanten-formulierung

$\Rightarrow$ das zu lösende LP im Frank-Wolfe Algorithmus kann in Kantenform gelöst werden (brauchen nicht exponentiell viele Wege in Formulierung der Zielfunktion) und die Bewertung der Kante ist $\frac{dc_a}{dx_a}(x_a^l) = \underbrace{x_a^l \cdot \tau'(x_a^l) + \tau(x_a^l)}_{c_a'(x_a^k)}$ mit x^l = Kantenfluss in momentaner Iteration l

$\Rightarrow$ $\min \nabla C_1(x^l)^T x$

unter $\sum_{P \in \mathcal{P}_k} f_P^l = d_k \quad k \in C$

$f_P \geq 0$

$\hat{=}$ Min Cost Multi Commodity Fluss Problem mit Kantenkosten $c_a'(x_a^l)$

da keine Kapazitäten

$\Rightarrow$ zerfällt in Kürzeste-Wege-Problem für jede Commodity k bzgl. Kantenkosten $c_a'(x_a^l)$

3.2 SATZ: Der erste Teil des Frank-Wolfe Algorithmus reduziert sich in jeder Iteration auf die Berechnung eines Kürzesten Weges für jedes Quelle-Senke Paar bzgl. der Kantenkosten $c_a'(x_a^l) = x_a^l \tau_a'(x_a^l) + \tau_a(x_a^l)$

Line-Search:

Sei y^ℓ die optimale Lösung dieses ersten Teils

$\Rightarrow y^\ell$ besteht aus $|C|$ vielen Wegen Q_k, $k \in C$, mit $y_{Q_k} = d_k$

vorige Lösung f^ℓ besteht bereits aus gewissen Wegen $P \in \mathcal{P}_k$

mit $\sum\limits_{P \in \mathcal{P}_k} f_P^\ell = d_k$, $k \in C$

Neue Lösung $f^{\ell+1}$ ergibt sich dann als

$$f^\ell + \Theta(y^\ell - f^\ell)$$

a) Fließt bereits Fluss auf Q_k, d.h. $f_{Q_k}^\ell > 0$

so ergibt sich für die entsprechende Komponente von $f^{\ell+1}$:

$$f_{Q_k}^{\ell+1} = f_{Q_k}^\ell + \Theta\left(d_k - f_{Q_k}^\ell\right) = \underbrace{(1-\Theta) f_{Q_k}^\ell + \Theta \cdot d_k}_{\substack{\text{Konvexkombination} \\ \text{alte und neue Lösung}}}$$

b) Fließt noch kein Fluss auf Q_k,

so ergibt sich die entsprechende Komponente von $f^{\ell+1}$ als

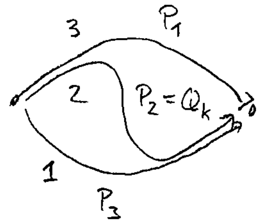
$$f_{Q_k}^{\ell+1} = f_{Q_k}^\ell + \Theta\left(d_k - f_{Q_k}^\ell\right) = \Theta d_k$$

c) und für alle Komponenten $P \in \mathcal{P}_k$, $P \neq Q_k$ wird Fluss weggenommen

$$f_P^{\ell+1} = f_P^\ell + \Theta\left(0 - f_P^\ell\right) = (1-\Theta) f_P^\ell$$

Beispiel:

$d_k = 6$



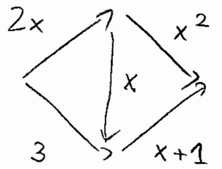
neuer Fluss für Commodity k ergibt sich zu

$$\begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} (1-\Theta) + \Theta \begin{pmatrix} 0 \\ 6 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 - 3\Theta \\ 2 + 4\Theta \\ 1 - \Theta \end{pmatrix} \geq 0$$

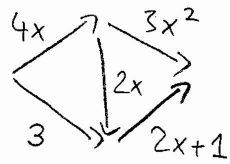
Wert von Θ ergibt sich durch die Line Search

bzgl. $\nabla_x C\left(x^k + \Theta(y^k - x^k)\right)^T (y^k - x^k)$ (*)

im Bsp.



$\longrightarrow$



$\tau_a(x)$ $\qquad\qquad$ $C'_a(x) = x_a \cdot \tau'_a(x_a) + \tau_a(x_a)$

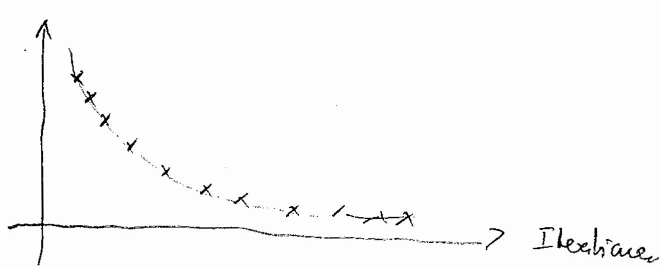
$$\nabla C(x) = \begin{pmatrix} 4x \\ 3 \\ 2x \\ 3x^2 \\ 2x+1 \end{pmatrix} \qquad \text{Wert von (*) für } \Theta: \begin{pmatrix} 4\cdot(5+\Theta) \\ 3\cdot(1-\Theta) \\ 2\cdot(2+4\Theta) \\ 3\cdot(3-3\Theta)^2 \\ 2(3+3\Theta)+1 \end{pmatrix}^T \cdot \begin{pmatrix} 6-5 \\ 0-1 \\ 6-2 \\ 0-3 \\ 6-3 \end{pmatrix} =: g(\Theta)$$

$\Downarrow$

3.3 PROPOSITION Line Search benötigt nur die Wege die Fluss führen + die neu berechneten Wege in der laufenden Iteration (also nicht alle potentielle exponentiell vielen). Die Bestimmung von Θ kann im Kantenraum erfolgen.

Abbruch entweder über die Optimalitätslücke (§ 3.1) oder über Beobachtungen der Weglängen bzgl. $c'_a(x_a)$ (Abbruch wenn sie ungefähr gleich sind, d.h. neu berechnete Wege unterscheiden sich kaum von den vorherigen, dies nutzt die UE Interpretation von SO)

Typischerweise ist der Gewinn groß in den ersten Iterationen, später kleiner



Bemerkungen

(1) Das Verfahren von Frank-Wolfe wird in diesem Zusammenhang auch als <u>convex combinations Algorithmus</u> bezeichnet

(2) Es kann auch für die Berechnung des UE verwendet werden

$$C(x(f)) = \sum_{a \in A} \underbrace{\int_0^{x_a(f)} \tau_a(t)\,dt}_{C_a(x_a)} \quad , \text{ Nebenbedingungen bleiben gleich}$$

$$\Rightarrow C'_a(x_a) = \tau_a(x_a)$$

Ansonsten bleibt alles gleich!

> **3.4 SATZ** Bei der Berechnung des UE mit Frank-Wolfe reduziert sich im ersten Teil die Berechnung auf die Berechnung eines kürzesten Weges für jedes Quelle-Senke Paar bzgl. der Kantenkosten $\tau_a(x_a^\ell)$

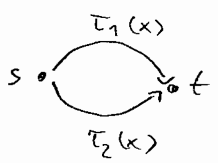
(3) Ein naheliegender "Fixpunkt-Ansatz" funktioniert i.A. nicht

$$f^{\ell+1} = \text{optimaler Fluss bzgl der Fahrzeiten } \tau_a(x_a^\ell(f^\ell))$$

aus der vorigen Iteration

da Oszillation auftreten kann

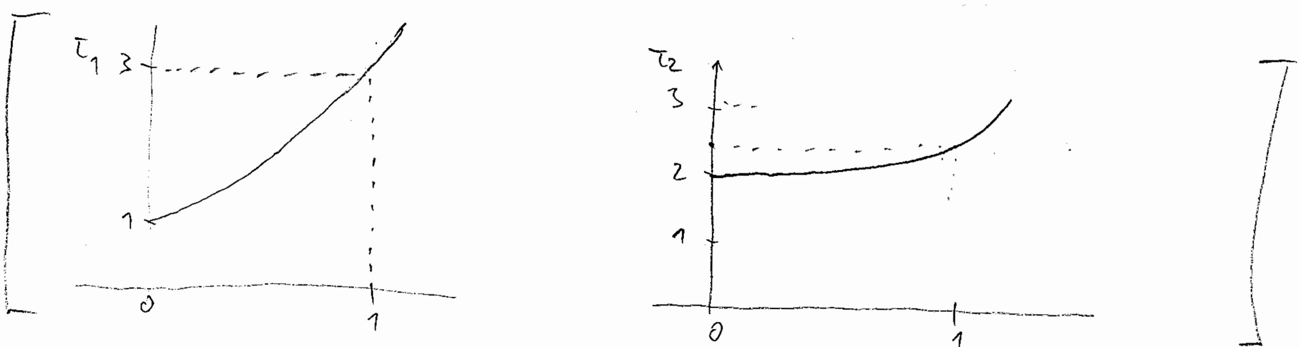
Beispiel:



$d = 1$

$\tau_1(x) = x_1$

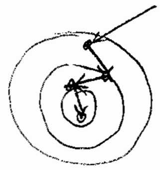
$\tau_2(x) = x_2$



Aufgabe 3.1

Startfluss $x_1 = 1$, $x_2 = 0$. Berechne Folge der Flüsse mit "Fixpunkt-Ansatz" und mit dem Frank-Wolfe Algo

(4) Bei Frank-Wolfe kann Zickzack-Verhalten auftreten



Es gibt verschiedene Verbesserungen um dies zu vermeiden, z.B. die Partan Methode (LeBlanc, Helgason, Boyce 85)

intelligentere Line Search durch Ausnutzung der zwei letzten Iterationen, d.h. x^l, y^l und x^{l-1}, y^{l-1}

$\Rightarrow$ 30% Verbesserung der Rechenzeit in unseren Rechnungen

$\boxed{\text{Aufgabe 3.2}}$ Was muss geändert werden, wenn zusätzlich der Fluss pro Kante durch eine Kapazität beschränkt ist, d.h. $x_a \le u_a$?

3.3 Das eingeschränkte Systemoptimum

<u>Motivation:</u>

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

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

<u>Unfairness einer Routenzuweisung f</u>

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

Bemerkung: unfairness (UE) = 1

unfairness (SO) $\to \infty$ (§ 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. $\mathcal{P}_k^\varepsilon = \{ P \in \mathcal{P}_k \mid \sum_{a \in P} \tau_a(UE_a) \le (1+\varepsilon) L_k(UE) \}$

$\Downarrow$

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

$\uparrow$ hängt nicht vom aktuellen Fluss f ab

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

$\Rightarrow$ Berechnung der Abstiegsrichtung führt auf die Aufgabe

Berechne pro Commodity $k \in C$ einen kürzesten Weg bzgl $c'_a(x_a^\ell)$ aus $\mathcal{P}_k^\varepsilon$

$\downarrow$

<u>Constrained shortest path Problem</u> (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 $\ell(P) = \sum_{a \in P} \ell_a \leq L$

Leider Komplexitätssprung: CSP ist (schwach) NP-schwer

genaueres Studium in § 4