

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

3.1 Der allgemeine Fall:

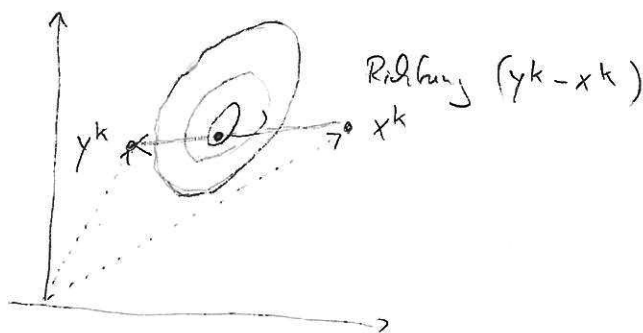
$$\text{min } f(x) \quad \text{unter } Ax = b \quad x \geq 0$$

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

(1) Bestimme eine Optimallösung y^k des LP

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

(2) Wähle x^{k+1} als Minimumes 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, \dots 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

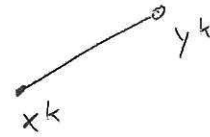
Es bleiben 2 Fragen:

(A) Wie macht man die Line Search?

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

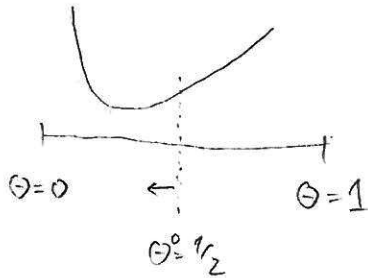
zu (A) Line Search

eine Möglichkeit: Lineare Suche auf



= Minimierung eines 1-dimensionalen konvexen stetigen Fkt.

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



dazu muss zusätzlich die Ableitung

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

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

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

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

↑

unter Umständen teurer (nicht bei SO)

daher auch andere Methoden

betrachtet (z.B. Methode von Armijo,

Numerische Mathematik)

zu (B) Abbruchkriterium

$$f \text{ konvex} \iff f(z) \geq f(x) + \nabla f(x)^T (z-x) \quad \forall x, z$$

↑
differenzierbar

$$\begin{aligned}
 x^* \text{ optimal} & \stackrel{\substack{=0 \\ \uparrow \\ \text{mit } z=x^*}}{=} f(x^*) \geq f(x^k) + \nabla f(x^k)^\top (x^* - x^k) \quad \forall k \\
 & \quad \uparrow \{x^k\} \text{ Folge des Frank-Wolfe-Verf.} \\
 & \geq f(x^k) + \nabla f(x^k)^\top (y^k - x^k) \quad \forall k \\
 & \quad \uparrow y^k \text{ minimiert } \nabla f(x^k)^\top x
 \end{aligned}$$

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

verbleibende "Optimalitätslücke"
 in Algorithmus bekannte Größen

\Rightarrow Optimalitätslücke kann im Verfahren kontrolliert werden

\Rightarrow Abbruch kann entsprechend gesteuert werden

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

Im (SO) haben wir das Minimierungsproblem

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

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

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

$$f_P \geq 0$$

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

↑
vgl. § 2

$$= \sum_{a \in A} \frac{dc_a}{dx_a}(x_a) \cdot \underbrace{\frac{\partial x_a}{\partial f_P}(f)}_{\text{Kettenregel}}$$

= 1 wenn $a \in P$ und 0 sonst

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

$$= \sum_{a \in P} \underbrace{(x_a \cdot \tau'_a(x_a) + \tau_a(x_a))}_{c'_a(x_a) \text{ aus § 1}}$$

$c'_a(x_a)$ aus § 1

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

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

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

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

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

↑
In Pfad-
formulierung

↑
In Kanten-
formulierung

=0 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 des Kante ist $\frac{dc_a}{dx_a}(x_a^l) = \underbrace{x_a^l \cdot \tau'(x_a^l) + \tau(x_a^l)}_{c'_a(x_a^l)}$ mit $x^l =$ Kantenfluss in momentaner Iteration l

$$=0 \quad \min \nabla C_1(x^l)^T x$$

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

$$f_P \geq 0$$

 $\hat{=}$

Min Cost Multi Commodity
Fluss Problem mit
Kantenkosten $c'_a(x_a^l)$

da keine Kapazitäten

=0 zerfällt in kürzeste-Weg 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

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

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

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

Neue Lösung f^{l+1} ergibt sich dann als

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

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

so ergibt sich für die entsprechende Komponente von f^{l+1} :

$$f_{Q_k}^{l+1} = f_{Q_k}^l + \Theta(d_k - f_{Q_k}^l) = \underbrace{(1-\Theta)f_{Q_k}^l + \Theta \cdot d_k}_{\text{Konvexkombination}}$$

alte und neue Lösung

b) Fließt noch kein Fluss auf Q_k ,

so ergibt sich die entsprechende Komponente von f^{l+1} als

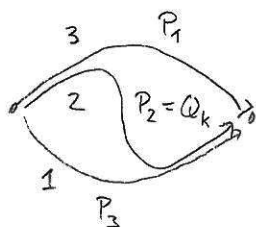
$$f_{Q_k}^{l+1} = f_{Q_k}^l + \Theta(d_k - f_{Q_k}^l) = \Theta d_k$$

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

$$f_P^{l+1} = f_P^l + \Theta(0 - f_P^l) = (1-\Theta)f_P^l$$

Beispiel:

$$d_k = 6$$

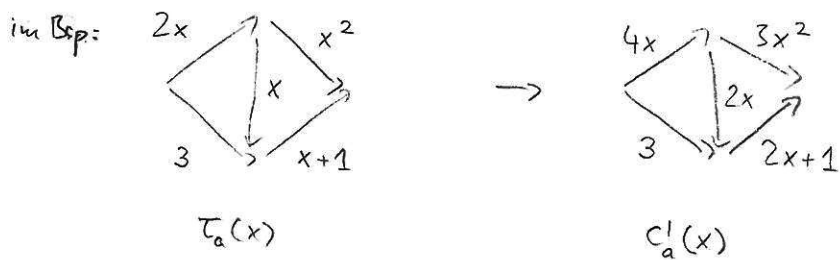


neuer Fluss für Commodity k ergibt sich zu

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

Wert von θ ergibt sich durch die Line Search

$$\text{Bzgl. } \nabla_x C_1(x^k + \theta(y^k - x^k))(y^k - x^k) \quad (*)$$



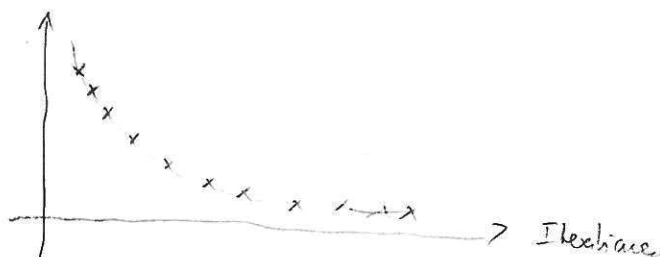
$$\nabla C(x) = \begin{pmatrix} 4x \\ 3 \\ 2x \\ 3x^2 \\ 2x+1 \end{pmatrix} \quad \text{Wert von } (*) \text{ für } \theta = \begin{pmatrix} 4 \cdot (5+\theta) \\ 3 \cdot (1-\theta) \\ 2 \cdot (2+4\theta) \\ 3 \cdot (3-3\theta)^2 \\ 2(3+3\theta)+1 \end{pmatrix}^T \cdot \begin{pmatrix} 6-5 \\ 0-1 \\ 6-2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

⇓

3.3 PROPOSITION Line Search benötigt nur die Werte der Flüsse fixieren + die neu berechneten Werte in der laufenden Iteration (also nicht alle potentiell expandieren). Die Bestimmung von θ kann im Kantensraum 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. wenn benachbarte Wege unterschieden sich kaum werden vorziehen, dies nutzt die UE Interpretation von S_0)

Typischerweise ist das Gerüst groß in den ersten Iterationen, später kleiner



Bemerkungen

(1) Das Verfahren von Frank-Wolfe wird in diesem Zusammenhang auch als convex combination algorithm bezeichnet

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

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

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

Kosten bleibt alles gleich!

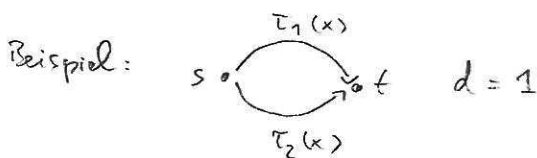
3.4 SATZ Bei der Berechnung des UE mit Frank-Wolfe reduziert sich im ersten Teil die Berechnung auf die Berechnung eines kürzesten Weges für jedes Quelle-Senke Paar bzgl. der Kantenkosten $\tau_a(x_a^e)$

(3) Ein nahe liegendes "Fixpunkt-Insatz" funktioniert i. d. nicht

$$f^{k+1} = \text{optimale Fluss bzgl. der Faktoren } \tau_a(x_a^e(f^k))$$

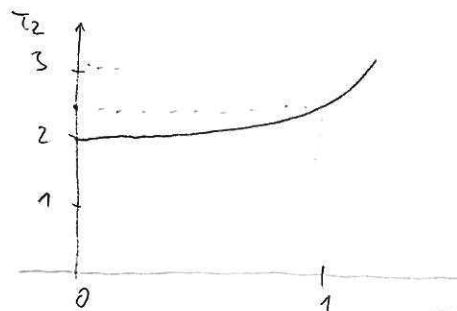
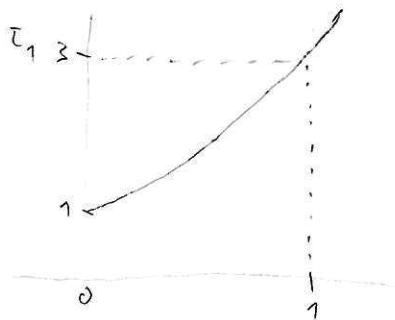
aus der vorigen Iteration

da Oszillation auftreten kann



$$\tau_1(x) = 1 + \frac{2x}{2-x}$$

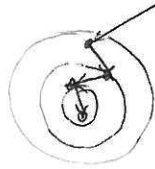
$$\tau_2(x) = 2 + \frac{x/2}{2-x}$$



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

Aufgabe 4

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



Es gibt verschiedene Verbesserungen um dies zu vermeiden, z.B.

die Postau Methode (LeBlanc, Helgason, Boyce 85)

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

=> 30% Verbesserung der Rechenzeit in unseren Rechnungen

Aufgabe 5

Was muss geändert werden, wenn zusätzlich
der Fluss pro Kante durch eine Kapazität
beschränkt ist, d.h. $x_a \leq u_a$?

3.3 Das eingeschränkte Systemoptimum

Motivation:Vermeide Preis des Inaktive $\frac{UE}{SO}$

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

Unfairness einer Routenzuweisung f

$$= \max_k \frac{\max \{ \tau_p(f) \mid p \in P_k \}}{\text{Fahrzeit } L_k(UE) \text{ im } UE}$$

Bemerkung: unfairness (UE) = 1

unfairness (SO) $\rightarrow \infty$ (§ 1)

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

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

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

↓

Herleitung des Frank-Wolfe Algorithmus wie bisher, nur mit

Menge \bar{P}_k^ε statt P_k ↑ hängt nicht vom aktuellen Fluss f ab

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

=0 Berechnung der Abstiegsrichtung führt auf die Aufgabe

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



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

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

Leider Komplexitätssprung: CSP ist (schwach) NP-schwer
genauerer Studium in § 4