

## Fraktionale Optimierung

Fraktional heißt in diesem Fall, die Zielfunktion ist gebrochen. Z.B. Trade-Off zwischen Kosten/Gewinnen berechnen.

*klassisches Optimierungsproblem (P):*

$$\max \sum_i c_i x_i$$

Suche Maximum in  $S$ , der Menge der zulässigen Lösungen. Wir nehmen an, P ist lösbar in „vernünftiger“ Zeit.

*multikriterielle Optimierung:*

Zwei Daten auf den Entscheidungsobjekten:

$$\sum_i a_i x_i \quad \sum_i b_i x_i, \quad x \in S$$

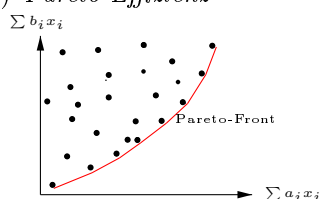
Was ist eine optimale Lösung?

(i) *gewichtete Summe*

$$\sum_i a_i x_i - \lambda \sum_i b_i x_i = \sum_i (a_i - \lambda b_i) x_i, \quad x \in S$$

$\lambda$  ist ein Skalierungsfaktor (z.B. Äpfel in Birnen umrechnen ;-)

(ii) *Pareto-Effizienz*



Pareto-Punkte: Punkte, die nur in eine Richtung verbessert werden können. Alle Pareto-Punkte bilden die Pareto-Front.

(iii) *fraktionale Optimierung*

Suche das optimale Verhältnis zwischen  $\sum_i a_i x_i$  und  $\sum_i b_i x_i$ .

$$(F) \quad \max \frac{\sum_i a_i x_i}{\sum_i b_i x_i} \quad x \in S$$

Es muß  $\sum_i b_i x_i > 0$  gelten. Manchmal kann es allerdings nötig sein, zusätzlich einen Basiswert hinzuzuzählen. In diesem Fall ist es auch möglich, eine Lösung mit  $\sum_i b_i x_i = 0$  zuzulassen:

$$(F') \quad \max \frac{\sum_i a_i x_i}{\sum_i b_i x_i + b_0} \quad x \in S$$

*klassischer Lösungsansatz:*

Definiere parametrische Funktion  $v(t)$  wie folgt:

$$v(t) := \max_{X \in S} \left\{ \sum_i a_i x_i - t \cdot \sum b_i x_i \right\}$$

Dabei ist für ein festes  $t_0$   $v(t_0)$  durch Lösungen von P vernünftig berechenbar.

$t$  ist ein "Schätzwert" für  $z^*$ . Parametrische Suche auf  $v(t)$  nach einem  $t^*$  mit  $t^* = z^*$ . Betrachte dazu mögliche Kandidaten  $t'$ .

(i)  $v(t') > 0$

$$\begin{aligned} v(t') &= \sum a_i x'_i - t' \cdot \sum b_i x'_i > 0 \\ &\Leftrightarrow \frac{\sum_i a_i x'_i}{\sum_i b_i x'_i} > 0 \\ &\Rightarrow t^* > t' \end{aligned}$$

(ii)  $v(t') = 0$

$$\begin{aligned} v(t') &= \sum a_i x'_i - t' \cdot \sum b_i x'_i = 0 && \Leftrightarrow \frac{\sum_i a_i x'_i}{\sum_i b_i x'_i} = t' \\ &\Rightarrow t' = t^*, \quad x' \text{ ist optimale Lösung} \end{aligned}$$

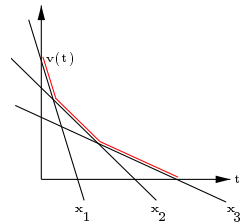
(iii)  $v(t') < 0$

$$\begin{aligned} v(t') &= \sum a_i x'_i - t' \cdot \sum b_i x'_i < 0 \\ &\Leftrightarrow \frac{\sum_i a_i x'_i}{\sum_i b_i x'_i} < 0 \\ &\Rightarrow t^* < t' \end{aligned}$$

Nullstellen von  $v(t)$  mit  $v(t^*) = 0$  liefern optimale Lösungen von (F) oder (F').

Annahme: Sei nun  $|S|$  endlich. Jede Lösung  $X \in S$  liefert eine Gerade  $\sum_i a_i x_i - t \cdot \sum_i b_i x_i$ .

Eigenschaften von  $v(t)$ :



- stückweise linear
- monoton fallend
- konvex

Suchstrategien auf  $v(t)$

(i) *Binäre Suche*

Suchintervall iterativ halbieren

$$t_\ell = 0$$

$$t_U = \frac{\max \sum_i a_i x_i}{\min \sum_i b_i x_i} \text{ Getrennte Optimierungsprobleme!}$$

*Exponentielle Suche* zum Finden der oberen Schranke

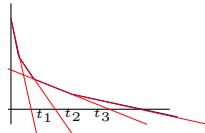
Nachteile:

- keine Laufzeitkomplexität

- Abbruchkriterium

Lösung: Stimmt Steigung an unterer und oberer Schranke überein, so hat man das richtige Geradensegment gefunden (stückweise Linearität).

(ii) *Newton'sche Methode*



Durch Konvexität ist der neue Testwert kleiner als die wahre Nullstelle, bis man das richtige Geradensegment getroffen hat.

Außerdem:  $t_i \rightarrow x^j$ ,  $t_{j+1}$  ist Nullstelle der Tangente  $i$ .

$$\sum_i a_i x_i^j - t_j \sum_i b_i x_i^j$$

ist bereits die Tangente, die wir suchen, und die Nullstelle ist

$$t_{j+1} = \frac{\sum_i a_i x_i^j}{\sum_i b_i x_i^j}.$$

$\Rightarrow$  letzter Zielwert ist die nächste Teststelle.

Satz (Radzik '92):

Sei  $|x| = n$ , dann benötigt Newtons Verfahren  $O(n^2 \log^2 n)$  Iterationen um  $t^*$  zu finden.

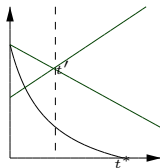
(iii) *Verfahren von Megiddo ('79)*

Voraussetzung: Es existiert ein kombinatorischer Algorithmus für (P), der nur Additionen, Subtraktionen und Vergleiche benötigt.

Grundidee: Bestimme  $v(t)$  allgemein für beliebiges  $t$ . Jeder Einzelwert in (P) wird zu linearer Funktion in  $v(t)$ .

Problem: Vergleiche, Ergebnis des Vergleichs hängt von  $t'$  ab.

Ziel: Verwende jenes Ergebnis des Vergleichs, das auch für  $t^*$  gilt.



Auswertung von  $v(t')$ :

$$v(t') < 0 \Rightarrow t^* < t' \Rightarrow \text{Vergleich links von } t'$$

$$v(t') > 0 \Rightarrow t^* > t' \Rightarrow \text{Vergleich rechts von } t'$$

Komplexität:

$A(n)$  für (P)  $\Rightarrow$  Laufzeit  $(A(n))^2$  für (F).

$A(n) = V(n) + R(n) \Rightarrow V(n)(V(n) + R(n)) + R(n)$

$\Rightarrow$  Gesamt:  $V(n) \cdot A(n)$

**Verbesserung 1** Führe ein Suchintervall  $[t_l, t_k]$  für  $t^*$  mit. Bei jedem Vergleich:

$$t' < t_l \Rightarrow v(t') > 0$$

$$t' > t_h \Rightarrow v(t') < 0$$

$t' \in [t_l, t_h] \Rightarrow v(t')$  berechnen und Intervall aktualisieren

**Verbesserung 2** Aufruf von  $A$  um  $v(t')$  auszuwerten aufschieben.

- (i) Wenn Ergebnis eines Vergleichs nicht sofort benötigt wird, Testwerte  $t'$  sammeln und dann binäre Suche auf den gesammelten  $t'$ -Werten durchführen.

**Bsp.:** (KP) hat Laufzeit  $nc$ , also hat das fraktionale Problem Laufzeit  $(nc)^2$ . Durch Aufschieb erreicht man eine Laufzeit von  $n(c + \log c(nc)) = n^2c \log c$ .

- (ii) Einige Zeit mit stückweise linearen Funktionen weiterrechnen.